01 Jan 2007

Near Optimal Neural Network-Based Output Feedback Control of Affine Nonlinear Discrete-Time Systems

Qinmin Yang

Jagannathan Sarangapani
Missouri University of Science and Technology, sarangap@mst.edu

Recommended Citation

Q. Yang and J. Sarangapani, "Near Optimal Neural Network-Based Output Feedback Control of Affine Nonlinear Discrete-Time Systems," *Proceedings of the IEEE 22nd International Symposium on Intelligent Control (2007, Singapore)*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2007.
The definitive version is available at https://doi.org/10.1109/ISIC.2007.4450950

22nd IEEE International Symposium on Intelligent Control
Part of IEEE Multi-conference on Systems and Control
Singapore, 1-3 October 2007

Near Optimal Neural Network-based Output Feedback Control of Affine Nonlinear Discrete-Time Systems

Qinmin Yang and S. Jagannathan

Abstract— **In this paper, a novel online reinforcement learning neural network (NN)-based optimal output feedback controller, referred to as adaptive critic controller, is proposed for affine nonlinear discrete-time systems, to deliver a desired tracking performance. The adaptive critic design consist of three entities, an observer to estimate the system states, an action network that produces optimal control input and a critic that evaluates the performance of the action network. The critic is termed adaptive as it adapts itself to output the optimal cost-to-go function which is based on the standard Bellman equation. By using the Lyapunov approach, the uniformly ultimate boundedness (UUB) of the estimation and tracking errors and weight estimates is demonstrated. The effectiveness of the controller is evaluated for the task of nanomanipulation in a simulation environment.**

I. INTRODUCTION

THERE are many ways of designing stable controllers for nonlinear systems. However, stability is a bare requirement. Most important consideration is optimality which normally is used to evaluate the performance of the system. In other words, a controller scheme should not only achieve the stability of the closed-loop system but also it has to keep the cost function as small as possible. Of the available methods, originated from Bellman's Principle of Optimality, dynamic programming (DP) has been extensively applied to generate optimal control inputs for nonlinear dynamic systems [1]-[2]. However, most of the methods are implemented either offline using iterative schemes or require the dynamics of the nonlinear systems to be known *a priori* [6]. Those requirements often make these methods inadequate on real-world systems.

To overcome the need for a system model or dynamics, reinforcement learning is originated from animal behavior research. Of the available reinforcement learning schemes, the temporal difference (TD) learning method [3]-[4] has found many applications in engineering. The advantage of using reinforcement learning in general is that it does not require the knowledge of the system dynamics. However an iterative approach is typically utilized making convergence a major issue for these schemes. Additionally, to obtain a satisfactory reinforcement signal for each action, the approach must visit every system state and apply the actions often enough [5], which in turn requires the system to be time-invariant, or stationary in the case of stochastic system.

On the other hand, in order to design suitable schemes for real-time applications, several appealing online neural controller designs methods were introduced in [6]-[9], referred to as forward dynamic programming (FDP) or adaptive critic designs (ACD). The central theme of this approach is to approximate the optimal control law and cost function by parametric structures, such as neural networks (NNs) by assuming the states are available for measurement. One of the main advantages of ACD is that they are trained over time using the feedback information, which in turn makes the approach appealing to most real-time systems.

On the other hand, an output feedback controller scheme is usually necessary when certain states of the plant are unavailable for measurement. The separation principle, which is normally employed for linear systems, does not hold for nonlinear systems [10]. In other words, a state observer, in general, does not guarantee the closed-loop system stability when it is used in conjunction with a stabilizing controller.

In this paper, we are considering NNs for the control of nonlinear discrete systems with quadratic-performance index. The entire system consists of three NNs: an observer NN to estimate the system states; an action NN to derive the optimal (or near optimal) control signal in order to track the desired system output; an adaptive critic NN to approximate the long-term cost function and to tune the action NN weights. Here the standard Bellman equation is considered as the long-term cost function and approximated by the critic NN. Lyapunov stability of the closed-loop system is demonstrated without using the separation principle and a numerical example is included.

II. BACKGROUND

In this paper, consider the following nonlinear affine system, given in the form

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_n(k+1) &= f(x(k)) + g(x(k))u(k) + d(k) \\ y(k) &= x_1(k) \end{aligned} \tag{1}$$

with the state $x(k) = [x_1(k), x_2(k), \cdots, x_n(k)]^T \in R^{nm}$ at time instant k and each $x_i(k) \in R^m$, $i = 1, \cdots, n$. $f(x(k)) \in R^m$ is a unknown nonlinear function vector, and $g(x(k)) \in R^{m \times m}$ is a diagonal matrix of unknown nonlinear functions, $u(k) \in R^m$ is the control input vector and $d(k) \in R^m$ is the unknown but

The authors are with the Department of Electrical & Computer Engineering, University of Missouri, Rolla, MO, 65401 USA (e-mail: qyy74@umr.edu). Research supported in part by NSF ECCS #0327877 and #0621924 grants.

bounded disturbance vector, whose bound is assumed to be a known constant, $\|d(k)\| \le d_m$. Here $\|\cdot\|$ stands for the Frobenius norm [15], which will be used through out this paper. It is also assumed that only the output vector $y(k) \in R^m$ is available at the k^{th} step.

Assumption 1: Let the diagonal matrix $g(x(k)) \in R^{m\times m}$ be a positive definite matrix for each $x(k) \in R^{nm}$, with $g_{\min} \in R^+$ and $g_{\max} \in R^+$ represent the minimum and maximum eigenvalues of $g(x(k))$ respectively.

Meanwhile, to introduce the issue of optimality into our design, the long-term cost function is defined as

$$\begin{aligned} J(k) &= J(x(k),u) = \sum_{i=t_0}^{\infty} \gamma^i r(k+i) \\ &= \sum_{i=t_0}^{\infty} \gamma^i [q(x(k+i)) + u^T(k+i)Ru(k+i)] \end{aligned} \tag{2}$$

where $J(k)$ stands for $J(x(k),u)$ for simplicity, and u is a control policy, R is a positive definite matrix and $q(x(k))$ is a positive definite function of the states, while γ $(0 \le \gamma \le 1)$ is the discount factor for the infinite-horizon problem. As observed from (2), the long-term cost function is the discounted sum of the immediate cost function or Lagrangian $r(k)$. Since some of the states are unavailable, we define it as

$$\begin{aligned} r(k) &= q(x(k)) + u^T(k)Ru(k) \\ &= (x_1(k) - x_{1d}(k))^T Q(x_1(k) - x_{1d}(k)) + u^T(k)Ru(k) \\ &= (y(k) - y_d(k))^T Q(y(k) - y_d(k)) + u^T(k)Ru(k) \end{aligned} \tag{3}$$

where Q is a positive definite matrix. In this paper, we are using a widely used standard quadratic cost function defined based on the output tracking error $e(k) = y(k) - y_d(k)$, which will be defined later in contrast with [9] and [11]. The immediate cost function $r(k)$ can be viewed as a system performance index for the current step.

The basic idea in adaptive critic or reinforcement learning design is to approximate the long-term cost function $J(k)$ (or its derivative, or both), and generate the control signal minimizing the cost. By using online learning, the function approximator will converge to the optimal cost function and the controller will approach optimality. As a matter of fact, for an optimal control law, which can be expressed as $u^*(k) = u^*(x(k))$, the optimal long-term cost function can be written alternatively as $J^*(k) = J^*(x(k), u^*(x(k))) = J^*(x(k))$, which is a function of the current state [6]. Next, one can state the following assumption.

Assumption 2: The optimal cost function $J^*(k)$ is finite and bounded over the compact set $S \subset R^m$ by J_m.

III. NN Observer Design

A. Observer Structure

For the system described by (1), we use the following NN-based state observer to estimate the actual state $x(k)$ as

$$\begin{aligned} &\hat{x}_1(k) = \hat{x}_2(k-1) \\ &\quad\vdots \\ &\hat{x}_n(k) = \hat{w}_o^T(k-1)\phi_o(v_o^T\hat{z}(k-1)) = \hat{w}_o^T(k-1)\phi_o\left(\hat{z}\left(k-1\right)\right) \end{aligned} \tag{4}$$

where $\hat{x}_i(k) \in R^m$, $i = 1,\cdots,n$ is the estimated value of $x_i(k) \in R^m$, and $\hat{z}(k-1) = \left[\hat{x}_1^T(k-1),\ldots,\hat{x}_n^T(k-1),u^T(k-1)\right]^T \in R^{(n+1)m}$ is the input vector to the NN observer at the k^{th} instant, $\hat{w}_o(k-1) \in R^{n_o\times m}$ and $v_o \in R^{(n+1)m\times n_o}$ denote the output and hidden layer weights of the NN observer, and n_o is the number of the hidden layer neurons. For simplicity purpose, the hidden layer activation function vector $\phi_o(v_o^T\hat{z}(k-1)) \in R^{n_o}$ is written as $\phi_o(\hat{z}(k-1))$. It is demonstrated in [16] that, if the hidden layer weights, v_o, is chosen initially at random and kept constant and n_o being sufficiently large, the NN approximation error can be made arbitrarily small since the activation function vector forms a basis.

B. Observer Error Dynamics

Define the state estimation error as

$$\tilde{x}_i(k) = \hat{x}_i(k) - x_i(k),\ i = 1,\ldots,n \tag{5}$$

where $\tilde{x}_i(k) \in R^m$, $i = 1,\ldots,n$ is the state estimation error. As a matter of fact, by comparing (1) and (4), one can find that the observer NN approximates the nonlinear function given by $f(x(k-1)) + g(x(k-1))u(k-1)$. Thus, ideally this nonlinear function can be expressed as

$$f(x(k-1)) + g(x(k-1))u(k-1) = w_o^T\phi_o(z(k-1)) + \varepsilon_o(z(k-1)) \tag{6}$$

where $w_o \in R^{n_o\times m}$ is the target observer NN weight matrix, $\varepsilon_o(z(k-1))$ is the NN approximation error, and the NN input is given by $z(k-1) = [x_1^T(k-1),\ldots,x_n^T(k-1),u^T(k-1)]^T \in R^{(n+1)m}$. Again, for convenience, the hidden layer activation function vector $\phi_o(v_o^T z(k-1)) \in R^{n_o}$ is written as $\phi_o(z(k-1))$.

Combining (4), (5) and (6), one obtains

$$\begin{aligned} \tilde{x}_n(k) &= \hat{x}_n(k) - x_n(k) \\ &= \hat{x}_n(k) - f(x(k-1)) - g(x(k-1))u(k-1) - d(k-1) \\ &= \tilde{w}_o^T(k-1)\phi_o(\hat{z}(k-1)) + w_o^T\phi_o(\tilde{z}(k-1)) - \varepsilon_o(z(k-1)) - d(k-1) \\ &= \xi_o(k-1) + d_o(k-1) \end{aligned} \tag{7}$$

where

$$\tilde{w}_o(k-1) = \hat{w}_o(k-1) - w_o \tag{8}$$

$$\xi_o(k-1) = \tilde{w}_o^T(k-1)\phi_o\left(\hat{z}\left(k-1\right)\right) \tag{9}$$

$$\tilde{z}(k-1) = \hat{z}(k-1) - z(k-1) \tag{10}$$

$$d_o(k-1) = w_o^T\phi_o(\tilde{z}(k-1)) - (\varepsilon_o(z(k-1)) + d(k-1)) \tag{11}$$

Therefore, the dynamics of the estimation error is obtained using (5) and (7) as

$$\begin{aligned} &\tilde{x}_1(k) = \tilde{x}_2(k-1) \\ &\quad\vdots \\ &\tilde{x}_n(k) = \xi_o(k-1) + d_o(k-1) \end{aligned} \tag{12}$$

IV. ONLINE LEARNING CONTROLLER DESIGN

The objective of our design is to present an online reinforcement learning NN controller for the system (1) such that a) all the signals in the closed-loop system remain uniformly ultimately bounded (*UUB)* [15]; b) the state $x_1(k)$ or the output $y(k)$ follows a desired trajectory $y_d(k) \in R^n$; and c) the long-term cost function (2) is minimized so that a near optimal control input can be generated. In this paper, the "online" means the NN weights are tuned "in real-time" by interacting with the plant, instead of an offline manner.

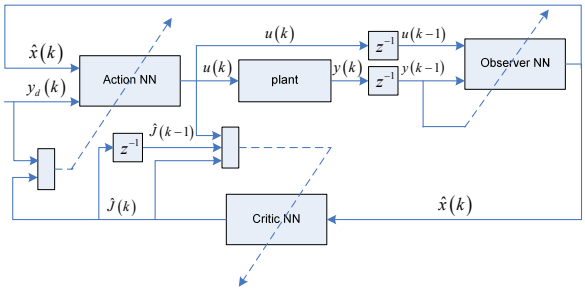

Fig. 1. Structure of neural online learning controller with output feedback.

The block diagram of the proposed controller is shown in Fig. 1. In our controller architecture, we consider all the NNs having a two-layer structure. Furthermore, in this paper, a novel tuning algorithm is proposed to make the NN weights robust so that PE condition is not needed, which will be discussed later. Next we present the controller design. Before we proceed, the following mild assumption is needed.

Assumption 3: The desired trajectory of the system output, $y_d(k)$ and its future values are bounded and available.

A. The Action NN Design

The output tracking error at instant *k* is first defined as

$$e_i(k) = x_i(k) - x_{id}(k) = x_i(k) - y_d(k+i-1),\ i = 1,...,n \quad (13)$$

Then future value of the tracking error using system dynamics from (1) can be rewritten as

$$e_n(k+1) = f(x(k)) + g(x(k))u(k) + d(k) - y_d(k+n) \quad (14)$$

The desired control signal can be given by

$$u_d(k) = g^{-1}(x(k))(-f(x(k)) + y_d(k+n) + l_1 e_n(k)) \quad (15)$$

where $l_1 \in R^{m\times m}$ is a design matrix selected such that the tracking error, $e_n(k)$, converges to zero.

Since both of $f(x(k))$ and $g(x(k))$ are unknown smooth nonlinear functions, the desired feedback control $u_d(k)$ cannot be implemented directly. Instead, in this paper, an action NN is employed to generate the control signal. Derived from (15) and considering Assumption 1 and 2, the desired control signal can be approximated as

$$u_d(k) = w_a^T \phi_a(v_a^T s(k)) + \varepsilon_a(s(k)) = w_a^T \phi_a(s(k)) + \varepsilon_a(s(k)) \quad (16)$$

where $s(k) = \left[\hat{x}^T(k), y_d^T(k)\right]^T \in R^{(n+1)\times m}$ is the input vector of the action NN. Note that the input includes the estimation results from the observer NN. As stated above, the action NN consists of two layers, and $w_a \in R^{n_a\times m}$ and $v_a \in R^{(n+1)m\times n_a}$ denote the desired weights of the output and hidden layer respectively with $\varepsilon_a(s(k))$ is the action NN approximation error, and n_a is the number of the neurons in the hidden layer. Since v_a is fixed, similarly, the hidden layer activation function vector $\phi_a(v_a^T s(k)) \in R^{n_a}$ is simply denoted as $\phi_a(s(k))$.

Considering the fact that the desired weights of the action NN are unknown, the actual NN weights have to be trained online and its output can be expressed as

$$u(k) = \hat{w}_a^T(k)\phi_a(v_a^T s(k)) = \hat{w}_a^T(k)\phi_a(s(k)) \quad (17)$$

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

Using the action NN output as the control signal, and substituting (16) and (17) into (14) yields

$$\begin{aligned} e_n(k+1) &= f(x(k)) + g(x(k))u(k) + d(k) - y_d(k+n) \\ &= l_1 e_n(k) + g(x(k))(\tilde{w}_a^T(k)\phi_a(s(k)) - \varepsilon_a(s(k))) + d(k) \\ &= l_1 e_n(k) + g(x(k))\zeta_a(k) + d_a(k) \end{aligned} \quad (18)$$

where

$$\tilde{w}_a(k) = \hat{w}_a(k) - w_a \quad (19)$$

$$\zeta_a(k) = \tilde{w}_a^T(k)\phi_a(s(k)) \quad (20)$$

$$d_a(k) = -g(x(k))\varepsilon_a(s(k)) + d(k) \quad (21)$$

Thus, the closed-loop dynamics in terms of output tracking error is expressed as

$$e_n(k+1) = l_1 e_n(k) + g(x(k))\zeta_a(k) + d_a(k) \quad (22)$$

Next the critic NN design is introduced.

B. The Critic NN Design

In this paper, a critic NN is employed to approximate the long-term cost function $J(k)$, which is unavailable at the k^{th} time instant in an online learning framework. First, the prediction error generated by the critic or the Bellman error [9] is defined as

$$e_c(k) = \gamma\hat{J}(k) - [\hat{J}(k-1) - r(k)] \quad (23)$$

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

$$\hat{J}(k) = \hat{w}_c^T(k)\phi_c(v_c^T\hat{x}(k)) = \hat{w}_c^T(k)\phi_c(\hat{x}(k)) \quad (24)$$

where $\hat{J}(k) \in R$ is the critic NN output which is designed as an approximation of $J(k)$. Taking the critic NN as a two-layer NN, $\hat{w}_c(k) \in R^{n_c\times 1}$ and $v_c \in R^{nm\times n_c}$ represent its actual weight matrix of the output and hidden layer respectively. The term n_c denotes the number of the neurons in the hidden layer. Originally, the system states $x(k) \in R^{nm}$ are selected as the critic NN input, similar to HDP. Since the states are unavailable, their estimated values are employed instead. The activation function vector of the hidden layer $\phi_c(v_c^T x(k)) \in R^{n_c}$ is denoted as $\phi_c(x(k))$ for simplicity. Provided with actual system states and enough hidden layer neurons, the optimal long-term cost function $J^*(k)$ can be approximated with arbitrarily small error $\varepsilon_c(k)$,

$$J^*(k) = w_c^T \phi_c(v_c^T x(k)) + \varepsilon_c(x(k)) = w_c^T \phi_c(x(k)) + \varepsilon_c(x(k)) \quad (25)$$

Similarly, the critic NN weight estimation error can be defined as

$$\tilde{w}_c(k) = \hat{w}_c(k) - w_c \quad (26)$$

where the approximation error is given by

$$\begin{aligned} \zeta_c(k) &= \hat{w}_c^T(k)\phi_c(\hat{x}(k)) - w_c^T \phi_c(x(k)) \\ &= \tilde{w}_c(k)\phi_c(\hat{x}(k)) + w_c^T \tilde{\phi}_c(x(k)) \end{aligned} \quad (27)$$

Thus, we obtain

$$\begin{aligned} e_c(k) &= \gamma \hat{J}(k) - \hat{J}(k-1) + r(k) \\ &= \gamma \zeta_c(k) + \gamma J^*(k) - \zeta_c(k-1) - J^*(k-1) \\ &\quad + r(k) - \varepsilon_c(k) + \varepsilon_c(k-1) \end{aligned} \quad (28)$$

Next we present the weight tuning algorithms for all NNs.

C. Weight Updating for the Observer NN

The observer NN weight update is driven by the state estimation error $\tilde{x}_1(k) = \hat{x}_1(k) - y(k)$, i.e.,

$$\hat{w}_o(k+1) = \hat{w}_o(k) - \alpha_o \phi_o(\hat{z}(k)) \left(\hat{w}_o^T(k)\phi_o(\hat{z}(k)) + l_2 \tilde{x}_1(k) \right)^T \quad (29)$$

where $l_2 \in R^{m \times m}$ is a design matrix, and $\alpha_o \in R^+$ is the adaptation gain for the NN observer.

D. Weight Updating for the Critic NN

Following the discussion from the last section, the objective function to be minimized by the critic NN can be defined as a quadratic function of tracking errors as

$$E_c(k) = \frac{1}{2} e_c^T(k) e_c(k) = \frac{1}{2} e_c^2(k) \quad (30)$$

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

$$\hat{w}_c(k+1) = \hat{w}_c(k) + \Delta \hat{w}_c(k) \quad (31)$$

where

$$\Delta \hat{w}_c(k) = \alpha_c \left[-\frac{\partial E_c(k)}{\partial \hat{w}_c(k)} \right] \quad (32)$$

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

Combining (23), (24), (30) with (32), the critic NN weight updating rule can be obtained by using the chain rule as

$$\begin{aligned} \Delta \hat{w}_c(k) &= -\alpha_c \frac{\partial E_c(k)}{\partial \hat{w}_c(k)} = -\alpha_c \frac{\partial E_c(k)}{\partial e_c(k)} \frac{\partial e_c(k)}{\partial \hat{J}(k)} \frac{\partial \hat{J}(k)}{\partial \hat{w}_c(k)} \\ &= -\alpha_c \gamma \phi_c(\hat{x}(k)) e_c(k) \end{aligned} \quad (33)$$

Thus, the critic NN weight updating algorithm is

$$\hat{w}_c(k+1) = \hat{w}_c(k) - \alpha_c \gamma \phi_c(\hat{x}(k)) (\gamma \hat{J}(k) + r(k) - \hat{J}(k-1)) \quad (34)$$

E. Weight Updating for the Action NN

The basis for adapting the action NN is to track the desired trajectory and to minimze the cost function. Therefore, the error for the action NN can be formed by using the functional estimation error $\zeta_a(k)$, and the error between the nominal desired long-term cost function $J_d(k) \in R$ and the critic signal $\hat{J}(k)$. Now we define the approximated cost function vector as $\bar{J}(k) = \left[\hat{J}(k) \quad \hat{J}(k) \quad \ldots \quad \hat{J}(k) \right]^T \in R^{m \times 1}$. Let

$$\begin{aligned} e_a(k) &= \sqrt{g(x(k))}\zeta_a(k) + \left(\sqrt{g(x(k))} \right)^{-1} (\bar{J}(k) - J_d(k)) \\ &= \sqrt{g(x(k))}\zeta_a(k) + \left(\sqrt{g(x(k))} \right)^{-1} \bar{J}(k) \end{aligned} \quad (35)$$

Given Assumption 1, we define $\sqrt{g(x(k))} \in R^{m \times m}$ as the principle square root of the diagonal positive definite matrix $g(x(k))$, i.e., $\sqrt{g(x(k))} \times \sqrt{g(x(k))} = g(x(k))$, and $\left(\sqrt{g(x(k))} \right)^T = \sqrt{g(x(k))}$ [11]. The desired long-term cost function $J_d(k)$ is nominally defined and is considered to be zero ("0"), which means as low as possible.

Hence, the action NN weights $\hat{w}_a(k)$ are tuned to minimize the error

$$E_a(k) = e_a^T(k) e_a(k) / 2 \quad (36)$$

Combining (18), (20), (22), (35), (36) and using the chain rule yields

$$\begin{aligned} \Delta \hat{w}_a(k) &= -\alpha_a \frac{\partial E_a(k)}{\partial \hat{w}_a(k)} = -\alpha_a \frac{\partial E_a(k)}{\partial e_a(k)} \frac{\partial e_a(k)}{\partial \zeta_a(k)} \frac{\partial \zeta_a(k)}{\partial \hat{w}_c(k)} \\ &= -\alpha_a \phi_a(s(k)) \left(e_n(k+1) - l_1 e_n(k) - d_a(k) + \bar{J}(k) \right)^T \end{aligned} \quad (37)$$

where $\alpha_a \in R^+$ is the adaptation gain of the action NN. However, $d_a(k)$ is typically unavailable, so as in the ideal case, we take it as zero. Further, we substitute $e_n(k+1)$ and $e_n(k)$ with $\hat{e}_n(k+1)$ and $\hat{e}_n(k)$ respectively, and obtain the weight updating algorithm for the action NN as following

$$\hat{w}_a(k+1) = \hat{w}_a(k) - \alpha_a \phi_a(s(k)) \left(\hat{e}_n(k+1) - l_1 \hat{e}_n(k) + \bar{J}(k) \right)^T \quad (38)$$

V. Main Theoretic Result

Assumption 4: Let w_o, w_a and w_c be the unknown output layer target weights for the observer, action and critic NNs respectively, and assume that they are upper bounded with

$$\|w_o\| \le w_{om}, \ \|w_a\| \le w_{am}, \text{ and } \|w_c\| \le w_{cm} \quad (39)$$

where $w_{om} \in R^+$, $w_{am} \in R^+$ and $w_{cm} \in R^+$ represent the bounds on the unknown target weights.

Fact 1: The activation functions for the action and critic NNs are bounded by known positive values, such that

$$\|\phi_o(k)\| \le \phi_{om}, \ \|\phi_a(k)\| \le \phi_{am}, \text{ and } \|\phi_c(k)\| \le \phi_{cm} \quad (40)$$

where $\phi_{om}, \phi_{am}, \phi_{cm} \in R^+$ is the upper bound for the activation functions. As a result, the term $\tilde{\phi}_o(k)$ in (10) is bounded by $\|\tilde{\phi}_o(k)\| \le 2\phi_{om}$. In our study, hyperbolic tangent sigmoid transfer function is used, which makes this assumption valid.

Assumption 5: The NN approximation errors $\varepsilon_o(z(k))$, $\varepsilon_a(s(k))$ and $\varepsilon_c(x(k))$ are bounded above over the compact set $S \subset R^m$ by ε_{om}, ε_{am} and ε_{cm} [16].

Fact 2: With the Assumption 1, 4 and Fact 1, the term $d_a(k)$ in (21) and $d_o(k)$ in (11) are bounded over the compact set $S \subset R^n$ by

$$\|d_a(k)\| \le d_{am} = g_{\max} \varepsilon_{am} + d_m \quad (41)$$

$$\left\|d_o(k)\right\| \le d_{om} = 2w_o^T \phi_{om} + \varepsilon_{om} + d_m \tag{42}$$

Combining Assumption 1, 3, and 4 and Facts 1, and 2, the main result is introduced in the following theorem.

Theorem 1: Consider the system (1) and let the Assumptions 1 through 4 hold with the disturbance bound d_m a known constant. Let system states be estimated by observer NN (4), the control input be provided by the action NN (17), with the critic NN (24) tuning the action NN weights. Further, let the weights of the observer and action NNs be tuned by (29), (34) and (38) respectively. Then the tracking error $e(k)$, and the NN weight estimates of the observer, action and critic NNs, $\hat{w}_o(k)$, $\hat{w}_a(k)$ and $\hat{w}_c(k)$ are *UUB*, with the bounds specifically given by (A.4) through (A.8) provided the controller design parameters are selected as

(a) $$0 < \alpha_c \gamma^2 \left\|\phi_c(k)\right\|^2 < 1 \tag{43}$$

(b) $$0 < \alpha_a \left\|\phi_a(k)\right\|^2 < g_{\min} / g_{\max}^2 \tag{44}$$

(c) $$0 < \alpha_c \left\|\phi_c(x(k))\right\|^2 < 1 \tag{45}$$

(d) $$0 < l_{1\max} < \sqrt{3}/3 \tag{46}$$

(e) $$\gamma > \sqrt{3}/3 \tag{47}$$

where α_o , α_a and α_c are NN adaptation gains, and γ is employed to define the cost function.

Proof: See Appendix.

Remark: The tracking errors can be made arbitrarily small through the selection of feedback gains.

VI. Simulation Results

To demonstrate the feasibility of the theoretic results, our design is evaluated on nanomanipulation system. Among the applications which require real-time control design, nanomanipulation [12] aims at handling nanometer size objects with nanometer precision. In our lab, manipulation of nano gold particle with diameter of 30 nm has been performed by an Atomic Force Microscope (AFM).

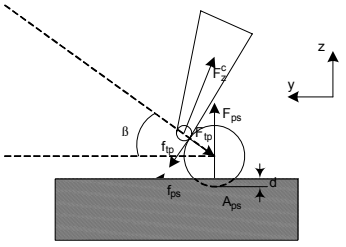

Fig. 2. Forces between AFM tip, nano particle and stage during pushing process.

The nanomanipulation system used in this paper is identical to the one in [13]. The simplified geometrical relationship between AFM tip, nano sphere and substrate (stage) is shown in Fig. 2. Briefly, the objective of nanomanipulation is to drive the stage of AFM towards the tip, which will in turn push nano particles along a desired path. The equations governing the system is shown as

$$\begin{aligned} \ddot{x}_s / w_x^2 + \dot{x}_s / w_x Q_x + x_s + \cos\theta f_{ps}(z_s, z_{sub}) &= \tau_x \\ \ddot{y}_s / w_y^2 + \dot{y}_s / w_y Q_y + y_s + \sin\theta f_{ps}(z_s, z_{sub}) &= \tau_y \\ \ddot{z}_s / w_z^2 + \dot{z}_s / w_z Q_z + z_s + F_{ps}(z_s, z_{sub}) &= \tau_z + A_{ps} \end{aligned} \tag{48}$$

where (x_s, y_s, z_s) is the position of the stage on x, y, and z axis respectively. (w_x, w_y, w_z) is the resonant frequency and (Q_x, Q_y, Q_z) is the amplification factor for the stage. (τ_x, τ_y, τ_z) is the stage driving force or the control input signal in this paper. θ is the angle between y axis and the pushing direction, and z_{sub} is the substrate surface height displacement, which is considered zero in this paper for simplification. Now f_{ps} is the friction force and F_{ps} is the attractive/repulsive interaction force between particle and substrate, which is a complex function of the pushing environment. For more details, please refer to [13] and [14]. Equation (48) when discretized can be expressed as (1) indicating that the manipulation system can be viewed as a second order nonlinear system. Meanwhile, only the position of the stage is measurable in typical AFM system. An observer is desirable to estimate unavailable states. The design parameters of the controller are set as follows:

TABLE 1.PARAMETERS USED FOR NANOMANIPULATION

Parameter	R,Q	l_1	l_2	$\alpha_o,\alpha_c,\alpha_a$	n_o	n_a,n_c
Value	0.1	0.1	0.4	$8\times e^{-6}$	40	20
Parameter	w_x	w_y	w_z	Q_x,Q_y,Q_z	θ	γ
Value	1570 rad/s	1570 rad/s	117.6 rad/s	20	30°	0.8

The simulation time step is 1×10^{-5}. The objective is to push particle with a constant speed. A proper force on the nano particle will indicate the particle being pushed by the tip, which could be observed by the stage movement in z axis.

Our online learning controller is applied on the system with the results shown in Fig. 3.

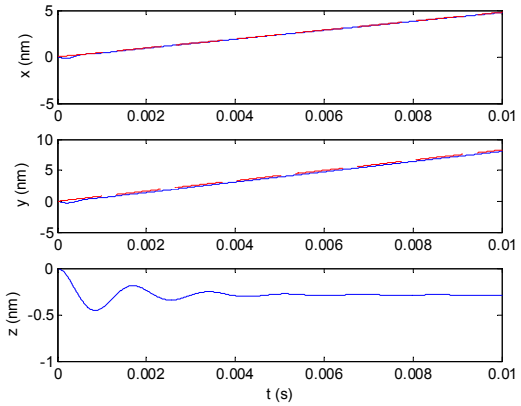

Fig. 3. Simulation results of the online learning output feedback controller on nanomanipulation system. Solid line: trajectories of the actual stage movement whereas dashed line indicates target trajectory.

From the figure, we can find that the proposed scheme is able

to achieve a good performance for complex tasks such as nanomanipulation. Although there are transients, mainly due to the training of the observer, the tip of the AFM gets stabilized soon and moves along a desired trajectory.

VII. CONCLUSION

Output tracking control of a general class of affine nonlinear discrete-time systems is formulated. A novel online reinforcement learning-based output feedback controller is designed to deliver a desired performance under bounded disturbance, and through an optimization of the quadratic long-term cost function. Unlike many applications where the controller is trained offline or iteratively, the control input is updated in an online fashion. Furthermore, to guarantee the stability of the closed-loop system, the UUB of the tracking errors and NN weight estimates is ensured by using standard Lyapunov analysis in the presence of bounded disturbances and approximation errors. The feasibility of our method is also strengthened through the simulation results on a nanomanipulation system.

APPENDIX

Proof of Theorem 1: Define the Lyapunov candidate as

$$\begin{aligned} L(k) = {} & \frac{\gamma_1}{2}\sum_{i=1}^{n}\left\|\tilde{x}_i(k-1)\right\|^2 + \frac{\gamma_2}{2}\sum_{i=1}^{n}\left\|\tilde{x}_i(k)\right\|^2 + \frac{\gamma_3}{3}\sum_{i=1}^{n}\left\|e_i(k-1)\right\|^2 \\ & + \gamma_4\sum_{i=1}^{n}\left\|e_i(k)\right\|^2 \Big/ 3 + \gamma_5 tr(\tilde{W}_o^T(k-1)\tilde{W}_o(k-1))/\alpha_o \\ & + \frac{\gamma_6}{\alpha_o} tr(\tilde{W}_o^T(k)\tilde{W}_o(k)) + \frac{\gamma_7}{\alpha_a} tr(\tilde{W}_a^T(k)\tilde{W}_a(k)) \\ & + \gamma_8 tr(\tilde{W}_c^T(k)\tilde{W}_c(k))/\alpha_c + \gamma_9\left\|\zeta_c(k-1)\right\|^2 \\ & + \gamma_{10}\left\|\zeta_c(k-1)\right\|^2 + \gamma_{11}\left\|e_1(k-1)\right\|^2 \end{aligned} \tag{A.1}$$

where $\gamma_i \in R^+$, $i = 1,...,11$ are design parameters. Hence, the first difference of the Lyapunov function is given by

$$\begin{aligned} \Delta L = {} & -\left(\gamma_6 - \gamma_2 - 2\gamma_7'\right)\left\|\zeta_o(k)\right\|^2 - \left(\gamma_5 - \gamma_1 - 2\gamma_7' l_{1\max}^2\right)\left\|\zeta_o(k-1)\right\|^2 \\ & - \left(\gamma_7' g_{\min} - \gamma_4 g_{\max}^2 - \gamma_3 g_{\max}^2 - \gamma_9\right)\left\|\zeta_a(k)\right\|^2 \\ & - \left(\gamma_9 - 2\gamma_8 R_{\max}/3\right)\left\|\zeta_a(k-1)\right\|^2 - \left(\gamma_8\gamma^2 - \gamma_7' - \gamma_{10}\right)\zeta_c^2(k) \\ & - \left(\gamma_{10} - \gamma_8/3\right)\zeta_c^2(k-1) - \left(\gamma_2/2 - 2\gamma_6 l_{2\max}^2\right)\left\|\tilde{x}_1(k)\right\|^2 \\ & - \left(\gamma_1/2 - 2\gamma_5 l_{2\max}^2\right)\left\|\tilde{x}_1(k-1)\right\|^2 - \left(\gamma_3/3 - \gamma_{11}\right)\left\|e_1(k)\right\|^2 \\ & - \left(\gamma_{11} - \gamma_8 Q_{\max}/3\right)\left\|e_1(k-1)\right\|^2 - \left(\gamma_4/3 - \gamma_4 l_{1\max}^2 - \gamma_3 l_{1\max}^2\right)\left\|e_n(k)\right\|^2 \\ & - \gamma_8\left(1 - \alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k) - \gamma_7'\left(g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2\right) \\ & \times\left\|\zeta_a(k) + \frac{\left(I - \alpha_a\left\|\phi_a(s(k))\right\|^2 g(x(k))\right)\beta(k)}{g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2}\right\|^2 \\ & - \gamma_6\left(1 - \alpha_o\left\|\phi_o(k)\right\|^2\right)\left\|\zeta_o(k) + l_2\tilde{x}_1(k) + W_o^T\phi_o(k)\right\|^2 \\ & - \gamma_5\left(1 - \alpha_o\left\|\phi_o(k-1)\right\|^2\right)\left\|\zeta_o(k-1) + l_2\tilde{x}_1(k-1) + W_o^T\phi_o(k-1)\right\|^2 \\ & + D_M^2 \end{aligned} \tag{A.2}$$

where

$$\begin{aligned} D_M^2 = {} & (2\gamma_5 + 2\gamma_6)w_{om}^2\phi_{om}^2 + 2\gamma_8 R_{\max} w_{am}^2\phi_{am}^2/3 + 2\gamma_7' w_{cm}^2\phi_{cm}^2 \\ & + \gamma_8(2\gamma^2 + 1)J_m^2/3 + (\gamma_1 + \gamma_2 + 2\gamma_7' + 2\gamma_7' l_{1\max}^2)d_{om}^2 \\ & + (\gamma_3 + \gamma_4 2\gamma_7')d_{am}^2 \end{aligned} \tag{A.3}$$

Referring to the standard Lyapunov analysis [15], equation (A.2) and (A.3) implies that $\Delta L \le 0$ as long as the conditions (43) – (47) are satisfied and following holds

$$\left\|\tilde{x}_1(k)\right\| \ge \sqrt{2}D_M \Big/ \sqrt{\gamma_2 - 4\gamma_6 l_{2\max}^2} \tag{A.4}$$

or

$$\left\|e_n(k)\right\| \ge \sqrt{3}D_M \Big/ \sqrt{\gamma_4 - 3(\gamma_3 + \gamma_4) l_{1\max}^2} \tag{A.5}$$

or

$$\left\|\zeta_o(k)\right\| \ge D_M \Big/ \sqrt{\gamma_6 - \gamma_2 - 2\gamma_7'} \tag{A.6}$$

or

$$\left\|\zeta_2(k)\right\| \ge D_M \Big/ \sqrt{\gamma_7' g_{\min} - (\gamma_3 + \gamma_4) g_{\max}^2 - \gamma_9} \tag{A.7}$$

or

$$\left\|\zeta_c(k)\right\| \le D_M \Big/ \sqrt{\gamma_8\gamma^2 - \gamma_7' - \gamma_{10}} \tag{A.8}$$

According to the standard Lyapunov extension theorem [15], the analysis above demonstrates that the tracking error $\left\|e(k)\right\|$ and the weights of the estimation errors are UUB. Further, the boundedness of $\left\|\zeta_a(k)\right\|$ and $\left\|\zeta_c(k)\right\|$ implies that the weight estimations $\left\|\hat{w}_a(k)\right\|$ and $\left\|\hat{w}_c(k)\right\|$ are also bounded.

REFERENCES

[1] R. Bellman and S. Dreyfus, Applied Dynamic Programming, Princeton, NJ: Princeton Univ. Press, 1962.

[2] D. Kirk, Optimal Control Theory: An Introduction, Englewood Cliffs, NJ: Prentice-Hall, 1970.

[3] C. J. C. H. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, pp. 257–277, 1992.

[4] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction", the MIT Press, Cambridge, MA, 1998.

[5] G. Boone, "Efficient reinforcement learning: Model-based acrobot control," in Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 229 - 234, Albuquerque, NM, 1997.

[6] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, etc., Handbook of Learning and Approximate Dynamic Programming, Wiley-IEEE Press, 2004.

[7] D. Prokhorov and D. Wunsch, "Adaptive critic designs", IEEE Trans. Neural Networks, Vol. 8, No.5, p.997-1007, 1997.

[8] P. J. Werbos, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," IEEE Transactions on Systems, Man, and Cybernetics 17, pp. 7–20, 1987.

[9] J. Si and Y. T. Wang, "On-line learning control by association and reinforcement," IEEE Trans. Neural Networks, vol. 12, no. 2, pp. 264–276,, 2001.

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

[11] P. He and S. Jagannathan, "Reinforcement learning-based output feedback control of nonlinear systems with input constraints", IEEE Trans. Syst., Man, Cybern., vol. 35, pp. 150–154, 2005.

[12] M. Sitti, "Survey of nanomanipulation systems", Proc. of the IEEE Conf. on Nanotechnology, pp.75–80, 2001.

[13] Qinmin Yang and S. Jagannathan, "Atomic force microscope-based nanomanipulation with drift compensation", Int. J. Nanotechnology, Vol. 3, No. 4, pp. 527-544, 2006.

[14] M. Sitti and H. Hashimoto, "Controlled Pushing of Nanoparticles: Modeling and Experiments", IEEE/ASME Trans. Mechatronics, 2000.

[15] S. Jagannathan, "Neural Network Control of Nonlinear Discrete-time Systems," Taylor & Francis, PA, 2006.

[16] B. Igelnik and Y. H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," IEEE Trans. Neural Networks, vol. 6, no. 6, pp. 1320 – 1329, 1995.