Control of Nonaffine Nonlinear Discrete-Time Systems Using Reinforcement-Learning-Based Linearly Parameterized Neural Networks

Qinmin Yang, *Student Member, IEEE*, Jonathan Blake Vance, *Member, IEEE*, and S. Jagannathan, *Senior Member, IEEE*

***Abstract*—A nonaffine discrete-time system represented by the nonlinear autoregressive moving average with eXogenous input (NARMAX) representation with unknown nonlinear system dynamics is considered. An equivalent affinelike representation in terms of the tracking error dynamics is first obtained from the original nonaffine nonlinear discrete-time system so that reinforcement-learning-based near-optimal neural network (NN) controller can be developed. The control scheme consists of two linearly parameterized NNs. One NN is designated as the critic NN, which approximates a predefined long-term cost function, and an action NN is employed to derive a near-optimal control signal for the system to track a desired trajectory while minimizing the cost function simultaneously. The NN weights are tuned online. By using the standard Lyapunov approach, the stability of the closed-loop system is shown. The net result is a supervised actor-critic NN controller scheme which can be applied to a general nonaffine nonlinear discrete-time system without needing the affinelike representation. Simulation results demonstrate satisfactory performance of the controller.**

***Index Terms*—Adaptive critic, adaptive dynamic programming, Lyapunov stability, neural network control, reinforcement learning control.**

I. Introduction

THE DESIGN of control laws for nonaffine, unknown, nonlinear, and discrete-time systems is difficult due to the inclusion of the control input inside the unknown nonlinearity. Neural networks (NNs), on the other hand, have been utilized to learn the unknown dynamics of nonlinear systems while relaxing the linear in the unknown parameter assumption. A single weight tuning layer or linearly parameterized NNs such as radial basis function networks are more powerful than a standard adaptive control [1], [2] where a system-dependent nonlinear regression matrix is not required with NNs. Moreover, a linearly parameterized NN is a compromise between computation and accuracy. Past literature [3] reports the design of adaptive NN controllers to affine unknown nonlinear discrete-time systems. For a nonaffine unknown discrete-time system, such controller techniques cannot be directly employed [9].

Manuscript received August 1, 2007; revised February 15, 2008. This work was supported in part by the NSF Grants ECCS#0327877 and ECCS#0621924, by the Department of Education GAANN program, and by the Intelligent Systems Center. This paper was recommended by Guest Editor F. L. Lewis.

The authors are with the Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65409-0040 USA (e-mail: jbvance@mst.edu).

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

Digital Object Identifier 10.1109/TSMCB.2008.926607

Reinforcement learning techniques [3]–[8] are widely used to determine the solution of the optimal control of nonlinear systems using a forward-in-time computation. However, most of them are implemented either by using iterative schemes or require that a valid model of the unknown nonlinear system dynamics be available *a priori*. Moreover, stability is not demonstrated.

To overcome the iterative schemes or the need for a model in traditional dynamic programming schemes, several appealing online NN controller design methods were introduced in [3]–[8], which were also referred to as forward dynamic programming [9] or adaptive critic designs (ACDs). The central theme of this family of methods is that the optimal control law and cost function are approximated by NNs, which are trained via backpropagation over time by using feedback from the nonlinear system instead of finding the exact minimum. Convergence analysis of the closed-loop system is normally not given. Therefore, a new NN learning algorithm based on the gradient descent rule is introduced in [4]. However, it employs a simplified binary cost function. By contrast, the work in [5] proposes a near-optimal controller design using the standard Bellman equation [9], but the method is only applicable to affine nonlinear systems.

Most frequently used nonaffine nonlinear discrete-time systems are described by the nonlinear autoregressive moving average with eXogenous input (NARMAX) representation [10]. An affinelike representation is first obtained, and then, a controller is designed. However, certain stringent assumptions are exerted (e.g., small control input signal magnitudes) which limit its applicability. In addition, certain approximation techniques are utilized wherein the higher order terms and disturbances are ignored.

By contrast, in this paper, a novel single hidden-layer tunable NN controller is introduced for nonaffine nonlinear unknown discrete-time systems. An affinelike equivalent representation in terms of error dynamics is first derived by using the mean value theorem without ignoring any higher order terms and in the presence of bounded disturbances, although transforming the nonaffine system into an affinelike is not a straightforward task.

Then, by using a quadratic-performance index as the cost function, a novel reinforcement-learning-based NN control scheme with an online learning feature is developed for the affinelike error dynamics. The entire closed-loop system consists of two NNs: 1) an action NN (or actor) to generate the optimal (or near optimal) control signal to track both the desired system output and to minimize the long-term cost function, and 2) an adaptive NN (or critic) to approximate the long-term cost function $J(x(k), u(x(k)))$ and tune the action NN weights so that a near-optimal control action can be generated. Closed-loop stability is still demonstrated. Next, the proposed control scheme is tested on a general nonaffine nonlinear discrete-time system and verified.

II. Nonaffine Nonlinear Discrete-Time System

A. System Dynamics

Consider a nonaffine nonlinear discrete-time system with disturbances written in NARMAX form [10] as

$$\begin{aligned} y(k+\tau) &= f\left(\overline{y}_k, \overline{u}_{k-1}, u(k), \overline{d}_{k+\tau-1}\right) \\ &= f\left(w_k, u(k), \overline{d}_{k+\tau-1}\right) \end{aligned} \tag{1}$$

where $w_k = [\overline{y}_k^{\mathrm{T}}, \overline{u}_{k-1}^{\mathrm{T}}]^{\mathrm{T}}$, $\overline{y}_k = [y(k), \ldots, y(k-n+1)]^{\mathrm{T}}$ is the output vector, and $\overline{u}_{k-1} = [u(k-1), \ldots, y(k-n+1)]^{\mathrm{T}}$ denotes the system input vector. The term $\overline{d}_{k+\tau-1} = [d(k+\tau-1), \ldots, d(k)]^{\mathrm{T}}$ is the disturbance vector, and τ represents the system delay or the relative degree of the system [11]. Note that the output $y(k)$ is considered measurable with initial values in a compact set S_{y_0}, and the input–output data history w_k is also measurable. The system (1) is considered controllable—a standard assumption used in all control design techniques. Furthermore, several mild assumptions are needed.

Assumption 1: The unknown nonlinear function $f(\cdot)$ in (1) is continuous and differentiable up to second order.

Assumption 2: The disturbance $d(k)$ is upper bounded $|d(k)| \leq d_M$, and the partial derivative $|\partial f/\partial d(k)| \leq D_M$ is also bounded, with D_M as a positive constant.

Assumptions 1 and 2 are commonly found in the control literature [1], [11]. With Assumption 2, by using the mean value theorem, (1) can be rewritten as

$$\begin{aligned} y(k+\tau) &= f\left(w_k, u(k), \overline{d}_{k+\tau-1}\right) \\ &= f\left(w_k, u(k), 0\right) + \delta_f^{\mathrm{T}} \overline{d}_{k+\tau-1} \\ &= f\left(w_k, u(k), 0\right) + \delta_{d_k} \end{aligned} \tag{2}$$

where

$$\begin{aligned} \delta_f &= \left[\left. \frac{\partial f}{\partial d(k+\tau-1)} \right|_{d(k+\tau-1)=d_\xi(k+\tau-1)} , \ldots, \left. \frac{\partial f}{\partial d(k)} \right|_{d(k)=d_\xi(k)} \right]^{\mathrm{T}} \\ \delta_{d_k} &= \delta_f^{\mathrm{T}} \overline{d}_{k+\tau-1} \end{aligned}$$

and $d_\xi(k)$ is between 0 and $d(k)$, or $d_\xi(k) = 0 + \lambda(d(k) - 0)$, $\lambda \in [0, 1]$.

Lemma 1: δ_{d_k} is bounded by $|\delta_{d_k}| \leq \tau D_M d_M$.

Proof: This lemma can be straightforwardly verified from (2) and Assumption 2.

The purpose of Lemma 1 is to first move the disturbance outside the nonaffine nonlinear function. Then, our objective is to design a control law to drive the system output $y(k)$ to track a desired trajectory $y_d(k)$. Before we proceed, let us construct the ideal nonaffine nonlinear discrete-time system as

$$y_n(k+\tau) = f\left(w_k, u(k), 0\right). \tag{3}$$

The ideal system is defined for the sake of analysis based on the original system by assuming that there are no disturbances.

Assumption 3: $\partial f/\partial u(k) = g(k)$ is bounded and satisfies $0 < g_{\min} \leq g(k) \leq g_{\max}$, where $g_{\min}$ and $g_{\max}$ are positive constants [2].

Assumption 4: The ideal system with no external disturbances (3) is invertibly stable [11], which means that bounded system output can be realized by bounded system input.

Assumptions 3 and 4 are commonly found in adaptive control literature. Thereafter, we can draw the conclusion that for any output trajectory $y_n(k+\tau) = f(w_k, u(k), 0)$, there exists a unique and continuous (smooth) function $u(k) = f^{-1}(w_k, y_n(k+\tau), 0)$ [11]. Next, the controller methodology is introduced.

III. Controller Methodology

A. Optimal Control

In this paper, we consider the long-term cost function as

$$\begin{aligned} J(k) = J\left(y(k), u\right) &= \sum_{i=t_0}^{\infty} \gamma^i r(k+i) \\ &= \sum_{i=t_0}^{\infty} \gamma^i \left[q\left(y(k+i)\right) + u^{\mathrm{T}}(k+i) R u(k+i) \right] \end{aligned} \tag{4}$$

where u is a given control policy, R is a positive design constant, t_0 is the initial time which can be set to zero without loss of generality, and $\gamma (0 \leq \gamma \leq 1)$ is the discount factor for the infinite-horizon problem [8]. One can observe from (4) that the long-term cost function is the discounted sum of the short-term cost or Lagrangian which is expressed as

$$\begin{aligned} r(k) &= q\left(y(k)\right) + u^{\mathrm{T}}(k) R u(k) \\ &= \left(y(k) - y_d(k)\right)^{\mathrm{T}} Q \left(y(k) - y_d(k)\right) + u^{\mathrm{T}}(k) R u(k) \\ &= Q e^2(k) + R u^2(k) \end{aligned} \tag{5}$$

where Q is a positive design constant. In this paper, we are using a widely applied standard quadratic cost function defined based on the tracking error $e(k) = y(k) - y_d(k)$ in contrast with that in [5]. The cost function $r(k)$ can also be viewed as the system performance index for the current step.

As a matter of fact, for an optimal control law, which can be expressed as $u*(k) = u*(y(k))$, the optimal long-term cost function can be written as $J*(k) = J*(y(k), u*(y(k))) = J*(k)$, which is just a function of the current system output [9]. Next, one can state the following assumption.

Assumption 5: The optimal cost function $J*(k)$ is finite and bounded over the compact set $S \subset R$ by J_M.

The optimal cost is the minimum over the entire control space.

B. Affinelike Dynamics

By applying the Taylor series expansion of (3) up to a second order with respect to $u(k)$ about the point $u(k-1)$ yields

$$\begin{aligned} y(k+\tau) &= f(w_k, u(k), 0) + \delta_{d_k} \\ &= f(w_k, u(k-1), 0) + \frac{\partial f(w_k, u(k-1), 0)}{\partial u}\Delta u(k) \\ &\quad + \frac{1}{2}\cdot\frac{\partial^2 f(w_k, u_\mu(k), 0)}{\partial u^2}\Delta u^2(k) + \delta_{d_k} \\ &= F(w_k, u(k)) + G(w_k)\Delta u(k) + \delta_{d_k} \end{aligned} \tag{6}$$

where

$$\begin{aligned} F(w_k, u(k)) &= f(w_k, u(k-1), 0) \\ &\quad + \frac{1}{2}\cdot\frac{\partial^2 f(w_k, u_\mu(k), 0)}{\partial u^2}\Delta u^2(k) \\ G(w_k) &= \frac{\partial f(w_k, u(k-1), 0)}{\partial u} \end{aligned}$$

and $u_\mu(k)$ is between $u(k)$ and $u(k+1)$ (or $u_\mu(k) = u(k+1) + \lambda(u(k+1) - u(k)), \lambda \in [0,1]$) by using the mean value theorem. In other words, no higher order terms in the Taylor series expansion are ignored, since they are incorporated into the second derivative. By observing (6), the ideal system (3) can be expressed as

$$y_n(k+\tau) = F(w_k, u(k)) + G(w_k)\Delta u(k). \tag{7}$$

The following lemmas are needed in order to transform the system into an equivalent affinelike form and to verify that both are equivalent.

Lemma 2: Considering any desired system trajectory $y_d(k+\tau) \in S$ and corresponding nominal desired control input $u_d(k) = f^{-1}(w_k, y_d(k+\tau), 0)$, there exists $u_\xi(k)$ between any nominal control input $u_n(k)$ and $u_d(k)$ to the system such that

$$\begin{aligned} F(w_k, u_n(k)) &= F(w_k, u_d(k)) + \frac{\partial F(w_k, u_\xi(k))}{\partial u} \\ &\times \frac{\partial f^{-1}(w_k, y_\xi(k+\tau), 0)}{\partial y}\cdot(y_n(k+\tau) - y_d(k+\tau)) \end{aligned} \tag{8}$$

where $u_\xi(k) = f^{-1}(w_k, y_\xi(k+\tau), 0)$.

Lemma 3: Considering the output of the ideal nonaffine system $y_n(k+\tau) = f(w_k, u_n(k), 0)$ for a given input $u_n(k)$, then there exists $y_\varsigma(k+\tau)$ between $y_n(k+\tau)$ and $y_d(k+\tau)$ such that

$$\begin{aligned} u_n(k) &= f^{-1}(w_k, y_n(k+\tau), 0) \\ &= f^{-1}(w_k, y_d(k+\tau), 0) \\ &\quad + \frac{\partial f^{-1}(w_k, y_\varsigma(k+\tau), 0)}{\partial y}(y_n(k+\tau) - y_d(k+\tau)) \\ &= u_d(k) + \frac{\partial f^{-1}(w_k, y_\varsigma(k+\tau), 0)}{\partial y}(y_n(k+\tau) - y_d(k+\tau)). \end{aligned} \tag{9}$$

Proof: Lemmas 2 and 3 can be readily obtained by using chain rule and mean value theorem.

Remark 1: Lemmas 2 and 3 locate two control-output pairs $(u_\xi(k), y_\xi(k+\tau))$ and $(u_\varsigma(k), y_\varsigma(k+\tau))$ for any control-output $(u_n(k), y_n(k+\tau))$ given the desired value $(u_d(k), y_d(k+\tau))$ satisfying (8) and (9) , respectively, where $u_\varsigma(k) = f^{-1}(w_k, y_\varsigma(k+\tau), 0)$. Thereafter, their relationship is investigated through following lemmas.

Lemma 4: Considering system (7) with Lemmas 2 and 3, we have

$$\begin{aligned} &\frac{\partial F(w_k, u_\xi(k))}{\partial u}\cdot\frac{\partial f^{-1}(w_k, y_\xi(k+\tau), 0)}{\partial y} \\ &\qquad + G(w_k)\frac{\partial f^{-1}(w_k, y_\varsigma(k+\tau), 0)}{\partial y} = 1. \end{aligned} \tag{10}$$

Lemma 5: For any $y_\varsigma(k+\tau) \in S$ and corresponding control input $u_\varsigma(k) = f^{-1}(w_k, y_\varsigma(k+\tau), 0)$, the following statement holds:

$$\frac{\partial f(w_k, u_\varsigma(k), 0)}{\partial u}\cdot\frac{\partial f^{-1}(w_k, y_\varsigma(k+\tau))}{\partial y} = 1. \tag{11}$$

Proof: It can be straightforward to verify (11) by differentiating $y_\varsigma(k+\tau) = f(w_k, f^{-1}(w_k, y_\varsigma(k+\tau)), 0)$ with respect to $y_\varsigma(k+\tau)$.

The aforementioned lemma shows that the nonaffine dynamics can be transformed into an equivalent affinelike form. Therefore, substituting (8) into (6) produces the system dynamics in terms of the tracking error as

$$\begin{aligned} e(k+\tau) &= y(k+\tau) - y_d(k+\tau) \\ &= F(w_k, u(k)) + G(w_k)\Delta u(k) + \delta_{d_k} - y_d(k+\tau) \\ &= F(w_k, u_d(k)) + \frac{\partial F(w_k, u_\xi(k))}{\partial u} \\ &\quad \times \frac{\partial f^{-1}(w_k, y_\xi(k+\tau), 0)}{\partial y}\cdot(y(k+\tau) - y_d(k+\tau)) \\ &\quad + G(w_k)\Delta u(k) + \delta_{d_k} - y_d(k+\tau). \end{aligned} \tag{12}$$

Making use of Lemma 4, (12) can be written as

$$\begin{aligned} e(k+\tau) &= F(w_k, u_d(k)) + G(w_k)\Delta u(k) + \delta_{d_k} - y_d(k+\tau) \\ &\quad + \left(1 - G(w_k)\frac{\partial f^{-1}(w_k, y_\varsigma(k+\tau), 0)}{\partial y}\right) \\ &\quad \cdot (y(k+\tau) - y_d(k+\tau)) \\ &= F(w_k, u_d(k)) + G(w_k)\Delta u(k) + \delta_{d_k} - y_d(k+\tau) \\ &\quad + \left(1 - G(w_k)\frac{\partial f^{-1}(w_k, y_\varsigma(k+\tau), 0)}{\partial y}\right) \\ &\quad \cdot e(k+\tau). \end{aligned} \tag{13}$$

Combing (11) and (13), one has

$$\begin{aligned} e(k+\tau) &= \frac{\partial f(w_k, u_\varsigma(k), 0)}{\partial u} \\ &\quad \times \left(\frac{F(w_k, u_d(k)) + \delta_{d_k} - y_d(k+\tau)}{G(w_k)} + \Delta u(k)\right). \end{aligned} \tag{14}$$

By defining $\partial f(w_k, u_\varsigma(k), 0)/\partial u = \kappa_k$, (14) can be rephrased as

$$\begin{aligned} e(k+\tau) &= \frac{\kappa_k}{G(w_k)}(F(w_k, u_d(k)) - y_d(k+\tau)) \\ &\quad + \kappa_k\Delta u(k) + \frac{\kappa_k}{G(w_k)}\delta_{d_k} \\ &= F_a(w_k, y_d(k+\tau), \kappa_k) + \kappa_k\Delta u(k) + \delta_{\kappa_k} \end{aligned} \tag{15}$$

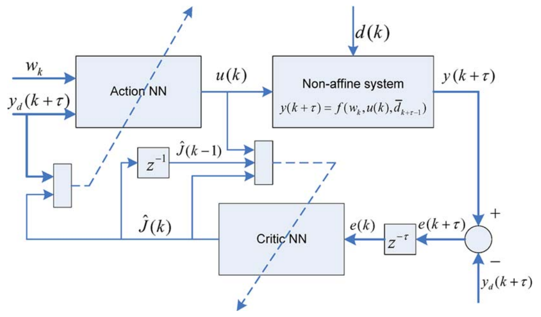

Fig. 1. Online reinforcement learning neural controller structure.

where $F_a(w_k, y_d(k+\tau), \kappa_k) = (\kappa_k/G(w_k))(F(w_k, u_d(k)) - y_d(k+\tau))$ and $\delta_{\kappa_k} = (\kappa_k/G(w_k))\delta_{d_k}$. Notice that $0 < g_{\min} \le \kappa_k \le g_{\max}$, $0 < g_{\min} \le G(w_k) \le g_{\max}$ due to Assumption 3. By referring to Lemma 1, one also observes that δ_{κ_k} is bounded above by $|\delta_{\kappa_k}| \le g_{\max}\tau D_M d_M/g_{\min}$.

By rewriting the nonaffine nonlinear discrete-time system into an equivalent affinelike representation described by (15) in terms of error dynamics and designing a controller for the affinelike system, the difficulty of designing controllers for original nonaffine systems can be overcome.

C. Online Controller Design

The objective is to design an online reinforcement learning NN controller for (15) such that the following are satisfied: 1) all the signals in the closed-loop system remain semi uniformly ultimately bounded (SUUB); 2) the output $y(k)$ follows a desired trajectory $y_d(k) \in S$; and 3) the long-term cost function (4) is minimized so that a near-optimal control input can be generated [5]. The critic and action NN weight matrices are initialized at zero and trained online.

The block diagram of the proposed controller is shown in Fig. 1, where the action NN provides the control signal to the nonlinear discrete-time system, whereas the critic NN approximates the long-term cost function. The two NN weight matrices are initialized at zero and trained online without any offline learning phase.

Now, any continuous and differentiable function up to the Nth degree $h(X) \in C^N(S)$ on a compact set S can be written as [12]

$$h(X) = W^{\mathrm{T}}\phi(V^{\mathrm{T}}X) + \varepsilon(X) \tag{16}$$

with $\varepsilon(X)$ as an NN functional reconstruction error vector. In our design, the output layer weight matrix W is adapted online, whereas the input layer weight matrix V is selected initially at random and held fixed during the entire learning process. It is demonstrated in [12] that if the number of hidden layer neurons is sufficiently large, the NN approximation error $\varepsilon(X)$ can be made arbitrarily small. According to Igelnik and Pao [12], The linearly parameterized NN turns out to be a practical compromise between a nonlinear multilayer NN and the NN with fixed basis functions, combining both simplicity of learning and efficiency of representation. Next, a novel weight tuning algorithm is proposed after stating a mild assumption.

Assumption 6: The desired trajectory $y_d(k)$ of the system output is bounded over the compact subset of S.

Next, the action and critic NN design and their weight tuning are discussed.

D. Action NN Design

Consider the affinelike representation given by (15), and select a desired control law

$$u_d(k) = u(k-1) - \frac{1}{\kappa_k}F_a\left(w_k, y_d(k+\tau), \kappa_k\right) \tag{17}$$

or desired change of control signal for the current step

$$\Delta u_d(k) = -\frac{1}{\kappa_k}F_a\left(w_k, y_d(k+\tau), \kappa_k\right). \tag{18}$$

By substituting (18) into (15), we get

$$\begin{aligned} e(k+\tau) &= F_a\left(w_k, y_d(k+\tau), \kappa_k\right) + \kappa_k \\ &\quad \times \left(-\frac{1}{\kappa_k}F_a\left(w_k, y_d(k+\tau), \kappa_k\right)\right) + \delta_{\kappa_k} \\ &= 0, \qquad \text{if } \delta_{\kappa_k} = 0. \end{aligned} \tag{19}$$

Therefore, the selection of the control law ensures the convergence of the tracking error to zero after τ steps if no disturbance is acting on the system.

However, since both of $F_a(w_k, y_d(k+\tau), \kappa_k)$ and κ_k are unknown smooth nonlinear functions describing the dynamics of the original nonaffine nonlinear system, the feedback control $u_d(k)$ cannot be implemented in practice. Instead, an action NN is employed to produce the control signal. From (17) and considering Assumptions 3 and 4, the control signal can be approximated by using an action NN given by

$$u_d(k) = w_a^{\mathrm{T}}\phi_a\left(v_a^{\mathrm{T}}s(k)\right) + \varepsilon_a\left(s(k)\right) = w_a^{\mathrm{T}}\phi_a(k) + \varepsilon_a(k) \tag{20}$$

where $s(k) = [w_k^{\mathrm{T}}, y_d(k+\tau)]^{\mathrm{T}}$ is the action NN input vector, n_a is the number of neurons in the hidden layer, and $w_a \in R^{n_a \times 1}$ and $v_a \in R^{2n \times n_a}$ denote the target weights of the output and hidden layer, respectively, with $\varepsilon_a(k) = \varepsilon_a(s(k))$ as the action NN functional approximation error. Since the input layer weight matrix v_a is fixed, the hidden layer activation function vector $\phi_a(v_a^{\mathrm{T}}s(k)) \in R^{n_a}$ is denoted as $\phi_a(k)$.

The actual NN output can be expressed as

$$u(k) = \hat{w}_a^{\mathrm{T}}(k)\phi_a\left(v_a^{\mathrm{T}}s(k)\right) = \hat{w}_a^{\mathrm{T}}(k)\phi_a(k) \tag{21}$$

where $\hat{w}_a(k) \in R^{n_a \times 1}$ is the weight matrix of the output layer.

Using the action NN output as the control signal and substituting (20) and (21) into (15) yield the error dynamics

$$\begin{aligned} e(k+\tau) &= F_a\left(w_k, y_d(k+\tau), \kappa_k\right) + \kappa_k \Delta u(k) + \delta_{\kappa_k} \\ &= \kappa_k\left(u(k) - u_d(k)\right) + \delta_{\kappa_k} \\ &= \kappa_k\left(\tilde{w}_a^{\mathrm{T}}(k)\phi_a(k) - \varepsilon_a(k)\right) + \delta_{\kappa_k} \\ &= \kappa_k\zeta_a(k) + d_a(k) \end{aligned} \tag{22}$$

where the weight estimation error of the actual NN is given by

$$\tilde{w}_a(k) = \hat{w}_a(k) - w_a \tag{23}$$

$$\zeta_a(k) = \tilde{w}_a^{\mathrm{T}}(k)\phi_a(k) \tag{24}$$

with

$$d_a(k) = -\kappa_k \varepsilon_a(k) + \delta_{\kappa_k}. \tag{25}$$

E. Critic NN Design

In order to stabilize the closed-loop system along with minimizing the cost function, a critic NN is employed to approximate the unknown long-term cost function $J(k)$. First, the prediction error generated by the critic or the Bellman error [4] is defined as

$$e_c(k) = \gamma \hat{J}(k) - \hat{J}(k-1) + r(k) \tag{26}$$

where the subscript "c" stands for the "critic" and

$$\hat{J}(k) = \hat{w}_c^{\mathrm{T}}(k)\phi_c\left(v_c^{\mathrm{T}}e(k)\right) = \hat{w}_c^{\mathrm{T}}(k)\phi_c(k) \tag{27}$$

with $\hat{J}(k) \in R$ is the critic NN output that approximates $J(k)$. The actual output layer weight matrix is denoted by $\hat{w}_c(k) \in R^{n_c \times 1}$, and $v_c \in R^{1 \times n_c}$ represents the input weights which will be selected at random initially and held thereafter. The term n_c denotes the number of the neurons in the hidden layer of the critic NN. Here, the tracking error $e(k)$ is selected as the critic NN input since this information is available, making the proposed scheme a variant of heuristic dynamic programming. Again for convenience, the activation function vector of the hidden layer $\phi_c(v_c^{\mathrm{T}}e(k)) \in R^{n_c}$ is simply denoted as $\phi_c(k)$. The optimal long-term cost function $J*(k)$ can be approximated with arbitrarily small reconstruction error $\varepsilon_c(k)$. The optimal cost function can be expressed as

$$J*(k) = w_c^{\mathrm{T}}\phi_c\left(v_c^{\mathrm{T}}e(k)\right) + \varepsilon_c\left(e(k)\right) = w_c^{\mathrm{T}}\phi_c(k) + \varepsilon_c(k). \tag{28}$$

The critic NN output layer weight estimation error can be defined as

$$\tilde{w}_c(k) = \hat{w}_c(k) - w_c \tag{29}$$

where

$$\zeta_c(k) = \tilde{w}_c^{\mathrm{T}}(k)\phi_c(k). \tag{30}$$

Thus, we obtain

$$\begin{aligned} e_c(k) = \gamma\zeta_c(k) + \gamma J*(k) - \zeta_c(k-1) - J*(k-1) \\ + r(k) - \varepsilon_c(k) + \varepsilon_c(k-1). \end{aligned} \tag{31}$$

Next, the NN weight update laws are introduced.

F. Weight Update for the Critic NN

Select the objective function to be minimized by the critic NN as a quadratic function of Bellman error as

$$E_c(k) = \frac{1}{2}e_c^{\mathrm{T}}(k)e_c(k) = \frac{1}{2}e_c^2(k). \tag{32}$$

By using a standard gradient-based adaptation method, the weight updating algorithm for the critic NN is given by

$$\hat{w}_c(k+\tau) = \hat{w}_c(k) + \Delta\hat{w}_c(k) \tag{33}$$

where

$$\Delta\hat{w}_c(k) = \alpha_c\left[-\frac{\partial E_c(k)}{\partial \hat{w}_c(k)}\right] \tag{34}$$

with $\alpha_c \in R$ as the adaptation gain.

Combining (26), (27), and (32) with (34), the critic NN weight updating rule can be obtained by using the chain rule as

$$\begin{aligned} \Delta\hat{w}_c(k) &= -\alpha_c\frac{\partial E_c(k)}{\partial \hat{w}_c(k)} = -\alpha_c\frac{\partial E_c(k)}{\partial e_c(k)}\frac{\partial e_c(k)}{\partial \hat{J}(k)}\frac{\partial \hat{J}(k)}{\partial \hat{w}_c(k)} \\ &= -\alpha_c\gamma\phi_c(k)\left(\gamma\hat{J}(k) + r(k) - \hat{J}(k)\right). \end{aligned} \tag{35}$$

Remark 2: The tracking error signal in the critic NN weight update can be viewed as a supervisory signal in the actor-critic controller [8], providing an additional source of evaluation or reward. As this error becomes close to zero, it can be viewed as gradual withdrawal of the additional feedback to shape the learned policy toward optimality. The supervisor may override bad commands from the critic by providing stability initially until the critic NN begins to learn in order to ensure safety and guarantee minimum standard of performance [8].

G. Weight Update for the Action NN

The objective for adapting the action NN is to track the desired output while lowering the long-term cost function. Therefore, the action NN error can be formed by using the functional estimation error $\zeta_a(k)$ and the critic signal $\hat{J}(k)$ as

$$\begin{aligned} e_a(k) &= \sqrt{\kappa_k}\zeta_a(k) + (\sqrt{\kappa_k})^{-1}\left(\hat{J}(k) - J_d(k)\right) \\ &= \sqrt{\kappa_k}\zeta_a(k) + (\sqrt{\kappa_k})^{-1}\hat{J}(k) \end{aligned} \tag{36}$$

where $\zeta_a(k)$ is defined in (24). The target long-term cost function $J_d(k)$ is considered to be zero ("0"), which implies that it is small as possible.

The action NN weights $\hat{w}_a(k)$ are tuned to minimize

$$E_a(k) = \frac{1}{2}e_a^{\mathrm{T}}(k)e_a(k). \tag{37}$$

Combining (22), (24), (36), and (37) and using the chain rule yield

$$\begin{aligned} \Delta\hat{w}_a(k) &= -\alpha_a\frac{\partial E_a(k)}{\partial \hat{w}_a(k)} \\ &= -\alpha_a\frac{\partial E_a(k)}{\partial e_a(k)}\frac{\partial e_a(k)}{\partial \zeta_a(k)}\frac{\partial \zeta_a(k)}{\partial \hat{w}_c(k)} \\ &= -\alpha_a\phi_a(k)\left(\kappa_k\zeta_a(k) + \hat{J}(k)\right)^{\mathrm{T}} \\ &= -\alpha_a\phi_a(k)\left(e(k+\tau) - d_a(k) + \hat{J}(k)\right)^{\mathrm{T}} \end{aligned} \tag{38}$$

where $\alpha_a \in R^+$ is the adaptation gain of the action NN. Since the unknown bounded disturbances $d_a(k)$ are typically unavailable, we assume the $d(k)$ and the mean value of $\varepsilon_a(k)$ over the compact subset of R to be zero [5] and obtain the weight updating algorithm for the action NN as

$$\hat{w}_a(k+\tau) = \hat{w}_a(k) - \alpha_a \phi_a(k)\left(e(k+\tau) + \hat{J}(k)\right)^{\mathrm{T}}. \quad (39)$$

Remark 3: Here, too, the tracking error acts like a supervisory [8] signal. As a consequence, the action NN weights are driven to near-optimal weights.

IV. Controller Design

Assumption 7: Let the unknown target output layer weights for the action and critic NNs be upper bounded such that

$$\|w_a\| \le w_{\mathrm{am}} \qquad \|w_c\| \le w_{\mathrm{cm}} \quad (40)$$

where $w_{\mathrm{am}} \in R^+$ and $w_{\mathrm{cm}} \in R^+$ represent the upper bounds. Here, $\|\cdot\|$ stands for the Frobenius norm [2].

Assumption 8: The activation functions for the action and critic NNs are bounded by known positive values, such that

$$\|\phi_a(k)\| \le \phi_{\mathrm{am}} \qquad \|\phi_c(k)\| \le \phi_{\mathrm{cm}} \quad (41)$$

where $\phi_{\mathrm{am}}, \phi_{\mathrm{cm}} \in R^+$ is the upper bound due to the hyperbolic tangent function selected for the hidden layer.

Assumption 9: The NN approximation errors for the action and critic NNs, $\varepsilon_a(k)$ and $\varepsilon_c(k)$, respectively, are bounded above over the compact set $S \subset R$ by $\varepsilon_{\mathrm{am}}$ and $\varepsilon_{\mathrm{cm}}$ [2].

Assumptions 7–9 are commonly used in NN control [2].

Lemma 6: With the Assumptions 3 and 9 and Lemma 1, the term $d_a(k)$ in (25) is bounded over the compact set $S \subset R$ by

$$|d_a(k)| \le d_{\mathrm{am}} = g_{\max}\varepsilon_{\mathrm{am}} + g_{\max}\tau D_M d_M / g_{\min}. \quad (42)$$

Combining Assumptions 1, 3, and 4 and Lemma 6, the main result in the form of a theorem is introduced next.

Theorem 1: Consider the nonlinear discrete-time system given by (2) whose dynamics can be expressed as (15). Let the Assumptions 1–9 hold with the disturbance bound d_M a known constant. Let the control input be provided by the action NN (21) with the critic NN output given by (27). Let the action NN and the critic NN weights be tuned by (35) and (39), respectively. Then, the tracking error $e(k)$ and the NN weight estimates of the action and critic NNs, $\tilde{w}_a(k)$ and $\tilde{w}_c(k)$, are semi global uniformly ultimately bounded (SUUB) with the bounds given by (A8) provided that the controller design parameters are selected as

$$0 < \alpha_a \phi_a^2(k) < \frac{g_{\min}}{g_{\max}^2} \qquad 0 < \alpha_c \phi_c^2(k) < 1/\gamma^2 \quad (43)$$

$$\gamma > \frac{1}{2} \quad (44)$$

where α_a and α_c are the NN adaptation gains and γ is employed to define the strategic utility function.

Proof: See the Appendix.

Remark 4: The need for an exact model of the nonlinear discrete-time system in many ACD approaches [7] is relaxed in this work through the supervised actor-critic architecture.

Remark 5: No explicit offline training phase is necessary. In addition, the proposed methodology does not require the stop/reset strategy utilized by adaptive critic schemes [7].

Remark 6: By using (35), one can show that the approximate cost function converges to a near-optimal cost which, in turn, is used for tuning the action NN weights (39). The action NN is functioning well when the tracking error signal is near zero. Then, the action NN weights will be driven by the approximate cost function to attain near-optimal weights. As a result, the action NN renders a near-optimal control input due to approximation errors and bounded disturbances. Existing ACDs ignore the approximation errors and bounded disturbances [7].

Remark 7: Equation (43) relates the selection of adaptation gains with the discount factor, whereas (44) provides how the discount factor can be chosen in order to ensure stability and convergence. Normally, the discount factor and adaptation gains are selected by trial and error.

V. Simulations

Consider the following second-order input–output nonaffine discrete-time system with delay given by

$$\begin{aligned} y(k+2) = {} & 0.2\cos(0.8(y(k)+y(k-1))) \\ & +0.4\sin(0.8(y(k)+y(k-1))+2u(k)+u(k-1)) \\ & +0.1(9+y(k)+y(k-1)) + \frac{2(u(k)+u(k-1))}{1+\cos(y(k))} + d \end{aligned} \quad (45)$$

where the control action starts at $k=1$ with the initial conditions given by $y(0)=0$, $u(0)=0$. We can observe that the system (45) satisfies Assumption 1, since the right-hand side of (45) is continuous and differentiable up to infinite order. A bounded uniformly distributed disturbance d will be used, and therefore, Assumption 2 is satisfied. Differentiating (45) with respect to $u(k)$ yields

$$\begin{aligned} \frac{\partial f}{\partial u(k)} = g(k) = {} & 0.8\cos(0.8(y(k)+y(k-1)) \\ & + 2u(k) + u(k-1)) + \frac{2}{1+\cos(y(k))}. \end{aligned}$$

It will be readily verified that $0.2 \le g(k) < \infty$ and Assumption 3 holds. Furthermore, the right-hand side of (45) is a monotonically increasing continuous function with respect to variable $u(k)$. As a result, for any desired $y(k+2)$ and given data-history $y(k)$, $y(k-1)$, $u(k-1)$, there exists a control input $u(k)$ satisfying (45). Hence, Assumption 4 is met. Moreover, from (45), the system delay is given by $\tau = 2$.

The reference output trajectory is selected as

$$y_d(k) = \begin{cases} 0.8 + 0.05\left(\sin\left(\frac{\pi k}{50}\right) + \sin\left(\frac{\pi k}{100}\right) + \sin\left(\frac{\pi k}{150}\right)\right), & \text{for } k > 0 \\ 0, & \text{for } k \le 0. \end{cases} \quad (46)$$

The objective of our controller is to drive the system to track the reference trajectory. The major challenge of this control problem is that the system does not appear to be in a standard affine form because the control input u appears nonlinearly within the dynamics. The saturation value of control input is considered as 0.5. The sampling time interval is set as 0.02 s, and a uniformly distributed noise d is considered bounded with an upper bound d_M. Other parameters are listed in Table I.

TABLE I
SUMMARY OF PARAMETERS

Parameter	R	Q	γ	α_c
Value	0.05	0.01	0.8	0.001
Parameter	α_a	n_a	n_c	d_M
Value	0.05	10	10	0.01

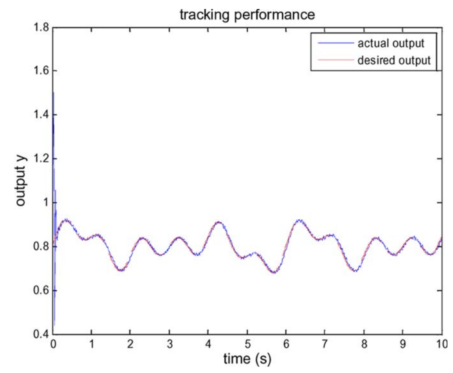

Fig. 2. Tracking performance of the online learning controller.

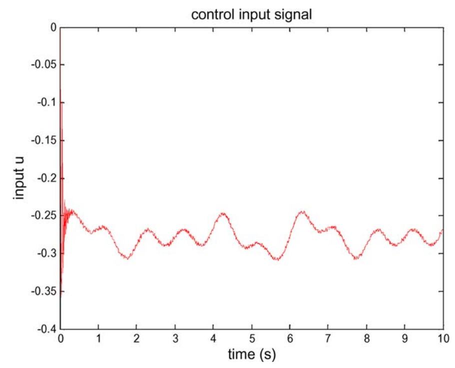

Fig. 3. Input signal of the online learning controller.

The adaptation gains of the NNs are chosen to satisfy (43) and (44) as shown in Table I. Furthermore, their output layer weights are initialized at zero, whereas their hidden layer weights are initialized at random. The tracking performance with the controller is shown in Fig. 2, which demonstrates a satisfactory performance even under the influence of noise, whereas the control input in Fig. 3 is bounded. Here, during the controller implementation, the nonaffine system is not transformed into an affinelike form.

VI. CONCLUSION

This paper presents a technique to obtain an equivalent affinelike system representation for a class of nonaffine systems in NARMAX form without losing any information. Bounded disturbance is also integrated to imitate practical applications. By using the system transformation, it becomes easier to conduct controller design and stability analysis.

The online control design using the supervisory actor-critic architecture and linearly parameterized NN renders closed-loop stability and relaxes the need for trial and error procedure of selecting controller parameters. The performance of the controller is tested on a general nonaffine nonlinear discrete-time system, and satisfactory performance was observed. Finally, the system transformation to affinelike form is only needed for controller design, whereas it is not required during application on nonaffine discrete-time systems.

APPENDIX

Proof of Theorem 1: Define a Lyapunov candidate as

$$\begin{aligned} L(k) = \sum_{i=1}^{4} L_i = \frac{\gamma_1}{3}\sum_{j=0}^{\tau-1} e^2(k+j) \\ + \frac{\gamma_2}{\alpha_a}\sum_{j=0}^{\tau-1} tr\left(\tilde{w}_a^{\mathrm{T}}(k+j)\tilde{w}_a(k+j)\right) \\ + \frac{\gamma_3}{\alpha_c}\sum_{j=0}^{\tau-1} tr\left(\tilde{w}_c^{\mathrm{T}}(k+j)\tilde{w}_c(k+j)\right) \\ + \gamma_4\sum_{j=0}^{\tau-1}\zeta_c^2(k+j) \end{aligned} \tag{A1}$$

where $\gamma_i \in R^+$, $i = 1, 2, 3, 4$, are design parameters. Hence, the first difference of the Lyapunov function is given by

$$\begin{aligned} \Delta L_1 &= \frac{\gamma_1}{2}\left(e^2(k+\tau) - e^2(k)\right) \\ &= \frac{\gamma_1}{2}\left(\left(\kappa_k\zeta_a(k) + d_a(k)\right)^2 - e^2(k)\right) \\ &\leq -\frac{\gamma_1}{2}e^2(k) + \gamma_1 g_{\max}^2\zeta_a^2(k) + \gamma_1 d_a^2(k). \end{aligned} \tag{A2}$$

Set $\gamma_2 = \gamma_2'\gamma_2''$, where $\gamma_2''((1-\alpha_a\phi_a^2(k)g_{\min})/(g_{\min} - \alpha_a\phi_a^2(k)g_{\max}^2)) \leq (1/2)$; therefore

$$\begin{aligned} \Delta L_2 \leq& -\gamma_2 g_{\min}\zeta_a^2(k) - \gamma_2\left(g_{\min} - \alpha_a\phi_a^2(k)g_{\max}^2\right) \\ &\times\left(\zeta_a(k) + \frac{\left(I - \alpha_a\phi_a^2(k)\kappa_k\right)}{g_{\min} - \alpha_a\phi_a^2(k)g_{\max}^2}\right)^2 \\ &+ \gamma_2'\frac{\left(\hat{J}(k) + d_a(k)\right)^2}{2} \\ \leq& -\gamma_2 g_{\min}\zeta_a^2(k) - \gamma_2\left(g_{\min} - \alpha_a\phi_a^2(k)g_{\max}^2\right) \\ &\times\left(\zeta_a(k) + \frac{\left(I - \alpha_a\phi_a^2(k)\kappa_k\right)}{g_{\min} - \alpha_a\phi_a^2(k)g_{\max}^2}\right)^2 \\ &+ \gamma_2'\zeta_c^2(k) + \gamma_2'\left(J*(k) + d_a(k)\right)^2. \end{aligned} \tag{A3}$$

Similarly

$$\begin{aligned} \Delta L_3 \leq& -\gamma_3\left(1 - \alpha_c\gamma^2\phi_c^2(k)\right)e_c^2(k) \\ &- \gamma_3\gamma^2\zeta_c^2(k) + \frac{\gamma_3}{4}\zeta_c^2(k-1) \\ &+ \frac{\gamma_3}{4}\left(\gamma J*(k) - J*(k-1)\right)^2 + \frac{\gamma_3}{4}Qe^2(k) \\ &+ \frac{\gamma_3}{8}R\zeta_a^2(k) + \frac{\gamma_3}{8}R\left(w_a^{\mathrm{T}}\phi_a(k)\right)^2 + \gamma_3\varepsilon_{\mathrm{cm}}^2 \end{aligned} \tag{A4}$$

$$\Delta L_4 = \gamma_4\left(\zeta_c^2(k) - \zeta_c^2(k-1)\right). \tag{A5}$$

Combining (A2), (A3), (A4), and (A5) yields

$$\begin{aligned}\Delta L(k) =& -\left(\frac{\gamma_1}{2}-\frac{\gamma_3}{4}Q\right)e^2(k)\\ &-\left(\gamma_2 g_{\min}-\gamma_1 g_{\max}^2-\frac{\gamma_3}{8}R\right)\zeta_a^2(k)\\ &-\left(\gamma_3\gamma^2-\gamma_2'-\gamma_4\right)\zeta_c^2(k)-\left(\gamma_4-\frac{\gamma_3}{4}\right)\zeta_c^2(k-1)\\ &-\gamma_3\left(1-\alpha_c\gamma^2\phi_c^2(k)\right)e_c^2(k)\\ &-\gamma_2\left(g_{\min}-\alpha_a\phi_a^2(k)g_{\max}^2\right)\\ &\times\left(\zeta_a(k)+\frac{\left(1-\alpha_a\phi_a^2(k)\kappa_k\right)}{g_{\min}-\alpha_a\phi_a^2(k)g_{\max}^2}\right)^2+D_M^2 \quad \text{(A6)}\end{aligned}$$

where

$$\begin{aligned}D_M^2 =& \left(\gamma_1+\frac{\gamma_2'}{2}\right)d_{\text{am}}^2+\left(\frac{\gamma_2'}{2}+\frac{\gamma_3}{4}(\gamma+1)^2\right)J_M^2\\ &+\frac{\gamma_3}{8}Rw_{\text{am}}^2\phi_{\text{am}}^2+\gamma_3\varepsilon_{\text{cm}}^2. \quad \text{(A7)}\end{aligned}$$

For the standard Lyapunov analysis, (A6) implies that $\Delta L \leq 0$ as long as the conditions (43) and (44) are satisfied and the following holds:

$$\|e(k)\| \geq \frac{2D_M}{\sqrt{2\gamma_1-\gamma_3 Q}} \quad \text{or}$$

$$\|\zeta_a(k)\| \leq \frac{2\sqrt{2}D_M}{\sqrt{8\gamma_2 g_{\min}-8\gamma_1 g_{\max}^2-\gamma_3 R}}$$

or

$$\|\zeta_c(k)\| \leq \frac{D_M}{\sqrt{\gamma_3\gamma^2-\gamma_2'-\gamma_4}}. \quad \text{(A8)}$$

According to the standard Lyapunov extension theorem [2], the aforementioned analysis demonstrates that the tracking error $\|e(k)\|$ and the weights of the estimation errors are SUUB.

References

[1] M. Krstic, I. Kanellakopoulos, and P. Kokotovic, *Nonlinear and Adaptive Control Design*. New York: Wiley, 1995.

[2] S. Jagannathan, *Neural Network Control of Nonlinear Discrete-time Systems*. Boca Raton, FL: Taylor & Francis, 2006.

[3] D. Prokhorov and D. Wunsch, “Adaptive critic designs,” *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997–1007, Sep. 1997.

[4] J. Si and Y. T. Wang, “On-line learning control by association and reinforcement,” *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, Mar. 2001.

[5] Q. Yang and S. Jagannathan, “Online reinforcement learning-based neural network controller design for affine nonlinear discrete-time systems,” in *Proc. Amer. Control Conf.*, 2007, pp. 4774–4779.

[6] P. J. Werbos, “A menu of designs for reinforcement learning over time,” in *Neural Networks for Control*, W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds. Cambridge, MA: MIT Press, 1990, ch. 3.

[7] J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, “Adaptive dynamic programming,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 32, no. 2, pp. 140–153, May 2002.

[8] M. T. Rosenstein and A. G. Barto, “Supervised actor-critic reinforcement learning,” in *Handbook of Learning and Approximate Dynamic Programming*, J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, Eds. Piscataway, NJ: IEEE Press, 2004, pp. 359–377.

[9] D. P. Bertsekas, *Dynamic Programming and Optimal Control. Belmont*. Belmont, MA: Athena Scientific, 2000.

[10] O. Adetona, S. Sathananthan, and L. H. Keel, “Robust adaptive control of nonaffine nonlinear plants with small input signal changes,” *IEEE Trans. Neural Netw.*, vol. 15, no. 2, pp. 408–416, Mar. 2004.

[11] M. S. Ahmed, “Neural-net-based direct adaptive control for a class of nonlinear plants,” *IEEE Trans. Autom. Control*, vol. 45, no. 1, pp. 119–124, Jan. 2000.

[12] B. Igelnik and Y. H. Pao, “Stochastic choice of basis functions in adaptive function approximation and the functional-link net,” *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, Nov. 1995.

Author photographs and biographies not available at the time of publication.