mathematics

MDPI

Article

Near-Optimal Tracking Control of Partially Unknown Discrete-Time Nonlinear Systems Based on Radial Basis Function Neural Network

Jiashun Huang [1], Dengguo Xu [1,2,*], Yahui Li [1] and Yan Ma [1]

[1] School of Automation, Guangxi University of Science and Technology, Liuzhou 545000, China; 18707519318@163.com (J.H.); lyh7ooo@163.com (Y.L.); mayan_yyjj@163.com (Y.M.)
[2] School of Physics and Electrical Engineering, Liupanshui Normal University, Liupanshui 553000, China
* Correspondence: dengguoxu@163.com

Abstract: This paper proposes an optimal tracking control scheme through adaptive dynamic programming (ADP) for a class of partially unknown discrete-time (DT) nonlinear systems based on a radial basis function neural network (RBF-NN). In order to acquire the unknown system dynamics, we use two RBF-NNs; the first one is used to construct the identifier, and the other one is used to directly approximate the steady-state control input, where a novel adaptive law is proposed to update neural network weights. The optimal feedback control and the cost function are derived via feedforward neural network approximation, and a means of regulating the tracking error is proposed. The critic network and the actor network were trained online to obtain the solution of the associated Hamilton–Jacobi–Bellman (HJB) equation within the ADP framework. Simulations were carried out to verify the effectiveness of the optimal tracking control technique using the neural networks.

Keywords: adaptive dynamic programming (ADP); optimal tracking control; RBF neural network (RBF-NN); nonlinear systems

MSC: 93C10; 49L20; 49L12

Citation: Huang, J.; Xu, D.; Li, Y.; Ma, Y. Near-Optimal Tracking Control of Partially Unknown Discrete-Time Nonlinear Systems Based on Radial Basis Function Neural Network. *Mathematics* **2024**, *12*, 1146. https://doi.org/10.3390/math12081146

Academic Editors: Alicia Cordero and Asier Ibeas

Received: 28 February 2024
Revised: 5 April 2024
Accepted: 8 April 2024
Published: 10 April 2024

1. Introduction

As is widely known, nonlinear system control is an important topic of control fields, especially for discrete-time nonlinear systems, and is difficult for traditional control methods. In recent decades, many different approaches to discrete-time system control have been proposed, such as adaptive control [1], fuzzy control [2], and PID control [3]. Optimal tracking control, one of the effective methods for nonlinear systems, has many practical engineering applications [4–6]. Its purpose is to design a control law that not only allows the system to track the desired trajectory but also minimizes a specific performance index. It is of great theoretical significance to explore the optimal tracking optimal control of nonlinear systems. Although dynamic programming is an effective method for solving optimal control problems, there is the problem of "curse of dimensionality" when dealing with relatively complex systems [7,8]. Moreover, it is difficult to solve the HJB equation derived from the optimal control of nonlinear systems, which has no analytical solution.

On the other hand, neural network control is used as a common control method for uncertainly nonlinear systems. In 1990, Narendra et al. first proposed an artificial neural network (ANN) adaptive control method for nonlinear dynamical systems [9]. Through neural network approximation, the uncertain system can be reconstructed using input and output data. Since then, multilayer neural networks (MNNs) have been successfully applied in pattern recognition and control systems [10]. This also has led to the generation of many types of neural networks, including the RBF-NN. In [11], Poggio et al. first proved that the RBF-NN is superior in approximating functions. Studies of RBF-NNs

have also shown that these neural networks have the ability to approximate any nonlinear function with a compact ensemble and arbitrary accuracy [12,13]. Compared to other ANNs, the RBF-NN does not have the complex structure of neural networks such as back propagation (BP) networks or recurrent neural networks (RNNs), and it is easier to select parameters [11,14,15]. Its good generalization ability, simple network structure, and avoidance of unnecessarily lengthy computations are advantages that make RBF-NNs attract attention [15,16]. Many research results have been published on neural network control for nonlinear systems [17–19].

Benefiting from neural networks and reinforcement learning (RL), the difficult problem of solving nonlinear HJB partial differential equations is solved. The ADP algorithm was proposed by Powell to approximate the solution of the HJB equation [20], which combines the theory and methods of RL, neural networks, adaptive control and optimal control. As developed, ADP has not only been considered as one of the core methods for solving the diversity of optimal control problems but also has been successfully applied to both continuous-time systems [21–23] and discrete-time systems [24–31] to search for solutions of the HJB equations online. Particularly, several works have attempted to solve the discrete time nonlinear optimal regulation problem using the ADP algorithm such as robust ADP [32–35], iterative/invariant ADP [36–39], off-policy RL [40–42] and the Q-Learning Algorithm [40,43].

In the past decades, many relevant studies have been conducted on the optimal tracking control of discrete-time nonlinear system using the ADP algorithm. However, in the existing literature on optimal tracking of nonlinear discrete-time systems, there is no RBF neural network-based ADP algorithm. In this paper, an optimal tracking control method based on RBF-NNs for discrete-time partially unknown nonlinear systems is proposed. Two RBF neural networks are used to approximate the unknown system dynamic as well as the steady-state control. After transforming the tracking problem into a regulation problem, the critic network and the actor network are used to obtain the nearly optimal feedback control, which allows the online learning process to require only current and past system data.

The contributions of article are as follows: (1) Unlike the classical technique of NN approximation, we propose a near-optimal tracking control scheme for a class of partially unknown discrete-time nonlinear systems based on RBF-NNs and prove the stability of the system. (2) Compared with [35,39], we additionally used an RBF-NN to directly approximate the steady-state controller of the unknown system. It can solve the requirement for the priori knowledge of the controlled system dynamics and reference system dynamics. Moreover, we propose a novel adaptive law to update the weight of the steady-state controller.

The paper is organized as follows. The problem statement is shown in Section 2. The design of the optimal tracking controller of the system with partially unknown nonlinear dynamics is given in Section 3, which includes the RBF-NN identifier, the RBF-NN steady-state controller, near optimal feedback controller, and stability analysis. Section 4 provides simulation results to validate the proposed control method and details the method comparison. Section 5 draws some conclusions.

2. Problem Statement

Consider the following affine nonlinear discrete-time system [31]:

$$x(k+1) = f[x(k)] + g[x(k)]u(k) \tag{1}$$

where $x(k) \in \mathbb{R}^n$ is the measurable system state and $u(k) \in \mathbb{R}^m$ is the control input. Assume that the nonlinear smooth function $f[x(k)] \in \mathbb{R}^n$ is an unknown drift function, $g[x(k)] \in \mathbb{R}^{n\times m}$ is a known function, and $\|g[x(k)]\|_F \leq g_1$ where the Frobenius norm $\|\cdot\|_F$ is applied. In addition, assume that $g[x(k)]$ has a generalized inverse matrix $g[x(k)]^+ \in \mathbb{R}^{m\times n}$ such that $g[x(k)]g[x(k)]^+ = I \in \mathbb{R}^{n\times n}$ where I is the identity matrix. Let $x(0)$ be the initial state.

The reference trajectory is generated by the following bounded command:

$$x_d(k+1) = \varphi(x_d(k)) \tag{2}$$

where $x_d(k) \in \mathbb{R}^n$ and $\varphi(x_d(k)) \in \mathbb{R}^n$, and $x_d(k)$ is the reference trajectory; it need only be a stable state trajectory or asymptotically stable.

The goal of this paper is to design a controller $u(k)$ that not only ensures the state of the system (1) tracks the reference trajectory but also minimizes a cost function. For the optimal tracking control technique, the cost functions are usually considered in quadratic form [4], that is

$$J(e(k), u(k)) = \sum_{k=0}^{\infty} e^T(k) Q e(k) + u^T(k) R u(k) \tag{3}$$

where $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ are symmetric positive definite; $e(k) = x(k) - x_d(k)$ is tracking error. For common solutions of tracking problems, the control input consists of two parts, a steady-state input u_d and a feedback input u_e [24]. Next, we will discuss how to obtain each part.

The steady-state controller is used to ensure perfect tracking. This perfect tracking equation is realized under the condition $x(k) = x_d(k)$. For this condition to be fulfilled, the steady-state part of the control $u_d(k)$ must exist to make $x(k)$ equivalent to $x_d(k)$. By substituting $x_d(k)$ and $u_d(k)$ into system (1), the reference state is

$$x_d(k+1) = f[x_d(k)] + g[x_d(k)] u_d(k) \tag{4}$$

where $x_d(k)$ and $x_d(k+1)$ are bounded to be tracked by the reference trajectory. If the system dynamics (1) are known, $u_d(k)$ is acquired by

$$u_d(k) = g[x_d(k)]^+ (x_d(k+1) - f[x_d(k)]) \tag{5}$$

where $g[x_d(k)]^+ = (g[x_d(k)]^T g[x_d(k)])^{-1} g[x_d(k)]^T$ is the generalized inverse of $g[x_d(k)]$ with $g[x_d(k)]^+ g[x_d(k)] = I$.

Remark 1. *In the subsequent discussion, the RBF network can be used to identify the unknown dynamics of system (1); hence, (5) can be computed.*

By using (1) and (4), the tracking error dynamics $e(k)$ are given by

$$\begin{aligned} e(k+1) &= f[x(k)] + g[x(k)] u(k) - x_d(k+1) \\ &= f_e(k) + g_e(k) u_e(k) \end{aligned} \tag{6}$$

where $f_e(k) = g(e(k) + x_d(k)) g(x_d(k))^+ (\varphi(x_d(k)) - f(x_d(k))) + f(e(k) + x_d(k)) - \varphi(x_d(k))$, $u_e(k) = u(k) - u_d(k)$, and $g_e(k) = g[x_d(k) + e(k)]$. $u_e(k) \in \mathbb{R}^m$ is the feedback control input. By minimizing the cost function, it is designed to stabilize the tracking error dynamics. For $e(k)$ in the control sequence, the cost function can be expressed as the following discrete time tracking HJB equation

$$\begin{aligned} J_e(e(k), u_e(k)) &= \sum_{k=0}^{\infty} e^T(k) Q e(k) + u_e^T(k) R u_e(k) \\ &= e^T(k) Q e(k) + u_e^T(k) R u_e(k) + J_e(e(k+1), u_e(k+1)) \\ &= r(k) + J_e(e(k+1), u_e(k+1)) \end{aligned} \tag{7}$$

where $r(k) = e^T(k) Q e(k) + u_e^T(k) R u_e(k)$, $J_e(e(k), u_e(k)) > 0$ for $\forall e(k), u_e(k) \neq 0$ and $J_e(e(k+1), u_e(k+1))$ denotes the cost function at the next tracking error dynamics $e(k+1)$. The tracking error $e(k)$ is used in the study of the cost function of the optimal tracking control problem.

In general, this feedback control $u_e(k)$ is found by minimizing (7) to solve the extremum condition in the optimal control framework [4]. This result is

$$u_e^*(k) = -\frac{1}{2}R^{-1}g_e(k)\frac{\partial J(e(k+1))}{\partial e(k+1)} \tag{8}$$

Then, the standard control input is obtained

$$u^*(k) = u_d(k) + u_e^*(k) \tag{9}$$

where $u_d(k)$ is obtained from (5), and $u_e^*(k)$ is obtained from (8).

As detailed in the subsequent discussion, in order to acquire the unknown dynamics in system (1), we used the RBF neural networks to reconstruct system dynamics. Moreover, faced with the problem of unable to find the analytical solution of (7) and the curse of dimensionality, the ADP algorithm was used to approximately solve the HJB Equation (7).

The main results of this paper are based on the following definitions and assumptions [30].

Definition 1. *A control law u_e is admissible with respect to (7) on the set Ω if u_e is continuous on a compact set $\Omega_u \in \mathbb{R}$ for $\forall e(k) \in \Omega$, $u_e(0) = 0$, and $J(e(0), u_e(\cdot))$ is finite.*

Assumption 1. *System (1) is controllable, and the system state $x(k) = 0$ is in equilibrium under control $u(k) = 0$. Input control $u(k) = u(x(k))$ satisfies $u(x(k)) = 0$ for $x(k) = 0$, and the cost function is a positive definite function for any $x(k)$ and $u(k)$.*

Lemma 1. *For the tracking error system (6), assume that $u_e(k)$ is an admissible control, the internal dynamics $f_e(k)$ is bounded, and*

$$\|f_e(k)\|^2 \leq \Gamma\lambda_{\min}(Q)\|e(k)\|^2/2 + (\Gamma\lambda_{\min}(R) - 2g_1^2)\|u_e(k)\|^2/2, \tag{10}$$

where $\lambda_{\min}(R)$ is the minimum eigenvalue of R, $\lambda_{\min}(Q)$ is the minimum eigenvalue of Q, and $\Gamma > 2g_1^2/\lambda_{\min}(R)$ is a known positive constant. Then, the tracking error system (6) is asymptotically stable.

Proof. We consider the following Lyapunov function,

$$V(k) = e^T(k)e(k) + \Gamma J_e(k) \tag{11}$$

where $J_e(k) = J_e(e(k), u_e(k))$ is defined in (7). Differencing the Lyapunov function yields

$$\Delta V(k) = e^T(k+1)e(k+1) - e^T(k)e(k) + \Gamma(J_e(k+1) - J_e(k)) \tag{12}$$

Using (6) and (7), we can obtain

$$\begin{aligned}\Delta V(k) =& (f_e(k) + g_e(k)u_e(k))^T(f_e(k) + g_e(k)u_e(k)) \\ &- e^T(k)e(k) - \Gamma(e^T(k)Qe(k) + u_e^T(k)Ru_e(k))\end{aligned} \tag{13}$$

Using the Cauchy–Schwarz inequality yields

$$\Delta V(k) \leq 2\|f_e(k)\|^2 - (\Gamma\lambda_{\min}(R) - 2g_1^2)\|u_e(k)\|^2 - \Gamma\lambda_{\min}(Q)\|e(k)\|^2 - \|e(k)\|^2 \tag{14}$$

For the purpose of asymptotically stabilizing the tracking system (6), i.e., $\Delta V(k) < 0$, it is necessary to satisfy the following

$$2\|f_e(k)\|^2 \leq \Gamma\lambda_{\min}(Q)\|e(k)\|^2 + (\Gamma\lambda_{\min}(R) - 2g_1^2)\|u_e(k)\|^2 \tag{15}$$

Thus, $\Delta V(k) < 0$ and the asymptotic stability of the tracking error system (6) are proved if the bound in (10) is satisfied. □

Remark 2. *Lemma 1 shows that under the condition that the internal dynamics $f_e(k)$ is bounded to satisfy (10), there exists an admissible control $u_e(k)$ that not only stabilizes the tracking error system (6) on Ω but also guarantees that the cost function $J_e(k)$ is finite.*

3. Optimal Tracking Controller Design with Partially Unknown Dynamics

In this section, firstly, we use an RBF-NN to approximate the unknown system dynamics $f[x(k)]$ and use another RBF-NN to approximate the steady-state controller $u_d(k)$. Secondly, two feedback neural networks are introduced to approximate the cost function and the optimal feedback control $u_e(k)$. Finally, the system stability is proved by selecting an appropriate Lyapunov function.

3.1. RBF-NN Identifier Design

In this subsection, in order to capture the unknown dynamics of the system (1), an RBF-NN-based identifier is proposed. Without losses of generality, this unknown dynamics is assumed to be a smooth function within a compact set. Using an RBF-NN, this unknown dynamics (1) is identified as

$$\hat{f}(x(k)) = \hat{w_f}(k)^{\mathrm{T}} h[x(k)] + \Delta_f(x) \tag{16}$$

where $\hat{w_f}(k)$ is the matrix of ideal output weights of the neural network and $h[x(k)]$ is the vector of radial basis functions, $\Delta_f(x)$ is the bounded approximation error, and $||\Delta_f(x)|| < \varepsilon_f$, where ε_f is a positive constant.

For any non-zero approximation error $\Delta_f(x)$, there exists optimal weight matrix ${w_f}^*$ such that

$$f(x(k)) = \hat{f}(x, w_f^*) - \Delta_f(x) \tag{17}$$

where w_f^* is the optimal weight of identifier, and $\hat{f}(x, w_f^*) = w_f^*(k)^{\mathrm{T}} h[x(k)]$. The output weights are updated, and the hidden weights remain unchanged when training, so the neural network model identification error is

$$\begin{aligned} \tilde{f}(x(k)) &= f[x(k)] - \hat{f}[x(k)] \\ &= \hat{f}(x, {w_f}^*) - \Delta_f[x(k)] - \hat{w_f}(k)^{\mathrm{T}} h[x(k)] \\ &= -\tilde{w_f}(k)^{\mathrm{T}} h[x(k)] - \Delta_f[x(k)] \end{aligned} \tag{18}$$

where$-\tilde{w_f}(k) = {w_f}^*(k) - \hat{w_f}(k)$.

The error function is defined as the following

$$E(k+1) = \frac{1}{2}[\tilde{f}(x(k))]^T[\tilde{f}(x(k))] \tag{19}$$

Using the gradient descent method, the weights are updated by

$$\begin{aligned} \Delta w_{f_j}(k+1) &= -\eta \frac{\partial E}{\partial w_{f_j}} \\ &= \eta(f(x(k)) - \hat{f}(x(k)))h[x(k)] \\ &= \eta(\tilde{f}(x(k)))h[x(k)] \end{aligned} \tag{20}$$

and

$$w_{f_j}(k) = w_{f_j}(k-1) + \Delta w_{f_j}(k) \tag{21}$$

where $\eta > 0$ is the learning rate of the identifier.

Inspired by the work in [36], we must state the following assumptions before proceeding.

Assumption 2. *The neural network identifying error is assumed to have an upper bound, namely*

$$\Delta_f(x)^{\mathrm{T}}\Delta_f(x) \leq \tilde{w}_f(k)^{\mathrm{T}}\tilde{w}_f(k)h[x(k)]^{\mathrm{T}}h[x(k)] \tag{22}$$

3.2. RBF-NN Steady-State Controller Design

We use the RBF-NN to approximate the steady-state control $u_d(k)$ directly, and the inverse dynamic NN is established to approximate [16,19].

We design the steady-state control $u_d(k)$ through the approximation of the RBF-NN

$$u_d(k) = \hat{w}_d{}^T(k)h[x_d(k)] \tag{23}$$

where $\hat{w}_d$ is the actual neural network weights; $h[x_d(k)]$ is the output of the hidden layers; and $u_d(k)$ is the output of the RBF-NN.

Let the ideal steady-state control $u_d^*(k)$ be

$$u_d^*(k) = w_d^{*T}h[x_d(k)] + \varepsilon_u \tag{24}$$

where w_d^* is the optimal neural network weights and ε_u is the error vector. Assuming that $x_d(k+1)$ is the reference output of the system at the point $k+1$, without considering external disturbances, the control input $u_d^*(k)$ satisfies

$$L[x_d(k), u_d^*(k)] - x_d(k+1) = 0 \tag{25}$$

where $L[x_d(k), u_d^*(k)] = f[x_d(k)] + g[x_d(k)]u_d^*(k)$.

Thus, we can define the error $e_m(k)$ of the approximation state as

$$e_m(k+1) = L[x_d(k), u_d(k)] - x_d(k+1) \tag{26}$$

where $L[x_d(k), u_d(k)] = f[x_d(k)] + g[x_d(k)]u_d(k)$.

(24) subtracted from (23) yields

$$\begin{aligned} u_d(k) - u_d^*(k) &= \hat{w}_d{}^T(k)h[x_d(k)] - w_d^{*T}(k)h[x_d(k)] - \varepsilon_u \\ &= \tilde{w}_d{}^T(k)h[x_d(k)] - \varepsilon_u \end{aligned} \tag{27}$$

where $\tilde{w}_d(k) = \hat{w}_d(k) - w_d^*(k)$ is weight approximation error.

The weights are updated by the following update law of the weights

$$\hat{w}_d(k+1) = \hat{w}_d(k) - \gamma[h(x_d(k))e_m(k+1) + \sigma\hat{w}_d(k)] \tag{28}$$

where $\gamma > 0$ and $\sigma > 0$ are the positive constant.

Assumption 3. *Within the set Ω_ε, the optimal neural network weights w^* and the approximation error are bounded.*

$$\| w_d^* \| \leqslant w_m, ||\varepsilon_u|| \leqslant \varepsilon_l \tag{29}$$

3.3. Near-Optimal Feedback Controller Design

In this subsection, we present an ADP algorithm based on the Bellman optimality. The goal is to find the optimal approximate feedback control law that minimizes the approximate cost function.

First, considering the HJB Equation (7) and the optimal feedback control (8), the cost function $J_e(e(k), u_e(k))$ is rewritten as $V^i(e(k))$. The initial cost function $V^0(e(k)) = 0$

may not represent the optimal value function. Then, a single control vector $u_e^0(k)$ can be solved by

$$\begin{aligned} V^0(e(k)) &= \min_{u_e(k)}\{e^T(k)Qe(k) + u_e^T(k)Ru_e(k) + V^0(e(k+1))\} \\ &= e^T(k)Qe(k) + (u_e^0(k))^T Ru_e^0(k) \end{aligned} \tag{30}$$

Updating the control law yields

$$\begin{aligned} u_e^1(k) &= \arg\min_{u_e(k)}\{e^T(k)Qe(k) + u_e^T(k)Ru_e(k) + V^0(e(k+1))\} \\ &= -\frac{1}{2}R^{-1}g_e^T(k)\frac{\partial V^0(e(k+1))}{\partial e(k+1)} \end{aligned} \tag{31}$$

Hence, for $i = 1, 2, \ldots$, the ADP algorithm can be realized in a continuous iterative process in

$$\begin{aligned} V^i(e(k)) &= \min_{u_e(k)}\Big\{e^T(k)Qe(k) + u_e^T(k)Ru_e(k) + V^i(e(k+1))\Big\} \\ &= e^T(k)Qe(k) + (u_e^i(k))^T Ru_e^i(k) + V^i(e(k+1)) \end{aligned} \tag{32}$$

and

$$\begin{aligned} u_e^{i+1}(k) &= \arg\min_{u_e(k)}\{e^T(k)Qe(k) + u_e^T(k)Ru_e(k) + V^i(e(k+1))\} \\ &= -\frac{1}{2}R^{-1}g_e^T(k)\frac{\partial V^i(e(k+1))}{\partial e(k+1)} \end{aligned} \tag{33}$$

where index i represents the number of iterations of the control law and the cost function, i.e., the update count of internal neuron to update the weight parameters, while index k represents time index of state. Moreover, it is worth noting in the iterative process of the ADP algorithm that the number of iterations of the cost function and the control law increases from zero to infinity.

To begin the development of the feedback control policy, we used neural networks to construct the critic network and the actor network.

The cost function $V_i(e(k))$ is defined as the critic network.

The output of the critic network is denoted as

$$\widehat{V}^i(e(k)) = w_{ci}^T z(\nu_{ci}^T e(k)) + \varepsilon_c(k) \tag{34}$$

where $z(\nu_{ci}^T e(k))$ is the hidden layer function, w_{ci} is the hidden layer weight of the critic network, ν_{ci} is the input layer weight of the critic network, and $\varepsilon_c(k)$ is the approximation error.

So, we define the prediction error of the critic network as

$$e_{ci}(k) = \widehat{V}^i(e(k)) - V^i(e(k)) \tag{35}$$

The error function of the critic network is defined as

$$E_{ci}(k) = \frac{1}{2}e_{ci}^T(k)e_{ci}(k). \tag{36}$$

Using the gradient descent method, the weights of the critic network are updated,

$$w_{ci}(k+1) = w_{ci}(k) - \alpha_c[\frac{\partial E_{ci}(k)}{\partial w_{ci}(k)}] \tag{37}$$

where $\alpha_c > 0$ is the learning rate of the critic network.

The inputs of the actor network is the system error $e(k)$, and the outputs of the actor network is the optimal feedback control $u_e(k)$. The output can be formulated as

$$\hat{u}_e^i(k) = w_{ai}^T z(v_{ai}^T e(k)) + \varepsilon_a(k), \tag{38}$$

where $z(v_{ai}^T e(k))$ is the hidden layer function, w_{ai} is the hidden layer weight of the actor network, v_{ai} is the input layer weight of the actor network, and $\varepsilon_a(k)$ is the approximation error.

Similar to the critic network, the prediction error of the actor network is defined as

$$e_{ai}(k) = \hat{u}_e^i(k) - u_e^i(k) \tag{39}$$

where $\hat{u}_e^i(k)$ is approximation optimal feedback control, and $u_e^i(k)$ is the optimal feedback control at the iterative number i.

The error function of the actor network is defined as

$$E_{ai}(k) = \frac{1}{2} e_{ai}^T(k) e_{ai}(k) \tag{40}$$

The weights of the actor network are also updated in the same way as the critic network; we use the gradient descent method

$$w_{ai}(k+1) = w_{ai}(k) - \beta_a [\frac{\partial E_{ai}(k)}{\partial w_{ai}(k)}], \tag{41}$$

where $\beta_a > 0$ is the learning rate of the actor network, and i is the update count of the internal neuron to update the weight parameters.

3.4. Stability Analysis

In this subsection, we give the stability proof by Lyapunov's stability theory.

Assumption 4. *Radial basis function* $h(t) = \exp\left(-\frac{\|x(t)-c(t)\|^2}{2b^2}\right)$ *of the maximum value is* $h_{max} = 1$, *where* $c(t)$ *is the center point and* b *is the width of radial basis function. Assuming the numbers of neurons is* $l \in [l_f, l_d]$ *for any radial basis function* $h \in [h[x(k)], h[x_d(k)]]$, *then*

$$\begin{aligned} &|h_i| \leqslant 1, \|h\| \leqslant \sqrt{l} \leqslant l, \\ &h^T h = \|h\|^2 \leqslant l \end{aligned} \tag{42}$$

We can know the maximum value $\|h\|^2$ *of the hidden layer with* l *neurons is* $l \in [l_f, l_d]$, *then we assume the maximum value* $\|h[x(k)]\|^2$ *of the hidden layer for the identifier* $\hat{f}(x(k))$ *is* l_f, *and the maximum value* $\|h_d[x(k)]\|^2$ *of the hidden layer for the steady-state controller* $u_d(k)$ *is* l_d.

Lemma 2. *The relationship between (25) and weight approximation error (27) satisfies the following equation.*

$$\tilde{w_d}^T(k) h[x_d(k)] = \frac{e_m(k+1)}{L_u} + \varepsilon_u \tag{43}$$

where $e_m(k)$ *is the error of the approximation state* $x_d(k)$, $L_u = \left.\frac{\partial L}{\partial u}\right|_{u=\xi}, \xi \in [u_d^*(k), u_d(k)]$, $g_1 \geqslant \left|\frac{\partial L}{\partial u}\right| > \epsilon > 0$, g_1 *and* ϵ *are positive constants.*

Proof. Subtracting w_d^* from both sides of (28), we obtain

$$\tilde{w_d}(k+1) = \tilde{w_d}(k) - \gamma [h[x_d(k)] e_m(k+1) + \sigma \hat{w_d}(k)] \tag{44}$$

Combining (25) and (27) with the mean value theorem, we can obtain

$$\begin{aligned} L[x_d(k), u_d(k)] &= L[x_d(k)], u_d^*(k) + \tilde{w_d}^T(k)h[x_d(k)] - \varepsilon_u \Big] \\ &= L[x_d(k), u_d^*(k)] + \Big[\tilde{w_d}^T(k)h[x_d(k)] - \varepsilon_u\Big] L_u \\ &= x_d(k+1) + \Big[\tilde{w_d}^T(k)h[x_d(k)] - \varepsilon_u\Big] L_u \end{aligned} \tag{45}$$

Further combining (45) with (26), we can obtain

$$\begin{aligned} e_m(k+1) &= L[x_d(k), u_d(k)] - x_d(k+1) \\ &= [\tilde{w_d}^T(k)h[x_d(k)] - \varepsilon_u] L_u \end{aligned} \tag{46}$$

After rearranging, we can obtain

$$\tilde{w_d}^T(k)h[x_d(k)] = \frac{e_m(k+1)}{L_u} + \varepsilon_u \tag{47}$$

The proof is completed. □

Lemma 3. *For simplicity of analysis, ε_u and $e_m(k+1)$ have an inequality relation though using Assumption 3 and Young's inequality.*

$$-2\varepsilon_u^T e_m(k+1) \leqslant k_0 \|\varepsilon_u\|^2 + \frac{1}{k_0}\|e_m(k+1)\|^2 \tag{48}$$

where k_0 is a positive constant.

Theorem 1. *For the optimal tracking problem (1)–(3), the RBF-NN identifier (16) is used to approximate $f(x(k))$, the steady-state controller $u_d(k)$ is approximated by the RBF-NN (23), and the feedforward networks (34), (38) are used to approximate the cost function $J(e(k), u(k))$ and the feedback controller $u_e(k)$, respectively. Assume that the parameters satisfy the following inequality*

$$\begin{aligned} &(a)\ \ 0 < \eta \leqslant \frac{1}{l_f} \\ &(b)\ \ 0 < g_1 \leqslant k_0 \\ &(c)\ \ 0 < (1+\sigma) l_d \gamma \leqslant \frac{1}{g_1} - \frac{1}{k_0} \\ &(d)\ \ 0 < (l_d + \sigma)\gamma \leqslant 1 \\ &(e)\ \ a_c \leq 2/\|z(v_{ci}^T e(k))\|^2 \\ &(f)\ \ \beta_a \leq 2/\|z(v_{ai}^T e(k))\|^2 \end{aligned} \tag{49}$$

where η is the learning rate of the RBF-NN identifier, σ and γ are the update parameters of the steady-state controller approximation network weights, a_c is the learning rate of the actor network, β_a is the learning rate of the critic network, and $z(v_{ci}^T e(k))$ and $z(v_{ai}^T e(k))$ are hidden layer functions of the actor network and the critic network. Then, the closed loop system (6) of approximation error is asymptotically stable when the parameter estimation errors are bounded.

Proof. Considering the following positive definite Lyapunov function candidate

$$\begin{aligned} J(k) &= J_1(k) + J_2(k) + J_3(k) + J_4(k) \\ &= \frac{1}{\eta}\tilde{w_f}(k)^T\tilde{w_f}(k) + \frac{1}{g_1}e_m(k)^T e_m(k) + \frac{1}{\gamma}\tilde{w_d}(k)^T\tilde{w_d}(k) + w_{ci}(k)^T w_{ci}(k) + w_{ai}(k)^T w_{ai}(k) \end{aligned} \tag{50}$$

where $J_1(k) = \frac{1}{\eta}\tilde{w}_f(k)^T\tilde{w}_f(k)$, $J_2(k) = \frac{1}{g_1}e_m(k)^Te_m(k) + \frac{1}{\gamma}\tilde{w}_d(k)^T\tilde{w}_d(k)$, $J_3(k) = w_{ci}(k)^Tw_{ci}(k)$, $J_4(k) = w_{ai}(k)^Tw_{ai}(k)$.

Firstly, differencing it according to the Lyapunov function of $J_1(k) = \frac{1}{\eta}\tilde{w}_f(k)^T\tilde{w}_f(k)$ yields

$$\begin{aligned}
\Delta J_1(k) &= J_1(k+1) - J_1(k) \\
&= \frac{1}{\eta}\tilde{w}_f(k+1)^T\tilde{w}_f(k+1) - \frac{1}{\eta}\tilde{w}_f(k)^T\tilde{w}_f(k) \\
&= \frac{1}{\eta}[\tilde{w}_f(k) + \eta\tilde{f}(x(k))h[x(k)]]^T[\tilde{w}_f(k) + \eta\tilde{f}(x(k))h[x(k)]] - \frac{1}{\eta}\tilde{w}_f(k)^T\tilde{w}_f(k) \\
&= \frac{1}{\eta}[\tilde{w}_f(k)^T\tilde{w}_f(k) - \frac{1}{\eta}\tilde{w}_f(k)^T\tilde{w}_f(k) + \eta^2[\tilde{f}(x(k))^T\tilde{f}(x(k))h[x(k)]^Th[x(k)] \\
&\quad + 2\eta\tilde{f}(x(k))\tilde{w}_f(k)^Th[x(k)] \\
&= \eta[[\tilde{w}_f(k)^Th[x(k)] + \Delta_f[x]]^T[\tilde{w}_f(k)^Th[x(k)] + \Delta_f[x]]h[x(k)]^Th[x(k)]] \\
&\quad + 2[\tilde{w}_f(k)^Th[x(k)] + \Delta_f[x]]\tilde{w}_f(k)^Th[x(k)] \\
&= \eta[\tilde{w}_f(k)^T\tilde{w}_f(k)h[x(k)]^Th[x(k)] + 2\tilde{w}_f(k)^Th[x(k)]\Delta_f[x] - 2\tilde{w}_f(k)^Th[x(k)] \\
&\quad - 2\tilde{w}_f(k)^T\tilde{w}_f(k)h[x(k)]^Th[x(k)] + \Delta_f[x]^T\Delta_f[x]h[x(k)]^Th[x(k)]
\end{aligned} \tag{51}$$

According to the Assumption 2, Assumption 4 and (42), (51) can be carried out to obtain

$$\begin{aligned}
\Delta J_1(k) &\leqslant \eta l_f^2 \parallel \tilde{w}_f(k) \parallel^2 - 2l_f \parallel \tilde{w}_f(k) \parallel^2 + \eta l_f^2 \parallel \tilde{w}_f(k) \parallel^2 \\
&\quad + 2\eta l_f\tilde{w}_f(k)^Th[x(k)]\Delta_f[x] - 2\tilde{w}_f(k)^Th[x(k)]\Delta_f[x] \\
&\leqslant \parallel \tilde{w}_f(k) \parallel^2 (2\eta l_f^2 - 2l_f) + (2l_f\eta - 2)\tilde{w}_f(k)^Th[x(k)]\Delta_f[x] \\
&\leqslant \parallel \tilde{w}_f(k) \parallel^2 (4l_f^2\eta - 4l_f)
\end{aligned} \tag{52}$$

Next, differencing according to the Lyapunov function of $J_2(k) = \frac{1}{g_1}e_m(k)^Te_m(k) + \frac{1}{\gamma}\tilde{w}_d(k)^T\tilde{w}_d(k)$ yields

$$\begin{aligned}
\Delta J_2(k) &= J_2(k+1) - J_2(k) \\
&= \frac{1}{g_1}[e_m(k+1)^Te_m(k+1) - e_m(k)^Te_m(k)] - \frac{1}{\gamma}\tilde{w}_d(k)^T\tilde{w}_d(k) + \frac{1}{\gamma}\tilde{w}_d(k+1)^T\tilde{w}_d(k+1) \\
&= \frac{1}{\gamma}\langle\tilde{w}_d(k) - \gamma[h[x_d(k)]e_m(k+1) + \sigma\hat{w}_d(k)]\rangle^T\langle\tilde{w}_d(k) - \gamma[h[x_d(k)]e_m(k+1) + \sigma\hat{w}_d(k)]\rangle \\
&\quad - \frac{1}{\gamma}\tilde{w}_d(k)^T\tilde{w}_d(k) + \frac{1}{g_1}[e_m(k+1)^Te_m(k+1) - e_m(k)^Te_m(k)] \\
&= \frac{1}{g_1}[e_m(k+1)^Te_m(k+1) - e_m(k)^Te_m(k)] - 2\tilde{w}_d(k)^Th[x_d(k)]e_m(k+1) \\
&\quad - 2\sigma\tilde{w}_d(k)^T\hat{w}_d(k) + \gamma h^T[x_d(k)]h[x_d(k)]e_m(k+1)^Te_m(k+1) + \gamma\sigma^2\hat{w}_d(k)^T\hat{w}_d(k) \\
&\quad + 2\gamma\sigma\hat{w}_d(k)^Th[x_d(k)]e_m(k+1)
\end{aligned} \tag{53}$$

where

$$\begin{aligned} 2\sigma\tilde{w}_d(k)^T\hat{w}_d(k) &= \sigma\tilde{w}_d(k)^T[\bar{w}_d(k)+w_d^*]+\sigma[\hat{w}_d(k)-\omega_d^*]^T\hat{w}_d(k) \\ &= \sigma \parallel \tilde{w}_d(k) \parallel^2 + \parallel \hat{w}_d(k) \parallel^2 + \tilde{w}_d(k)^T w_d^* - w_d^*\hat{w}_d(k)^T \\ &= \sigma[\parallel \tilde{w}_d(k) \parallel^2 + \parallel \hat{w}_d(k) \parallel^2 - \parallel w_d^* \parallel^2], \end{aligned}$$

$$\gamma h^T[x_d(k)]h[x_d(k)]e_m(k+1)^T e_m(k+1) \leqslant \gamma l_d \|e_m(k+1)\|^2, \tag{54}$$

$$2\gamma\sigma\hat{w}_d{}^T(k)h[x_d(k)]e_m(k+1) \leqslant \gamma\sigma l_d[\parallel \hat{w}_d(k) \parallel^2 + \|e_m(k+1)\|^2],$$

$$\gamma\sigma^2\hat{w}_d{}^T(k)\hat{w}_d(k) = \gamma\sigma^2\|\hat{w}_d(k)\|^2$$

Considering (26) and $g_1 \geqslant \left|\frac{\partial L}{\partial u}\right| > \epsilon > 0$, we can deduce

$$\frac{1}{g_1} - \frac{2}{L_u} \leqslant \frac{1}{g_1} - \frac{2}{g_1} = -\frac{1}{g_1} < 0 \tag{55}$$

Recall Lemmas 2 and 3; substituting (54) into (53) yields

$$\begin{aligned} \Delta J_2(k) &\leqslant \left[-\frac{1}{g_1} + \gamma(1+\sigma)l_d + \frac{1}{k_0}\right]\|e_m(k+1)\|^2 + \sigma(\gamma l_d + \gamma\sigma - 1) \parallel \hat{w}_d(k) \parallel^2 \\ &\quad - \frac{1}{g_1}\|e_m(k)\|^2 - \sigma \parallel \bar{w}_d(k) \parallel^2 + \sigma\omega_m^2 + k_0\varepsilon_l^2 \\ &= -\left[\frac{1}{g_1} - (1+\sigma)l_d\gamma - \frac{1}{k_0}\right]\|e_m(k+1)\|^2 + \sigma[(l_d+\sigma)\gamma - 1] \parallel \hat{w}_d(k) \parallel^2 \\ &\quad - \frac{1}{g_1}\Big[\|e_m(k)\|^2 - \beta\Big] - \sigma \parallel \tilde{w}_d(k) \parallel^2 \end{aligned} \tag{56}$$

where $\beta = g_1(\sigma w_m^2 + k_0\varepsilon_l^2)$ is a positive constant.

Next, we consider the following Lyapunov function

$$J_3(k) + J_4(k) = w_{ci}(k)^T w_{ci}(k) + w_{ai}(k)^T w_{ai}(k). \tag{57}$$

Then, differencing it according to the Lyapunov function of (57) yields

$$\begin{aligned} \Delta J_3(k) + \Delta J_4(k) &= \{w_{ci}(k+1)^T w_{ci}(k+1) + w_{ai}(k+1)^T w_{ai}(k+1)\} \\ &\quad - \{w_{ci}(k)^T w_{ci}(k) + w_{ai}(k)^T w_{ai}(k)\} \\ &= a_c\|e_{ci}(k)\|^2\big(-2 + a_c\|z(v_{ci}^T e(k))\|^2\big) \\ &\quad + \beta_a \parallel e_{ai}(k) \parallel^2 (-2 + \beta_a \parallel z(v_{ai}^T e(k)) \parallel^2). \end{aligned} \tag{58}$$

Finally, $\Delta J(k)$ is derived from (52), (56), and (58)

$$\begin{aligned} \Delta J(k) &= \Delta J_1(k) + \Delta J_2(k) + \Delta J_3(k) + \Delta J_4(k) \\ &\leqslant 4 \parallel \tilde{w}_f(k) \parallel^2 (l_f^2\eta - l_f) - \sigma \parallel \tilde{w}_d(k) \parallel^2 - \left[\frac{1}{g_1} - (1+\sigma)l_d\gamma - \frac{1}{k_0}\right]\|e_m(k+1)\|^2 \\ &\quad + \sigma[(l_d+\sigma)\gamma - 1] \parallel \hat{w}_d(k) \parallel^2 - \frac{1}{g_1}\Big[\|e_m(k)\|^2 - \beta\Big] + a_c\|e_{ci}(k)\|^2\big(-2 + a_c\|z(v_{ci}^T e(k))\|^2\big) \\ &\quad + \beta_a \parallel e_{ai}(k) \parallel^2 (-2 + \beta_a \parallel z(v_{ai}^T e(k)) \parallel^2). \end{aligned} \tag{59}$$

Based on the above analysis, when the parameters are selected to fulfill the following condition with $\|e_m(k)\|^2 \geqslant \beta$,

$$\begin{aligned} &0 < \eta \leqslant \frac{1}{l_f} \\ &0 < g_1 \leqslant k_0 \\ &0 < (1+\sigma) l_d \gamma \leqslant \frac{1}{g_1} - \frac{1}{k_0} \\ &0 < (l_d + \sigma)\gamma \leqslant 1 \\ &a_c \leq 2/\|z(v_{ci}^T e(k))\|^2 \\ &\beta_a \leq 2/\|z(v_{ai}^T e(k))\|^2 \end{aligned} \tag{60}$$

we can obtain $\Delta J(k) \leqslant 0$. □

The working process of the proposed control technique is shown in Figure 1. As shown in Figure 1, with $x(k)$, $u_d(k)$ and $u_e^i(k)$, the estimated error $e(k+1)$ can be obtained by using the RBF-NN identifier and the steady-state controller. Corresponding to the steady-state controller $u_d(k)$, we can obtain the reference trajectory $x_d(k)$. Using the ADP algorithm, we can obtain nearly optimal feedback controller $\hat{u}_e^i(k)$. Then, the actual controller $u(k) = \hat{u}_e^i(k) + u_d(k)$ and system dynamics $x(k+1)$ can be obtained. In addition, by using $x_d(k)$ and $x(k)$ the estimated tracking error $e(k)$ can be obtained, $e(k+1)$ can be further obtained. Finally, we can reconstruct the system dynamics to track the reference trajectory.

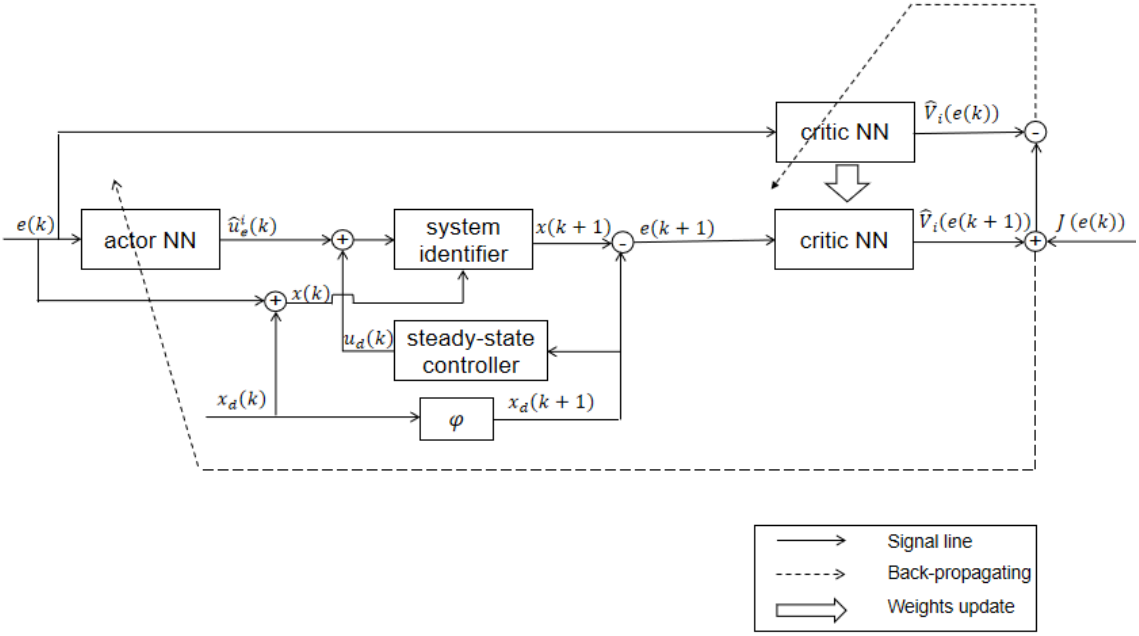

Figure 1. The structure schematic of the proposed technique.

4. Simulation

In this section, we give the simulation results of our method and compare it with other methods [36]. A discrete-time nonlinear system is introduced to demonstrate the effectiveness of the proposed tracking control method. The case is derived from [24]. We assume that the nonlinear smooth function $f \in \mathbb{R}^n$ is an unknown nonlinear drift function and $g \in \mathbb{R}^{n \times m}$ is a known function. The corresponding $f[x(k)]$ and $g[x(k)]$ are given as

$$f[x(k)] = \begin{bmatrix} f_1[x(k)] \\ f_2[x(k)] \end{bmatrix} = \begin{bmatrix} -\sin(0.5x_2(k))x_1^2(k) \\ -\cos(1.4x_2(k))\sin(0.9x_1(k)) \end{bmatrix}$$
$$g[x(k)] = \begin{bmatrix} (x_1(k))^2 + 1.5 & 0.1 \\ 0 & 0.2((x_1(k) + x_2(k))^2 + 1) \end{bmatrix} \tag{61}$$

The reference trajectory $x_d(k)$ for the above system is defined as

$$x_d(k) = \begin{bmatrix} 0.25\sin(10^{-3}k) \\ 0.25\cos(10^{-3}k) \end{bmatrix} \tag{62}$$

where $u(k) \in \mathbb{R}^2 \in [u_1(k), u_2(k)]^T$, and $time(s)$ of y-axis is chosen to have a $k(1, \ldots, 10,000)$ multiplied by $ts = 0.001$ in the simulation.

4.1. Simulation Result of the Proposed Method

In this subsection, we give the simulation result for our proposed method.

Firstly, in order to deal with the unknown dynamics, we need to use two RBF networks to obtain the RBF identifier and the RBF steady-state controller. The RBF networks have a three-layer structure with two input neurons, hidden layers have nine neurons, and output layer have two neurons. The parameters c_i and b_j of the radial basis functions are chosen to be $c_i = \begin{bmatrix} -2 & -1.5 & -1.0 & -0.5 & 0 & 0.5 & 1.0 & 1.5 & 2 \\ -2 & -1.5 & -1.0 & -0.5 & 0 & 0.5 & 1.0 & 1.5 & 2 \end{bmatrix}$ and $b_j = [b_1, b_2] = [2, 2]$, and the initial weights w_0 are chosen to be random numbers between $(0, 1)$. For the RBF identifier with its weights updating law (21) to update the weights $\hat{w_f}$, the unknown function f is identified by the input/output data $x(k)$. For the RBF steady-state controller, the reference trajectory data x_d and the weights updating law (28) are used to update the weights $\hat{w_d}$ to identify the steady-state controller u_d. Because $g_1 \geqslant \frac{\partial L}{\partial u} = 1$, we can select $g_1 = 5$. According to $0 < g_1 \leqslant k_0$ of Theorem 1, we can select $k_0 = 10$. For the control parameters η, because hidden layers have nine neurons, $l = 9$, $0 < \eta \leqslant \frac{1}{l} \leqslant \frac{1}{9}$, we select $\eta = 0.1$. With control parameters γ, σ, we can know $0 < (1 + \sigma)9\gamma \leqslant \frac{1}{5} - \frac{1}{10} = \frac{1}{10} = 0.10$ and $0 < (9 + \sigma)\gamma \leqslant 1$ from Theorem 1 and thus select $\gamma = 0.01, \sigma = 0.001$. The initial state is set as $x(0) = [0, 0]^T$. We trained the RBF networks with 10,000 steps of acquired data. Figures 2 and 3 show the RBF-NN identifiers to approximate the tracking curves of the unknown dynamics $\tilde{f} \in [\tilde{f}_1[x(k)], \tilde{f}_2[x(k)]]^T$.

Then, based on the ADP algorithm of Bellman optimality, Equation (6) was used to obtain the tracking error e and the optimal feedback control u_e to train the critic network and the actor network, respectively. Meanwhile, the obtained standard control inputs $u = \hat{u}_e^i + u_d$ were used in system (1), which keeps on looping until the value function $V^i(e(k))$ converge and the tracking error $e(k)$ is zero, where the performance index is selected as $Q = I$ and $R = I$, where I is the identity matrix with appropriate dimension. For the actor network and the critic network, we used the same parameter settings. The initial weights of the critic networks and actor networks are randomly chosen between $(-10, 10)$. The input layer has 2 neurons, the hidden layer has 15 neurons, the output layer has 2 neurons, and the learning rate is 0.1. The hidden layer uses the function *tansig* and the function *purelin*, and the output layer uses the function *trainlm*. Though parameter settings, we trained the actor network and the critic network with 5000 training steps to reach the given accuracy 1×10^{-9}. Figure 4 shows the curves of the system control $u \in [u_1, u_2]$. In Figures 5 and 6, we can see the curves of the state trajectory x and the reference trajectory x_d.

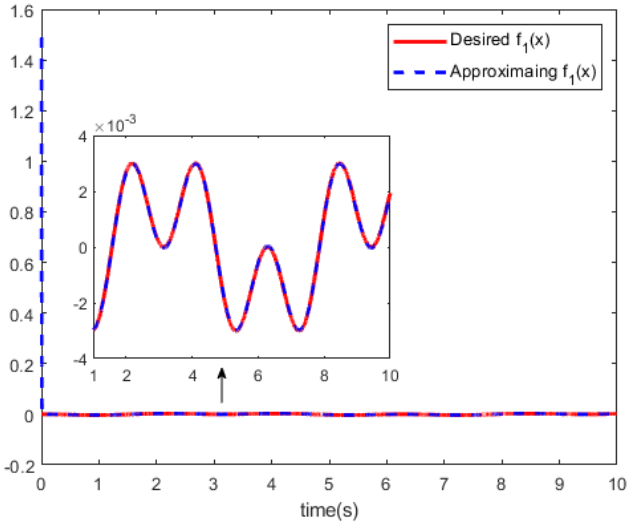

Figure 2. The unknown function $f_1(x)$ and approximation of the unknown function $\tilde{f}_1(x)$.

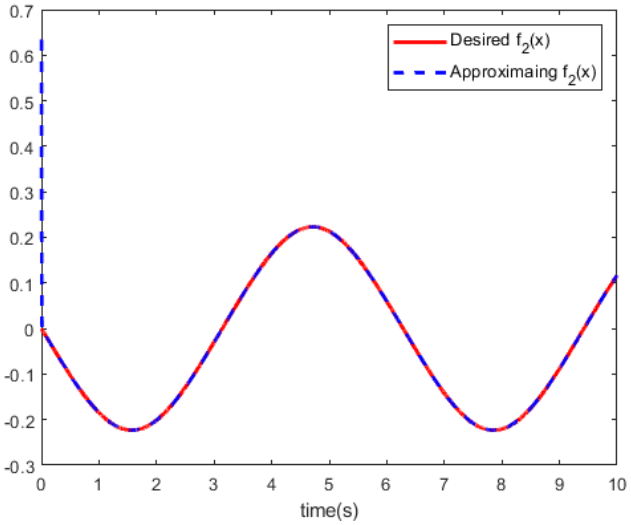

Figure 3. The unknown function $f_2(x)$ and approximation of the unknown function $\tilde{f}_2(x)$.

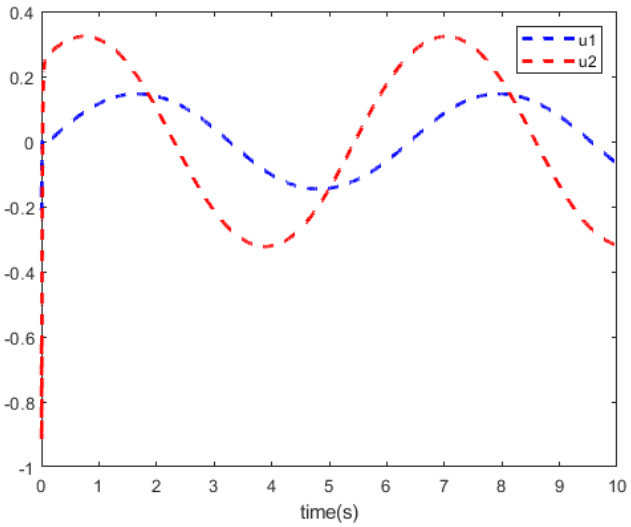

Figure 4. The system control input u_1 and the system control input u_2.

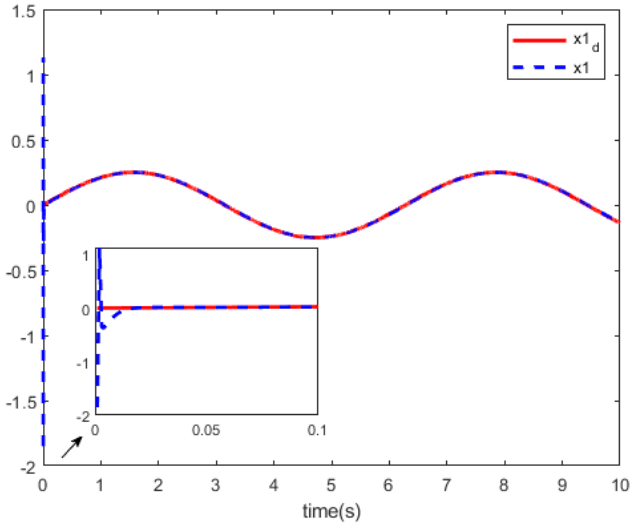

Figure 5. The state trajectory x_1 and the reference trajectory x_{1d} using our tracking control method.

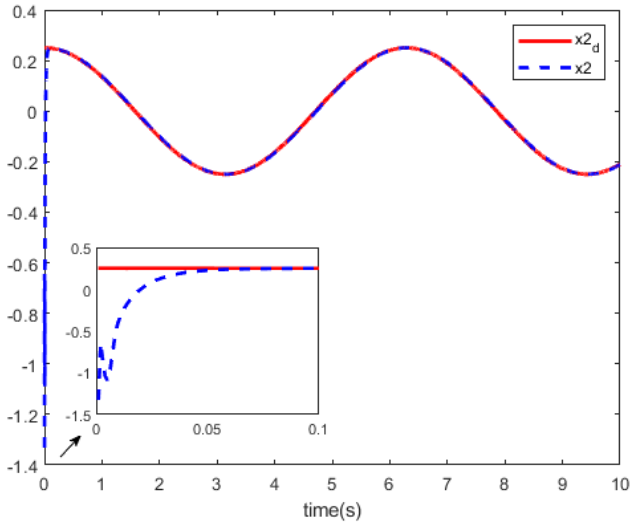

Figure 6. The state trajectory x_2 and the reference trajectory x_{2d} using our tracking control method.

Based on above the results, the simulation results show that this tracking technique obtains a relatively satisfactory tracking performance for partially unknown discrete-time nonlinear systems.

4.2. Comparison with Other Methods

In this subsection, we will compare with the research results in [36], which use a BP neural network to approximate the unknown system dynamics. In the comparison, we use the same system dynamics and desired tracking trajectory as (61) and (62) with the initial state $x(0) = [0, 0]^T$ and the performance index $R = Q = I$.

To begin with, an NN identifier is established by a three-layer BP neural network, which is chosen to have a 4–10–2 structure with four input neurons, eight hidden neurons, and two output neurons. The feedforward-neuro-controller is also established by a three-layer BP NN, which is chosen to have a 2–10–2 structure with two input neurons, eight hidden neurons, and two output neurons. For the NN identifier and the feedforward-neuro-controller, the parameter settings of the neural networks are identical, where the hidden layers use the sigmoidal function *tansig*, the output layers use the linear function *purelin*, the learning rate is 0.1, and the initial weights are chosen to be random numbers between $(0, 1)$.

For the actor network and the critic network, we also use the same parameter settings. A 2–15–2 structure is chosen for the critic networks and actor networks, the initial weights are randomly chosen between $(-10, 10)$, and the learning rate is 0.1. The hidden layer uses the function *tansig* and the function *purelin*, and the output layer uses the function *trainlm*. Then, the given accuracy is 1×10^{-9}. In Figures 7 and 8, we can see the curves of the state trajectory x and the reference trajectory x_d using tracking control methods for references.

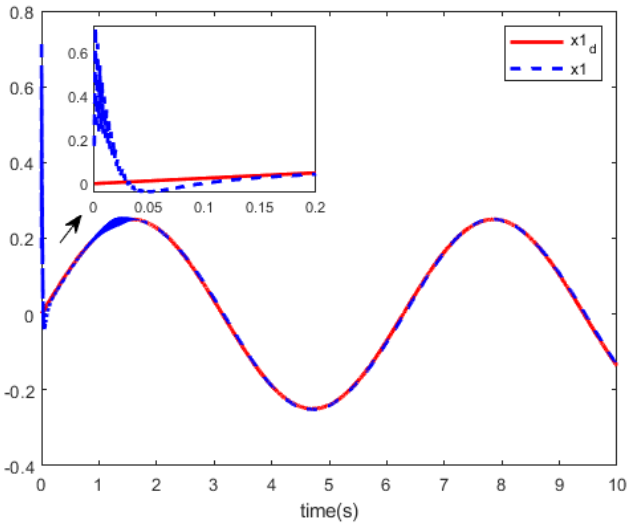

Figure 7. The state trajectory x_1 and the reference trajectory x_{1d} using tracking control methods for references.

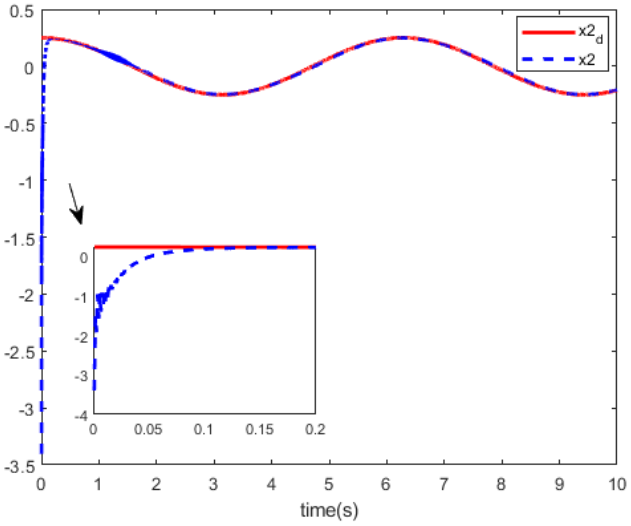

Figure 8. The state trajectory x_2 and the reference trajectory x_{2d} using tracking control methods for references.

Comparing the two methods, from Figures 5–8, we can see that our method has better performance in tracking the reference trajectory.

5. Conclusions

This paper proposes an effective scheme to find the near-optimal tracking controller for a class of partially unknown discrete-time nonlinear systems based on RBF-NNs. In dealing with unknown variables, two RBF-NNs are used to approximate the unknown function and the steady-state controller. Moreover, the ADP algorithm is introduced to obtain the optimal feedback control for tracking the error dynamics, two feedforward neural networks are utilized as structures to approximate the cost function and the feedback controller. Finally,

simulation results show a relatively satisfactory tracking performance, which verifies the effectiveness of the optimal tracking control technique. In future work, we may consider completely unknown dynamics and event-triggering conditions.

Author Contributions: Methodology, D.X.; software, Y.L.; investigation, J.H.; resources, Y.M. All the authors contributed equally to the development of the research. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant No. 61463002, the Guizhou Province Natural Science Foundation of China under Grant No. Qiankehe Fundamentals-ZK[2021] General 322, and the Doctoral Foundation of Guangxi University of Science and Technology Grant No. Xiaokebo 22z04.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhai, D.; Lu, A.-Y.; Dong, J.; Zhang, Q. Adaptive Tracking Control for a Class of Switched Nonlinear Systems under Asynchronous Switching. *IEEE Trans. Fuzzy Syst.* **2018**, *6*, 1245–1256. [CrossRef]
2. Zhang, Y.; Wang, X.; Wang, Z. Discrete-Time Adaptive Fuzzy Finite-Time Tracking Control for Uncertain Nonlinear Systems. *IEEE Trans. Fuzzy Syst.* **2024**, *2*, 649–659. [CrossRef]
3. Zhao, D.; Wang, Z.; Ho, D.W.C.; Wei, G. Observer-Based PID Security Control for Discrete Time-Delay Systems under Cyber-Attacks. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *6*, 3926–3938. [CrossRef]
4. Lewis, F.L.; Vrabie, D.; Syrmos, V.L. *Optimal Control*, 3rd ed.; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2012.
5. Mannava, A.; Balakrishnan, S.N.; Tang, L.; Landers, R.G. Optimal tracking control of motion systems. *IEEE Trans. Control Syst. Technol.* **2012**, *20*, 1548–1558. [CrossRef]
6. Sharma, R.; Tewari, A. Optimal nonlinear tracking of spacecraft attitude maneuvers. *IEEE Trans. Control Syst. Technol.* **2013**, *12*, 677–682. [CrossRef]
7. Bellman, R.E. *Dynamic Programming*; Princeton University Press: Princeton, NJ, USA, 1957.
8. Lewis, F.L.; Syrmos, V.L. *Optimal Control*; Wiley: New York, NY, USA, 1995.
9. Narendra, K.S.; Parthasarathy, K. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Netw.* **1990**, *1*, 4–27. [CrossRef] [PubMed]
10. Narendra, K.S.; Mukhopadhyay, S. Adaptive control of nonlinear multivariable systems using neural networks. In Proceedings of the 32nd IEEE Conference on Decision and Control, San Antonio, TX, USA, 15–17 December 1993; pp. 737–752.
11. Poggio, T.; Girosi, F. Networks for approximation and learning. *Proc. IEEE* **1990**, *78*, 1481–1497. [CrossRef]
12. Hartman, E.J.; Keeler, J.D.; Kowalski, J.M. Layered Neural Networks with Gaussian Hidden Units as Universal Approximations.*Neural Comput.* **1990**, *2*, 210–215. [CrossRef]
13. Park, J. Universal approximation using radial basis function networks. *Neural Comput.* **1993**, *3*, 246–257. [CrossRef]
14. Powell, M.J.D. Radial Basis Functions for Multivariable Interpolation: A Review. In *Algorithms for Approximation*; Mason, J.C., Cox, M.G., Eds.; Clarendon Press: Oxford, UK, 1987; pp. 143–167.
15. Nelles, O.; Isermann, R. A Comparison between RBF Networks and Classical Methods for Identification of Nonlinear Dynamic Systems. In *Adaptive Systems in Control and Signal Processing*; Pergamon: Oxford, UK, 1995.
16. Ge, S.S.; Zhang, J.; Lee, T.H. Adaptive MNN control for a class of non-affine NARMAX systems with disturbances. *Syst. Control Lett.* **2004**, *53*, 1–12. [CrossRef]
17. Kobayashi, H.; Ozawa, R. Adaptive neural network control of tendon-driven mechanisms with elastic tendons. *Automatica* **2003**, 1509–1519. [CrossRef]
18. Zhang, H.; Luo, Y.; Liu, D. Neural-Network-Based Near-Optimal Control for a Class of Discrete-Time Affine Nonlinear Systems with Control Constraints. *IEEE Trans. Neural Netw.* **2009**, 20, 1490–1503. [CrossRef] [PubMed]
19. Liu, J.K. *Radial Basis Function (RBF) Neural Network Control for Mechanical Systems: Design, Analysis and Matlab Simulation*; Springer: Berlin/Heidelberg, Germany, 2013.
20. Powell, W. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*; Wiley Series in Probability and Statistics; Wiley: Hoboken, NJ, USA, 2007.
21. Vrabie, D.; Lewis, F. Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Netw.* **2009**, *4*, 237–246. [CrossRef] [PubMed]
22. Liu, D.; Yang, X.; Li, H. Adaptive optimal control for a class of continuous-time affifine nonlinear systems with unknown internal dynamics. *Neural Comput. Appl.* **2013**, *11*, 1843–1850. [CrossRef]
23. Bhasin, S.; Kamalapurkar, R.; Johnson, M.; Vamvoudakis, K.; Lewis, F.L.; Dixon, W.E. A novel actor–critic–identififier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica* **2013**, *1*, 82–92. [CrossRef]

24. Dierks, T.; Jagannathan, S. Optimal tracking control of affine nonlinear discrete-time systems with unknown internal dynamics. In Proceedings of the 48h IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference, Shanghai, China, 15–18 December 2009.
25. Al-Tamimi, A.; Lewis, F.L.; Abu-Khalaf, M. Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof. *IEEE Trans. Syst. Man Cybern. B Cybern.* **2008**, *8*, 943–949. [CrossRef] [PubMed]
26. Prokhorov, D.V.; Wunsch, D.C. Adaptive critic designs. *IEEE Trans. Neural Netw.* **1997**, *9*, 997–1007. [CrossRef] [PubMed]
27. Luo, Y.; Zhang, H. Approximate optimal control for a class of nonlinear discrete-time systems with saturating actuators. *Prog. Natural Sci.* **2008**, 1023–1029. [CrossRef]
28. Dierks, T.; Jagannathan, S. Online optimal control of nonlinear discrete-time systems using approximate dynamic programming. *Control Theory Appl.* **2011**, 361–369. [CrossRef]
29. Si, J.; Wang, Y.-T. Online learning control by association and reinforcement. *IEEE Trans. Neural Netw.* **2001**, *5*, 264–276. [CrossRef] [PubMed]
30. Liu, D.; Wei, Q. Policy Iteration Adaptive Dynamic Programming Algorithm for Discrete-Time Nonlinear Systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, 621–634. [CrossRef] [PubMed]
31. Kiumarsi, B.; Lewis, F.L. Actor-critic-based optimal tracking for partially unknown nonlinear discrete-time systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, 140–151. [CrossRef] [PubMed]
32. Ren, L.; Zhang, G.; Mu, C. Data-based H_∞ control for the constrained-input nonlinear systems and its applications in chaotic circuit systems. *IEEE Trans. Circuits Syst.* **2020**, *8*, 2791–2802. [CrossRef]
33. Zhao, F.; Gao, W.; Liu, T.; Jiang, Z.-P. Event-triggered robust adaptive dynamic programming with output-feedback for large-scale systems. *IEEE Trans. Control Netw. Syst.* **2023**, *8*, 63–74. [CrossRef]
34. Zhang, H.; Cui, L.; Zhang, X.; Luo, Y. Data-Driven Robust Approximate Optimal Tracking Control for Unknown General Nonlinear Systems Using Adaptive Dynamic Programming Method, in IEEE Transactions on Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2011**, *11*, 2226–2236. [CrossRef]
35. Lin, Q.; Wei, Q.; Liu, D. A novel optimal tracking control scheme for a class of discrete-time nonlinear systems using generalised policy iteration adaptive dynamic programming algorithm. *Int. J. Syst. Sci.* **2017**, *48*, 525–534. [CrossRef]
36. Huang, Y.; Liu, D. Neural-network-based optimal tracking control scheme for a class of unknown discrete-time nonlinear systems using iterative ADP algorithm. *Neurocomputing* **2014**, *125*, 46–56. [CrossRef]
37. Zhang, Y.; Zhao, B.; Liu, D.; Zhang, S. Event-triggered control of discrete-time zero-sum games via deterministic policy gradient adaptive dynamic programming. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *8*, 4823–4835. [CrossRef]
38. Zhu, Y.; Zhao, D.; He, H. Invariant adaptive dynamic programming for discrete-time optimal control. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *11*, 3959–3971. [CrossRef]
39. Zhang, H.; Wei, Q.; Luo, Y. 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. Syst.* **2008**, *38*, 937–942. [CrossRef] [PubMed]
40. Li, J.; Chai, T.; Lewis, F.L.; Ding, Z.; Jiang, Y. Off-Policy Interleaved Q-Learning: Optimal Control for Affine Nonlinear Discrete-Time Systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *5*, 1308–1320. [CrossRef]
41. Sun, C.; Li, X.; Sun, Y. A parallel framework of adaptive dynamic programming algorithm with off-policy learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *8*, 3578–3587. [CrossRef] [PubMed]
42. Duan, J.; Guan, Y.; Li, S.E.; Ren, Y.; Sun, Q.; Cheng, B. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *11*, 6584–6598. [CrossRef] [PubMed]
43. Song, S.; Zhu, M.; Dai, X.; Gong, D. Model-Free Optimal Tracking Control of Nonlinear Input-Affine Discrete-Time Systems via an Iterative Deterministic Q-Learning Algorithm. *IEEE Trans. Neural Networks Learn. Syst.* **2024**, *1*, 999–1012. [CrossRef] [PubMed]