Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Data-Based Nonaffine Optimal Tracking Control Using Iterative DHP Approach ⋆

Mingming Ha, * **Ding Wang,** **,*** **Derong Liu** ****

* *School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China (e-mail: hamingming_0705@foxmail.com.*
** *Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: dingwang@bjut.edu.cn)*
*** *Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing University of Technology, Beijing 100124, China*
**** *School of Automation, Guangdong University of Technology, Guangzhou 510006, China (e-mail: derong@gdut.edu.cn)*

Abstract: In this paper, a data-based optimal tracking control approach is developed by involving the iterative dual heuristic dynamic programming algorithm for nonaffine systems. In order to gain the steady control corresponding to the desired trajectory, a novel strategy is established with regard to the unknown system function. Then, according to the iterative adaptive dynamic programming algorithm, the updating formula of the costate function and the new optimal control policy for unknown nonaffine systems are provided to solve the optimal tracking control problem. Moreover, three neural networks are used to facilitate the implementation of the proposed algorithm. In order to improve the accuracy of the steady control corresponding to the desired trajectory, we employ a model network to directly approximate the unknown system function instead of the error dynamics. Finally, the effectiveness of the proposed method is demonstrated through a simulation example.

Keywords: Adaptive dynamic programming, data-based optimal tracking control, iterative dual heuristic dynamic programming, neural network.

1. INTRODUCTION

The optimal tracking control is a significant topic of the control community. Its objective is to make the controlled systems track the desired trajectories. Nowadays, adaptive dynamic programming (ADP) methods (Jiang and Zhang (2018); Liu et al. (2017); Liu et al. (2018); Wang et al. (2017); Wang et al. (2020); Wang and Liu (2018); Wei et al. (2017);) are widely used to solve optimal control problems. Some works (Kiumarsi and Lewis (2015); Luo et al. (2016); Qin et al. (2014); Wang et al. (2012); Zhang et al. (2008); Zhu et al. (2016)) have been reported to solve the tracking control problem by using ADP in the last decades. For the affine nonlinear system $x_{k+1} = f(x_k) + g(x_k)\mu_x(x_k)$, where x_k is the system state, $\mu_x(x_k)$ is the control input and $f(\cdot)$ and $g(\cdot)$ are system functions, there exist abundant works to solve the optimal tracking control problem. The heuristic dynamic programming (HDP) algorithm was employed to make the discrete-time nonlinear affine systems track the desired trajectories in Zhang et al. (2008). Meanwhile, the rigorous convergence analysis is provided. Additionally, Wang et al. (2012) investigated the finite-horizon neuro-optimal tracking control by transforming the controlled affine systems into the augmented systems. However, these two methods (Wang et al. (2012); Zhang et al. (2008)) need to establish the model of the error dynamics, which reduces the accuracy of the model. On the other hand, the system functions $f(\cdot)$ and $g(\cdot)$ need to be obtained to compute the steady control input $\mu_d(d_k)$ corresponding to the reference trajectory, such as $\mu_d(d_k) = g^+(d_k)(d_{k+1} - f(d_k))$, where d_k is the desired trajectory and $g^+(\cdot)$ is the generalized inverse of $g(\cdot)$. From then on, for affine systems with input constraints, the policy evaluation and the policy improvement are applied to solve the tracking problem by Kiumarsi and Lewis (2015). For continuous-time nonlinear systems, by using the reinforcement learning, Modares and Lewis (2014) trained the learning algorithm to learn the optimal tracking control input and studied the convergence and stability problems of whole the system. The model-free optimal tracking controller is designed by introducing the reinforcement learning in Luo et al. (2016), which learns the optimal tracking control policy from the real system data samples. Nevertheless, the proposed algorithm needs to be given an initial admissible control policy and a series of activation functions in neural networks need to be manually designed.

Almost all previous research works are aimed at the affine nonlinear systems. However, for nonaffine systems

⋆ This work was supported in part by Beijing Natural Science Foundation under Grant JQ19013, and in part by the National Natural Science Foundation of China under Grant 61773373 and Grant 61533017.

or unknown systems, few of study investigates the optimal tracking control problem. There are two main difficulties:

(1) How do we obtain $\mu_d(d_k)$ for unknown nonaffine systems?
(2) How do we solve the optimal control law in the iterative dual heuristic dynamic programming (DHP) scheme for nonaffine systems?

In this paper, we focus on these difficulties and solve the data-based optimal tracking control problem. Furthermore, some superior results are obtained. The rest paper is organized as follows. In Section 2, the tracking control problem is first transformed to the regulation problem by constructing a new augmented system. Then, a new optimal control policy and the new iterative DHP updating rules are provided in Section 3. In Section 4, the data-based iterative DHP algorithm and the solution of $\mu_d(d_k)$ are presented. Finally, a simulation example is used to verify the effectiveness of the proposed method.

2. PROBLEM STATEMENT

Consider the following nonaffine systems:

$$x_{k+1} = \mathcal{F}(x_k, \mu_x(x_k)), k \in \mathbb{N}, \tag{1}$$

where $x_k \in \mathbb{R}^n$ is the n-dimensional state vector, $\mu_x(x_k) \in \mathbb{R}^m$ is the m-dimensional control vector, $\mathbb{N} = \{0, 1, 2, \dots\}$ and $\mathcal{F}(\cdot)$ is differentiable with respect to its arguments. For the optimal tracking control problem, its objective is to find the optimal control policy $\mu_x^*(x_k)$, so as to make the nonaffine system track a desired trajectory. Next, we define the desired trajectory in the following form:

$$d_{k+1} = \kappa(d_k), \tag{2}$$

where $d_{k+1} \in \mathbb{R}^n$ and $\kappa(\cdot)\colon \mathbb{R}^n \to \mathbb{R}^n$ is a differentiable function in d_k. Define the tracking error vector as follows:

$$e_k = x_k - d_k. \tag{3}$$

In addition, we define the steady control corresponding to the desired trajectory as $\mu_d(d_k)$ and assume that the control input exists and satisfies

$$d_{k+1} = \mathcal{F}(d_k, \mu_d(d_k)). \tag{4}$$

$\mu_d(d_k)$ can be obtained by solving (4). In order to facilitate analysis, we assume that $\mu_d(d_k)$ can be denoted as follows:

$$\mu_d(d_k) = \varphi(d_k). \tag{5}$$

In other words, $\mu_d(d_k)$ is the solution of (4). We can obtain it by using the analytic method or various numerical methods.

Then, we define a new control input in the following:

$$\mu_e(e_k) = \mu_x(x_k) - \mu_d(d_k). \tag{6}$$

According to (1)–(6), we obtain a new augmented system as

$$\begin{cases} e_{k+1} = \mathcal{F}(e_k + d_k, \mu_e(e_k) + \varphi(d_k)) - \kappa(d_k) \\ d_{k+1} = \kappa(d_k). \end{cases} \tag{7}$$

Therefore, the augmented system (7) can be rewritten as

$$\mathcal{X}_{k+1} = \mathscr{F}(\mathcal{X}_k, \mu_e(e_k)), \tag{8}$$

where $\mathscr{F}\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$, $\mathcal{X}_k = [e_k^\mathsf{T}, d_k^\mathsf{T}]^\mathsf{T}$ and $\mu_e(e_k)$ are the $2n$-dimensional state vector and m-dimensional control input of the new system, respectively. Since $\mathcal{F}$ and κ are differentiable, $\mathscr{F}$ is also differentiable in its arguments. It is assumed that the system (8) is controllable on the set Ω and $\mathscr{F}$ is Lipschitz continuous on Ω.

In order to solve optimal tracking control problems, we define the following performance index and need to find a control sequence to minimize it

$$\mathcal{J}(\mathcal{X}_k, \mu_e(e_k)) = \sum_{l=k}^{\infty} \mathcal{U}(\mathcal{X}_l, \mu_e(e_l)), \tag{9}$$

where $\mathcal{U}(\mathcal{X}_l, \mu_e(e_l))$ is the positive definite utility function and satisfies $\mathcal{U}(0, 0) = 0$. Inspired by the works of Kiumarsi and Lewis (2015), Wang et al. (2012) and Zhang et al. (2008), the utility function is selected as follows:

$$\begin{aligned} \mathcal{U}(\mathcal{X}_l, \mu_e(e_l)) &= \begin{bmatrix} e_l^\mathsf{T} & d_l^\mathsf{T} \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e_l \\ d_l \end{bmatrix} + \mu_e^\mathsf{T}(e_l) R \mu_e(e_l) \\ &= e_l^\mathsf{T} Q e_l + \mu_e^\mathsf{T}(e_l) R \mu_e(e_l) \\ &= \mathcal{U}(e_l, \mu_e(e_l)), \end{aligned} \tag{10}$$

where $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ are the symmetric positive definite matrices.

In view of the form of (10), the performance index of the augmented system can be simplified as

$$\mathcal{J}(e_k, \mu(e_k)) = \sum_{l=k}^{\infty} \{e_l^\mathsf{T} Q e_l + \mu_e^\mathsf{T}(e_l) R \mu_e(e_l)\}. \tag{11}$$

Therefore, the main part of system (8) can be considered as:

$$e_{k+1} = \mathcal{G}(e_k, \mu_e(e_k)). \tag{12}$$

It is assumed that $\mathcal{G}(0, 0) = 0$ holds.

Furthermore, the performance index is rewritten as

$$\begin{aligned} \mathcal{J}(e_k, \mu_e(e_k)) &= e_k^\mathsf{T} Q e_k + \mu_e^\mathsf{T}(e_k) R \mu_e(e_k) \\ &\quad + \sum_{l=k+1}^{\infty} \{e_l^\mathsf{T} Q e_l + \mu_e^\mathsf{T}(e_l) R \mu_e(e_l)\} \\ &= e_k^\mathsf{T} Q e_k + \mu_e^\mathsf{T}(e_k) R \mu_e(e_k) \\ &\quad + \mathcal{J}(e_{k+1}, \mu_e(e_{k+1})). \end{aligned} \tag{13}$$

According to Bellman's optimality principle, the optimal value function $\mathcal{J}^*$ satisfies the following equation

$$\mathcal{J}^*(e_k) = \min_{\mu_e(e_k)} \{\mathcal{U}(e_k, \mu_e(e_k)) + \mathcal{J}^*(e_{k+1})\}. \tag{14}$$

The optimal control policy $\mu_e^*(e_k)$ should satisfy

$$\mu_e^*(e_k) = \arg \min_{\mu_e(e_k)} \{\mathcal{U}(e_k, \mu_e(e_k)) + \mathcal{J}^*(e_{k+1})\}. \tag{15}$$

Then, the optimal tracking control of the original system is given by

$$\mu_x^*(x_k) = \mu_d(d_k) + \mu_e^*(e_k), \tag{16}$$

where $\mu_d(d_k)$ is obtained by the equation (5).

3. THE OPTIMAL TRACKING CONTROL BASED ON ITERATIVE DHP ALGORITHM

This section includes two subsections. In the first subsection, the value iterative algorithm is elaborated and a new optimal control policy for nonaffine systems is obtained. Then, the novel iterative DHP algorithm for nonaffine tracking systems is derived in the second subsection.

3.1 Derivation of the Iterative ADP Algorithm

Due to the fact that it is difficult to solve the Bellman's equation directly, we present the value iteration algorithm to obtain the numerical solution.

We initialize the value function as $\mathcal{J}_0(\cdot) = 0$. The corresponding control input is denoted as follows:

$$\begin{aligned}\mu_{e0}(e_k) =& \arg\min_{\mu(e_k)}\{e_k^{\mathsf{T}}Qe_k + \mu_e^{\mathsf{T}}(e_k)R\mu_e(e_k) + \mathcal{J}_0(e_{k+1})\}\\ =& 0.\end{aligned} \tag{17}$$

Furthermore, the value function is updated as follows:

$$\begin{aligned}\mathcal{J}_1(e_k) =& \min_{\mu_k}\{e_k^{\mathsf{T}}Qe_k + \mu_e^{\mathsf{T}}(e_k)R\mu_e(e_k) + \mathcal{J}_0(e_{k+1})\}\\ =& e_k^{\mathsf{T}}Qe_k + \mu_{e0}^{\mathsf{T}}(e_k)R\mu_{e0}(e_k) + \mathcal{J}_0(e_{k+1})\}\\ =& e_k^{\mathsf{T}}Qe_k.\end{aligned} \tag{18}$$

Therefore, the value iteration algorithm can be implemented between the control policy

$$\mu_{ei}(e_k) = \arg\min_{\mu_e(e_k)}\{e_k^{\mathsf{T}}Qe_k + \mu_e^{\mathsf{T}}(e_k)R\mu_e(e_k) + \mathcal{J}_i(e_{k+1})\} \tag{19}$$

and the value function

$$\begin{aligned}\mathcal{J}_{i+1}(e_k) =& \min_{\mu_k}\{e_k^{\mathsf{T}}Qe_k + \mu_e^{\mathsf{T}}(e_k)R\mu_e(e_k) + \mathcal{J}_i(e_{k+1})\}\\ =& e_k^{\mathsf{T}}Qe_k + \mu_{ei}^{\mathsf{T}}(e_k)R\mu_{ei}(e_k) + \mathcal{J}_i(\mathcal{G}(e_k, \mu_{ei}(e_k))),\end{aligned} \tag{20}$$

where $i = 1, 2, \ldots$.

It is noteworthy that how we obtain the solution of (19) for nonaffine systems. For affine systems, the optimal control law can be gotten by letting $\partial(e_k^{\mathsf{T}}Qe_k + \mu_e^{\mathsf{T}}(e_k)R\mu_e(e_k) + \mathcal{J}_i(e_{k+1}))/\partial e_k = 0$. However, it is not available for nonaffine systems. Therefore, it is necessary to develop a novel method to obtain the optimal control policy. We use the gradient-based method to find $\mu_{ei}(e_k)$ and minimize $\mathcal{J}_{i+1}(e_k)$. First, we randomly initialize $\mu_{ei}(e_k)$ as $\mu_{ei}^{(0)}(e_k)$. Then, the updating rule of $\mu_{ei}(e_k)$ is a gradient-based adaptation rule formulated by

$$\begin{aligned}\mu_{ei}^{(j+1)}(e_k) =& \mu_{ei}^{(j)}(e_k) - \alpha_\mu\frac{\partial(e_k^{\mathsf{T}}Qe_k + \mu_{ei}^{\mathsf{T}}(e_k)R\mu_{ei}(e_k))}{\partial\mu_{ei}(e_k)}\\ &- \alpha_\mu\frac{\partial\mathcal{J}_i(e_{k+1}))}{\partial\mu_{ei}(e_k)}\\ =& \mu_{ei}^{(j)}(e_k) - 2\alpha_\mu R\mu_{ei}^{(j)}(e_k)\\ &- \alpha_\mu\left(\frac{\partial e_{k+1}}{\partial\mu_{ei}^{(j)}(e_k)}\right)^{\mathsf{T}}\frac{\partial\mathcal{J}_i(e_{k+1})}{\partial e_{k+1}},\end{aligned} \tag{21}$$

where $\alpha_\mu \in (0, 1)$ is the learning rate with respect to $\mu_e(e_k)$ and j, unlike i, is the iteration step of (21).

Next, by introducing a theorem, we will discuss how to obtain $\partial e_{k+1}/\partial\mu_{ei}^{(j)}(e_k)$ in (21) without modeling the error dynamics.

Theorem 1. Define the control input of the original system as $\mu_x(x_k)$ in (1) and the control input of the new system as $\mu_e(e_k)$ in (12) , then $\partial e_{k+1}/\partial\mu_e(e_k)$ and $\partial e_{k+1}/\partial e_k$ satisfy the following equations:

$$\frac{\partial e_{k+1}}{\partial\mu_e(e_k)} = \frac{\partial x_{k+1}}{\partial\mu_x(x_k)}, \frac{\partial e_{k+1}}{\partial e_k} = \frac{\partial x_{k+1}}{\partial x_k}. \tag{22}$$

Proof. To the best of our knowledge, according to (4), $\varphi(d_k)$ is related to d_k and the system function $\mathcal{F}(\cdot)$. Also, according to the augmented system (7), the term $\partial e_{k+1}/\partial\mu_e(e_k)$ satisfies the following condition:

$$\begin{aligned}\frac{\partial e_{k+1}}{\partial\mu_e(e_k)} =& \frac{\partial(\mathcal{F}(e_k + d_k, \mu_e(e_k) + \varphi(d_k)) - \kappa(d_k))}{\partial\mu_e(e_k)}\\ =& \frac{\partial\mathcal{F}(e_k + d_k, \mu_e(e_k) + \varphi(d_k))}{\partial(\mu_e(e_k) + \varphi(d_k))}.\end{aligned} \tag{23}$$

On the other hand, based on the original system (1), we have

$$\begin{aligned}\frac{\partial x_{k+1}}{\partial\mu_x(x_k)} =& \frac{\partial\mathcal{F}(x_k, \mu_x(x_k))}{\partial\mu_x(x_k)}\\ =& \frac{\partial\mathcal{F}(e_k + d_k, \mu_e(e_k) + \varphi(d_k))}{\partial(\mu_e(e_k) + \varphi(d_k))}.\end{aligned} \tag{24}$$

Then, $\partial e_{k+1}/\partial\mu_e(e_k) = \partial x_{k+1}/\partial\mu_x(x_k)$ holds. Similarly, $\partial e_{k+1}/\partial e_k = \partial x_{k+1}/\partial x_k$ also holds.

The proof is completed.

According to (21) and Theorem 1, the updating rule of $\mu_{ei}(e_k)$ can be rewritten as follows:

$$\begin{aligned}\mu_{ei}^{(j+1)}(e_k) =& \mu_{ei}^{(j)}(e_k) - 2\alpha_\mu R\mu_{ei}^{(j)}(e_k)\\ &- \alpha_\mu\left(\frac{\partial x_{k+1}}{\partial(\mu_{ei}^{(j)}(e_k) + \varphi(d_k))}\right)^{\mathsf{T}}\frac{\partial\mathcal{J}_i(e_{k+1})}{\partial e_{k+1}}.\end{aligned} \tag{25}$$

3.2 Derivation of Iterative DHP Algorithm

First, we assume that the value function $\mathcal{J}_i(e_k)$ is smooth so that $\partial\mathcal{J}_i(e_k)/\partial e_k$ exists. According to (25), we find that the control law $\mu_i(e_k)$ at each step of iteration has to be computed by $\partial\mathcal{J}_i(e_{k+1})/\partial e_{k+1}$, which is not an easy task. Therefore, in the following, we will present the iterative DHP algorithm to implement the iterative ADP algorithm. Define the costate function as follows:

$$\lambda_i(e_k) = \frac{\partial\mathcal{J}_i(e_k)}{\partial e_k}. \tag{26}$$

First, we start with an initial costate function $\lambda_0(\cdot) = 0$. Then, for $\lambda_{i+1}(e_k) = \partial\mathcal{J}_{i+1}(e_k)/\partial e_k$, according to (20), the following equation can be deduced

$$\begin{aligned}\lambda_{i+1}(e_k) =& \frac{\partial\mathcal{U}(e_k, \mu_{ei}(e_k))}{\partial e_k} + \frac{\partial\mathcal{J}_i(e_{k+1})}{\partial e_k}\\ =& \frac{\partial e_k^{\mathsf{T}}Qe_k}{\partial e_k} + \left[\frac{\partial\mu_{ei}(e_k)}{\partial e_k}\right]^{\mathsf{T}}\frac{\partial\mu_{ei}^{\mathsf{T}}(e_k)R\mu_{ei}(e_k)}{\partial\mu_{ei}(e_k)}\\ &+ \left[\frac{\partial e_{k+1}}{\partial e_k}\right]^{\mathsf{T}}\frac{\partial\mathcal{J}_i(e_{k+1})}{\partial e_{k+1}}\\ &+ \left[\frac{\partial\mu_{ei}(e_k)}{\partial e_k}\right]^{\mathsf{T}}\left[\frac{\partial e_{k+1}}{\partial\mu_{ei}(e_k)}\right]^{\mathsf{T}}\frac{\partial\mathcal{J}_i(e_{k+1})}{\partial e_{k+1}}\\ =& 2Qe_k + 2\left[\frac{\partial\mu_{ei}(e_k)}{\partial e_k}\right]^{\mathsf{T}}R\mu_{ei}(e_k)\\ &+ \left[\frac{\partial e_{k+1}}{\partial e_k} + \frac{\partial e_{k+1}}{\partial\mu_{ei}(e_k)}\frac{\partial\mu_{ei}(e_k)}{\partial e_k}\right]^{\mathsf{T}}\lambda_i(e_{k+1}).\end{aligned} \tag{27}$$

Next, using Theorem 1, we eventually obtain

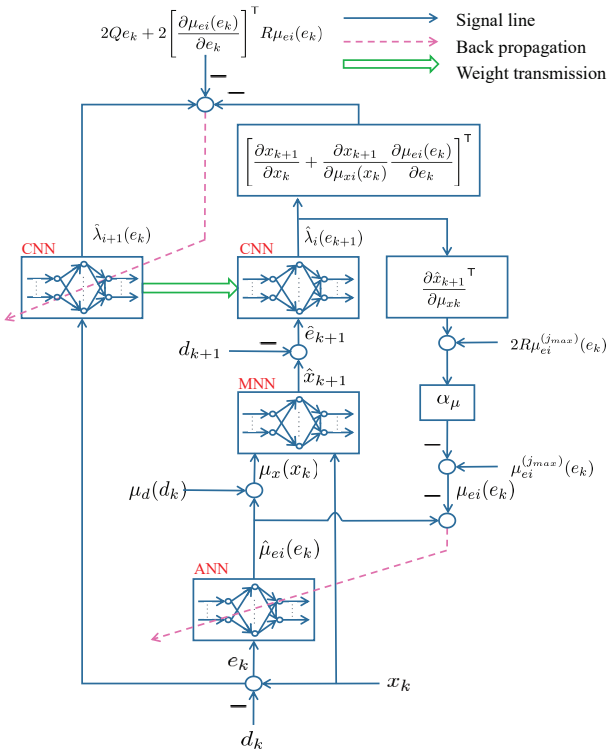

Fig. 1. The flowchart of the optimal tracking control algorithm

$$\lambda_{i+1}(e_k) = 2Qe_k + 2\left[\frac{\partial \mu_{ei}(e_k)}{\partial e_k}\right]^{\mathsf{T}} R\mu_{ei}(e_k) + \left[\frac{\partial x_{k+1}}{\partial x_k} + \frac{\partial x_{k+1}}{\partial \mu_{xi}(x_k)}\frac{\partial \mu_{ei}(x_k)}{\partial e_k}\right]^{\mathsf{T}} \lambda_i(e_{k+1}). \tag{28}$$

Therefore, in the iterative DHP algorithm, the costate function sequence $\{\lambda_i\}$ and the control sequence $\{\mu_{ei}\}$ are updated by implementing the iteration between (25) and (28). From (25) and (28), the control policy can directly be computed by the costate function and the costate function can be obtained by solving the system model rather than the error dynamic model.

4. DATA-BASED ITERATIVE DHP IMPLEMENTATION

Three subsection are included in this section, namely, the model neural network (MNN), the critic neural network (CNN) and the action neural network (ANN). First, the model network is used to approximate the unknown nonaffine system and the new approach to obtain $\mu_d(d_k)$ is introduced. Second, the training process of the critic network is shown. Finally, the construction of the action network is elaborated in the last subsection. On the other hand, the flowchart of the proposed algorithm is displayed in Fig. 1.

4.1 The Model Network

For unknown nonaffine systems, the model network needs to be established to estimated the system state. We employ the three-layer neural network to construct the model network. The output of the model network is formulated as follows

$$\hat{x}_{k+1} = w_{m2}^{\mathsf{T}}\delta(w_{m1}^{\mathsf{T}}x_{mk} + b_1) + b_2, \tag{29}$$

where w_{m1} and w_{m2} are the weight matrices, b_1 and b_2 are threshold vectors, $x_{mk} = [x_k^{\mathsf{T}}, \mu_x^{\mathsf{T}}(x_k)]^{\mathsf{T}}$ and $\delta(\cdot)$ is the activation function.

Define the approximate error as

$$\mathcal{E}_m = \frac{1}{2}(\hat{x}_{k+1} - x_{k+1})^{\mathsf{T}}(\hat{x}_{k+1} - x_{k+1}). \tag{30}$$

The gradient descent algorithm is used to update weights and thresholds. The updating rules are denoted as follows:

$$\begin{aligned} w_{m1} &:= w_{m1} - \theta\left[\frac{\partial \mathcal{E}_m}{\partial w_{m1}}\right], \\ w_{m2} &:= w_{m2} - \theta\left[\frac{\partial \mathcal{E}_m}{\partial w_{m2}}\right], \\ b_1 &:= b_1 - \theta\left[\frac{\partial \mathcal{E}_m}{\partial b_1}\right], \\ b_2 &:= b_2 - \theta\left[\frac{\partial \mathcal{E}_m}{\partial b_2}\right], \end{aligned} \tag{31}$$

where $\theta \in (0,1)$ is the learning rate and the symbol $:=$ denotes the assignment operation. When the training process is finished, weights and thresholds are not updated.

Since the system function is unknown, it is difficult to solve (4). Therefore, the expression of the model network is utilized to obtain $\mu_d(d_k)$. (4) can be rewritten as follows:

$$\hat{d}_{k+1} = w_{m2}^{\mathsf{T}}\delta(w_{m1}^{\mathsf{T}}d_{mk} + b_1) + b_2, \tag{32}$$

where $d_{mk} = [d_k^{\mathsf{T}}, \mu_d^{\mathsf{T}}(d_k)]^{\mathsf{T}}$. Then, the various numerical methods can be applied to obtain solution of (32). It is noteworthy that the model performance directly determines the accuracy of $\mu_d(d_k)$. Therefore, thresholds are added in the model network to improve the modeling precision.

4.2 The Critic Network

According to the iterative DHP algorithm, the critic network is used to approximate the costate function. The input of the critic network is the tracking error vector e_k. The output is denoted as

$$\hat{\lambda}_{i+1}(e_k) = w_{c2}^{\mathsf{T}}\delta(w_{c1}^{\mathsf{T}}e_k). \tag{33}$$

Define the approximation error as follows:

$$\mathcal{E}_c = \frac{1}{2}(\hat{\lambda}_{i+1}(e_k) - \lambda_{i+1}(e_k))^{\mathsf{T}}(\hat{\lambda}_{i+1}(e_k) - \lambda_{i+1}(e_k)), \tag{34}$$

where $\lambda_{i+1}(e_k)$ is computed by (28).

Similarly, the updating rules of the weight matrices are given by using the gradient descent algorithm

$$\begin{aligned} w_{c1} &:= w_{c1} - \eta\left[\frac{\partial \mathcal{E}_c}{\partial w_{c1}}\right], \\ w_{c2} &:= w_{c2} - \eta\left[\frac{\partial \mathcal{E}_c}{\partial w_{c2}}\right], \end{aligned} \tag{35}$$

where $\eta \in (0,1)$ is the learning rate of the critic network.

4.3 The Action Network

In the action network, the output is formulated as

$$\hat{\mu}_{ei}(e_k) = w_{a2}^{\mathsf{T}}\delta(w_{a1}^{\mathsf{T}}e_k). \tag{36}$$

Define the approximation error in the following:

$$\mathscr{E}_a = \frac{1}{2}(\hat{\mu}_{ei}(e_k) - \mu_{ei}(e_k))^{\mathsf{T}}(\hat{\mu}_{ei}(e_k) - \mu_{ei}(e_k)). \tag{37}$$

The optimal control policy $\mu_{ei}(e_k)$ at each iteration step can be obtained through iteration updating. According to (25) and (26), the updating rule is given as follows

$$\begin{aligned}\mu_{ei}^{(j+1)}(e_k) = \mu_{ei}^{(j)}(e_k) - 2\alpha_\mu R\mu_{ei}^{(j)}(e_k) \\ - \alpha_\mu\left(\frac{\partial x_{k+1}}{\partial(\mu_{ei}(e_k)+\varphi(d_k))}\right)^{\mathsf{T}}\lambda_i(e_{k+1}).\end{aligned} \tag{38}$$

The updating rules for weights are obtained

$$\begin{aligned} w_{a1} &:= w_{a1} - \zeta\left[\frac{\partial\mathscr{E}_a}{\partial w_{a1}}\right], \\ w_{a2} &:= w_{a2} - \zeta\left[\frac{\partial\mathscr{E}_a}{\partial w_{a2}}\right], \end{aligned} \tag{39}$$

where $\zeta \in (0,1)$ is the learning rate of the action network.

5. SIMULATION STUDIES

In this section, the performance of the proposed tracking algorithm is demonstrated through a simulation example. The example derived from Luo et al. (2016) with some modifications is considered as

$$\begin{aligned}x_{k+1} = &\begin{bmatrix}\tanh(x_{1k}) + 0.05\tanh(x_{2k}) \\ -0.3\tanh(x_{1k}) + \tanh(x_{2k})\end{bmatrix} \\ &+ \begin{bmatrix}0 \\ \sin(u_{sk})\end{bmatrix},\end{aligned} \tag{40}$$

where $x_0 = [0.9, -0.8]^{\mathsf{T}}$.

First, the model network needs to be constructed. In order to improve the accuracy of the model network, we set the number of hidden layer neurons as 40. In the implementations, we use the MATLAB neural network toolbox. The learning rate is $\theta = 0.02$. Additionally, the initial values of weight matrices and threshold vectors are set by default. Then, 1000 data samples generated by the nonaffine system are used to train the model network for 150 iteration steps. Fig. 2 demonstrates the performance of the model network by using 500 data samples to test. Here, training and testing data samples are randomly selected in $x \in [-1,1]$ and $\mu_x \in [-1,1]$. In this example, we employ the performance error measure as $e_m = \text{abs}(\hat{x}_{k+1} - x_{k+1})$ to help us clearly verify the performance of the model network, where abs$(\cdot)$ denotes the absolute values of elements in $\hat{x}_{k+1} - x_{k+1}$.

Next, we define the desired trajectory as follows:

$$d_{k+1} = \begin{bmatrix}0.9963d_{1k} + 0.0498d_{2k} \\ -0.2492d_{1k} + 0.9888d_{2k}\end{bmatrix}, \tag{41}$$

where $d_0 = [0.1, 0.2]^{\mathsf{T}}$. We use the proposed method to make the unknown system (40) track the desired trajectory. For this purpose, we construct the critic and action networks with structures 2–8–2 and 2–8–1, correspondingly. The learning rates are set as $\eta = \zeta = 0.05$ and the

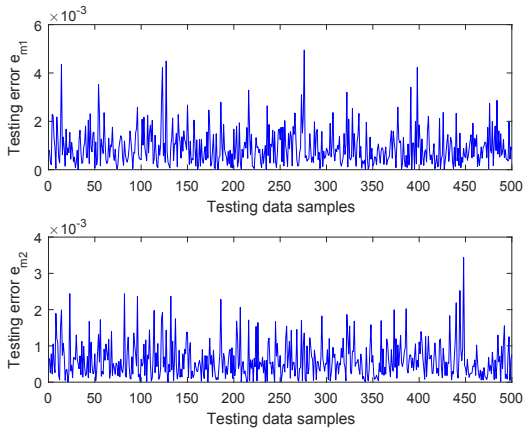

Fig. 2. The testing error of the model network

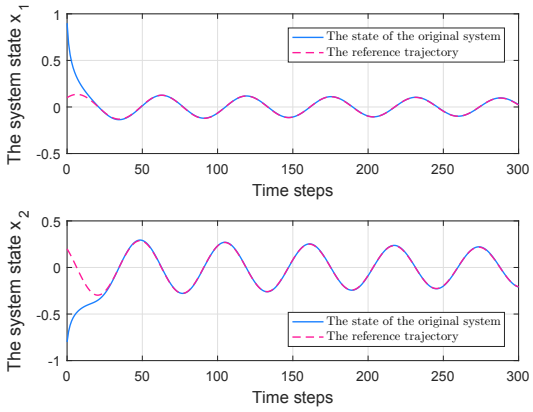

Fig. 3. The state trajectory of the original system and the reference trajectory

weights are initialized randomly. In the DHP algorithm, the weight matrices in the utility function are chosen as $Q = 0.1I_{2\times2}$ and $R = I_{1\times1}$ and the learning rate in the updating rule of $\mu_{ei}(e_k)$ is also set as $\alpha_\mu = 0.05$.

After learning is completed, the action network is regraded as the tracking controller to control the nonaffine system (40). The convergence curves of the state and the tracking error are shown in Figs. 3 and 4, respectively. The results demonstrate that the state vector reaches the desired trajectory fast, within 30 time steps. The curves of the control inputs corresponding to the original and augmented system are shown in Fig. 5.

Additionally, it is worth mentioning that $\mu_d(d_k)$ needs to be obtained in the learning process. According to the approach described in Section 4.1, the steady control corresponding to the desired trajectory can be computed by the expression of the model network. Here, we employ the function "fsolve" in MATLAB to solve $\mu_d(d_k)$. The curves of $\mu_d(d_k)$ by solving (4) and (32) are displayed in Fig. 6, which verifies the effectiveness of the developed approach.

6. CONCLUSION

In this paper, for unknown nonaffine systems, the new updating rule of the costate function based on DHP is

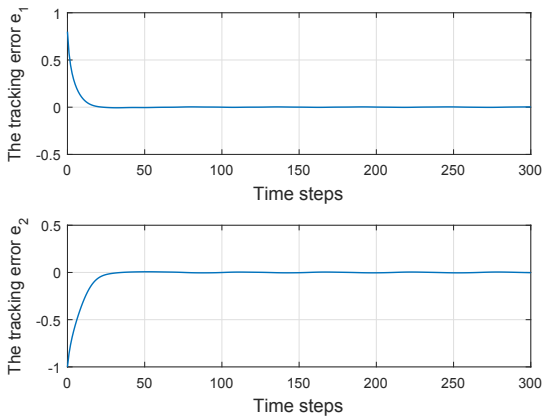

Fig. 4. The tracking error

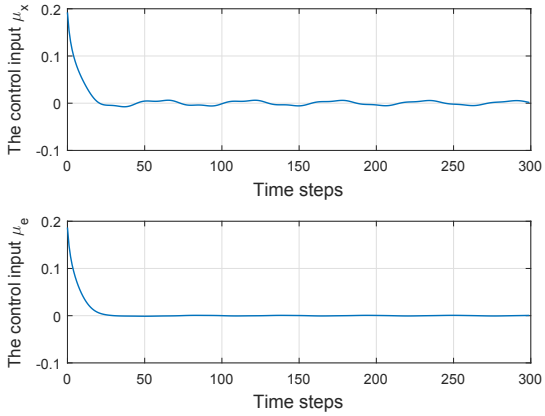

Fig. 5. The control input of the original system and the augmented system

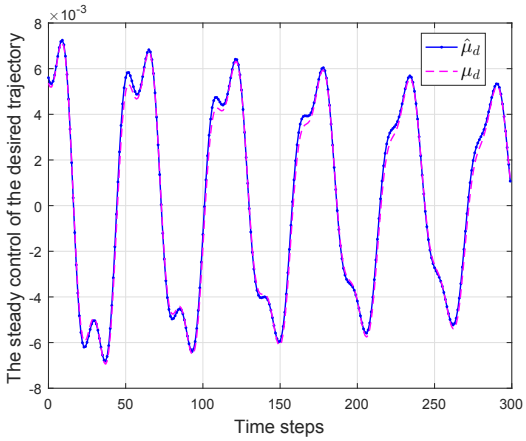

Fig. 6. The steady control input corresponding to the desired trajectory of the proposed approach and its real value

proposed. Additionally, the steady control corresponding to the desired trajectory is obtained by solving the expression of the model network. It should be noted that the developed approach to estimate $\mu_d(d_k)$ is actually a model-based technique. On the other hand, the optimal control $\mu_{ei}(e_k)$ at each iteration step for nonaffine systems is given. The simulation example also verifies the performance of the proposed methods. However, $\mu_d(d_k)$ requires the high-precision model network, which directly influence the effectiveness of the tracking controller. We will make further discussion in future works.

REFERENCES

Jiang, H. and Zhang, H. (2018). Iterative adp learning algorithms for discrete-time multi-player games. *Artificial Intelligence Review*, 50, 75–91.

Kiumarsi, B. and Lewis, F.L. (2015). Actor-critic-based optimal tracking for partially unknown nonlinear discrete-time systems. *IEEE Transactions on Neural Networks and Learning Systems*, 26, 140–151.

Liu, D., Wei, Q., Wang, D., Yang, X., and Li, H. (2017). *Adaptive Dynamic Programming with Applications in Optimal Control*. Springer, Cham, New York.

Liu, D., Xu, Y., and Wei, Q. (2018). Residential energy scheduling for variable weather solar energy based on adaptive dynamic programming. *IEEE/CAA Journal of Automatica Sinica*, 5, 36–46.

Luo, B., Liu, D., Huang, T., and Wang, D. (2016). Model-free optimal tracking control via critic-only q-learning. *IEEE Transactions on Neural Networks and Learning Systems*, 27, 2134–2144.

Modares, H. and Lewis, F.L. (2014). Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica*, 50, 1780–1792.

Qin, C., Zhang, H., and Luo, Y. (2014). Online optimal tracking control of continuous-time linear systems with unknown dynamics by using adaptive dynamic programming. *International Journal of Control*, 87, 1000–1009.

Wang, D., Ha, M., and Qiao, J. (2020). Self-learning optimal regulation for discrete-time nonlinear systems under event-driven formulation. *IEEE Transactions on Automatic Control*, 65, 1272–1279.

Wang, D., He, H., Zhong, X., and Liu, D. (2017). Event-driven nonlinear discounted optimal regulation involving a power system application. *IEEE Transactions on Industrial Electronics*, 64, 8177–8186.

Wang, D. and Liu, D. (2018). Learning and guaranteed cost control with event-based adaptive critic implementation. *IEEE Transactions on Neural Networks and Learning Systems*, 29, 6004–6014.

Wang, D., Liu, D., and Wei, Q. (2012). Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach. *Neurocomputing*, 78, 14–22.

Wei, Q., Liu, D., Liu, Y., and Song, R. (2017). Optimal constrained self-learning battery sequential management in microgrid via adaptive dynamic programming. *IEEE/CAA Journal of Automatica Sinica*, 4, 168–176.

Zhang, H., Wei, Q., and Luo, Y. (2008). 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.*, 38, 937–942.

Zhu, Y., Zhao, D., and Li, X. (2016). Using reinforcement learning techniques to solve continuous-time non-linear optimal tracking problem without system dynamics. *IET Control Theory and Applications*, 10, 1339–1347.