01 Jan 2007

# Online Reinforcement Learning Neural Network Controller Design for Nanomanipulation

Qinmin Yang

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

## Recommended Citation

Q. Yang and J. Sarangapani, "Online Reinforcement Learning Neural Network Controller Design for Nanomanipulation," *Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (2007, Honolulu, HI)*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2007.
The definitive version is available at https://doi.org/10.1109/ADPRL.2007.368192

**Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007)**

# Online Reinforcement Learning Neural Network Controller Design for Nanomanipulation

Qinmin Yang and S. Jagannathan

*Abstract*— **In this paper, a novel reinforcement learning neural network (NN)-based controller, referred to adaptive critic controller, is proposed for affine nonlinear discrete-time systems with applications to nanomanipulation. In the online NN reinforcement learning method, one NN is designated as the critic NN, which approximates the long-term cost function by assuming that the states of the nonlinear systems is available for measurement. An action NN is employed to derive an optimal control signal to track a desired system trajectory while minimizing the cost function. Online updating weight tuning schemes for these two NNs are also derived. By using the Lyapunov approach, the uniformly ultimate boundedness (UUB) of the tracking error and weight estimates is shown. Nanomanipulation implies manipulating objects with nanometer size. It takes several hours to perform a simple task in the nanoscale world. To accomplish the task automatically the proposed online learning control design is evaluated for the task of nanomanipulation and verified in the simulation environment.**

*Index Terms* — **Neural network, reinforcement learning, on-line learning, dynamic programming, Lyapunov method, nanomanipulation.**

## I. INTRODUCTION

Dynamic programming (DP) has been extensively applied [1] for the optimal control of nonlinear dynamic systems, However, one of the drawbacks of DP is the computation cost with the dimension of the nonlinear system, which is referred to as the “curse of dimensionality”. Therefore, adaptive or approximation methods for DP (e.g. see [2]) have been developed recently although most of them [3] are implemented either by offline using iterative schemes or require the dynamics of the nonlinear systems to be known *a priori*. Unfortunately, these requirements are often not practical for real-world systems, since the exact model of the nonlinear system is usually not available. Additionally, stability of the closed-loop system using adaptive DP-based methods are not discussed which limited its applicability for control applications until now.

On the other hand, reinforcement learning is originated from animal behavior research and its interactions with the environment. Differing from the traditional supervised learning in neural network (NN), there is no desired behavior or training examples employed within reinforcement learning schemes. Nevertheless, it is common to apply reinforcement learning for optimal controller design, since the cost function can be directly seen as a form of reinforcement signal. Of the available reinforcement learning schemes, the temporal difference (TD) learning method [4]-[5] has found many applications in the engineering area. The advantage of reinforcement learning in general is that it does not require the knowledge of the system dynamics even though an iterative approach is typically utilized. To obtain a satisfactory reinforcement signal for each action, the approach must visit each system state and apply each action often enough [7], and requires the system to be time-invariant, or stationary in the case of stochastic system.

To overcome the iterative offline methodology for real-time applications, several appealing online neural controller designs methods were introduced in [3], [8]-[9]. They are also referred to as forward dynamic programming (FDP) or adaptive critic designs (ACD). The central theme of this approach is that the optimal control law and cost function are approximated by parametric structures, such as neural networks (NNs), which are trained over time along with the feedback information. In other words, in ACD methods, instead of finding the exact minimum, the ACDs try to approximate the Bellman equation: $J\left(x(k)\right)=\min_{u(k)}\left\{J\left(x(k+1)\right)+U\left(x(k),x(k+1)\right)\right\}$, where $x(k)$ is the state and $u(k)=u(x(k))$ is a specific control law at time step $k$, the *strategic* utility function $J(x(k),u)=J(x(k))$ represents the cost or performance measure associated with going from $k$ to final step $N$, $U(x(k),x(k+1))$ is the utility function denoting the cost incurred in going from $x(k)$ to $x(k+1)$ using control $u(k)$, and $J(k+1)$ is the cost or performance measure associated in going from state $k+1$ to the final step $N$. In the ACD literature, NNs are widely used for approximation.

In [6], a new NN learning algorithm based on gradient descent rule is introduced and the approach is evaluated on three examples. However, no proof of the convergence or stability of the system was presented. By contrast, Lyapunov analysis was derived in [10] and [11]. The approach presented in [11] is specific to robotic systems whose dynamics are introduced in continuous-time. On the other hand, [6] and [10] employ simplified binary reward or cost function which is a variant of the standard Bellman equation. To the best of our knowledge, there is no published work considering the convergence proof of the closed-loop system with standard Bellman equation.

In this paper, we are considering NNs for the control of nonlinear discrete systems with quadratic-performance index as the cost function. The whole system consists of two NNs: an action NN to derive the optimal (or near optimal) control signal to track not only the desired system output but also to minimize

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 ECS #0327877 and ECS#0621924 grants.

the long-term cost function; an adaptive critic NN to approximate the long-term cost function $J(x(k))$ and to tune the action NN weights. Closed-loop stability is shown.

Nanomanipulation [12] aims at manipulating and handling nanometer size objects and structures with nanometer precision. It is also the first and critical step for any complex functional nano devices. Typically, assembly of small nano structures built by nanomanipulation today consists of ten to twenty particles, and may take an experienced user a whole day to construct.

A significant amount of work on modeling interactive forces during manipulation was introduced in [13]-[14]. Based on that model, real-time controllers can be designed to automate nanomanipulation. On the other hand, due to extremely complex environmental conditions during nanomanipulation tasks, any iteration based optimal controllers will fail to obtain satisfactory result. In other words, for every single manipulation attempt, the environmental conditions or the system dynamics is different from the other, which means the learning process obtained in current attempt/iteration can not be used directly during the next iteration. Thus, in this paper, the online learning controller design is implemented and evaluated on the task of nanomanipulation and simulation results show its effectiveness.

## II. BACKGROUND

### A. Optimal Control

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

$$\begin{aligned} x(k+1) &= f_0(x(k),u(k)) \\ &= f(x(k)) + g(x(k))u(k) + d(k) \end{aligned} \tag{1}$$

with the state $x(k) = [x_1(k), x_2(k), \cdots, x_n(k)]^T \in R^n$ at time instant $k$. $f(x(k)) \in R^n$ is a unknown nonlinear function vector, and $g(x(k)) \in R^{n\times n}$ is a matrix of unknown nonlinear functions, $u(k) \in R^n$ is the control input vector and $d(k) \in R^n$ is the unknown but bounded disturbance vector, whose bound is assumed to be a known constant, $\|d(k)\| \le d_m$. Here $\|\bullet\|$ stands for the Frobenius norm [17], which will be used through out this paper. It is also assumed that the state vector $x(k)$ is available at the kth step.

***Assumption 1***: Let the diagonal matrix $g(x(k)) \in R^{n\times n}$ be a positive definite matrix for each $x(k) \in R^n$, with $g_{\min} \in R^+$ and $g_{\max} \in R^+$ representing the minimum and maximum eigenvalues of the matrix $g(x(k))$ respectively, such that $0 < g_{\min} \le g_{\max}$.

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, $u$ is a given control policy, $R$ is a positive definite matrix and $\gamma$ $(0 \le \gamma \le 1)$ 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 expressed as

$$\begin{aligned} r(k) &= q(x(k)) + u^T(k)Ru(k) \\ &= (x(k) - x_d(k))^T Q(x(k) - x_d(k)) + u^T(k)Ru(k) \end{aligned} \tag{3}$$

where $Q$ is a positive definite matrix. In this paper, we are using a widely applied standard quadratic cost function defined based on the tracking error $e(k)$, which will be defined later in contrast with [6] and [10]. The immediate cost function $r(k)$ can be viewed as the system performance index for the current step.

The basic idea in the adaptive critic or reinforcement learning design is to approximate the long-term cost function J (or its derivative, or both), and generate the control signal minimizing the cost. By using learning through an algorithm, the online approximator will converge to the optimal cost function and the controller will converge to the optimal controller correspondingly. 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 just a function of the current state [16]. Next, one can state the following assumption.

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

Next it will be shown that nanomanipulation will be expressed as a nonlinear discrete-time system (1).

### B. Nanomanipulation

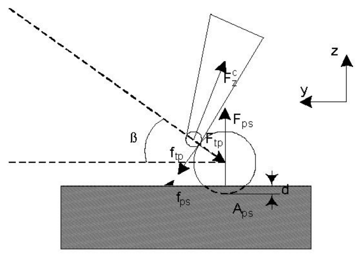

Fig. 1. Geometry and the interacting forces between AFM tip, nano particle and stage during pushing process.

Nowadays, assemblies of small nano structures built by nanomanipulation are typically realized by using an Atomic Force Microscope (AFM) as the manipulator. Initially used as the imaging tool, the tip of AFM is also utilized as manipulation end effector.

The simplified geometrical relationship between AFM tip, nano sphere and substrate (stage) is shown in Fig. 1. Briefly speaking, the objective of nanomanipulation is to drive the tip of AFM to push nano particles along a desired track. An alternative way is to drive the stage instead of the tip to accomplish the push task. In our experiment, the latter approach is selected.

The model development and analysis involves the adhesion forces between AFM tip, substrate and nano particle to be pushed. In the nano world, gravitational forces are relatively very small and, therefore, are neglected. The main components of the adhesion forces are van der Waals, capillary, and electrostatic forces [14].

After taking all those adhesion and friction forces into consideration, a satisfactory model is built in [13] and [14], which is adopted in this paper. Since we are driving the stage to accomplish the task, the equation governing the system is

$$\begin{aligned}
&\frac{1}{w_x^2}\ddot{x}_s+\frac{1}{w_xQ_x}\dot{x}_s+x_s+\cos\theta f_{ps}(z_s,z_{sub})=\tau_x\\
&\frac{1}{w_y^2}\ddot{y}_s+\frac{1}{w_yQ_y}\dot{y}_s+y_s+\sin\theta f_{ps}(z_s,z_{sub})=\tau_y\\
&\frac{1}{w_z^2}\ddot{z}_s+\frac{1}{w_zQ_z}\dot{z}_s+z_s+F_{ps}(z_s,z_{sub})=\tau_z+A_{ps}
\end{aligned}\tag{4}$$

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. $(\tau_x, \tau_y, \tau_z)$ is the stage driving force which is seen as the control input signal. The term $\theta$ is the angle between y axis and the pushing direction, and $z_{sub}$ is the substrate surface height displacement, which is simulated to be a sinusoid function 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 (4) represents the manipulation system which can be viewed as a nonlinear system of second order.

To fulfill Assumption 1, we define the tracking error as

$$e_s=\begin{bmatrix}x_s\\y_s\\z_s\end{bmatrix}-\begin{bmatrix}x_d\\y_d\\z_d\end{bmatrix}\tag{5}$$

where $(x_d, y_d, z_d)$ is the desired movement of the stage. Based on that, filtered tracking error can be defined as $s=\dot{e}_s+\Lambda e_s$, with $\Lambda$ a positive definite design parameter matrix. Common usage is to select $\Lambda$ diagonal with large positive entries. Therefore, the system dynamics can be rewritten in term of the filtered tracking error as follows

$$\begin{aligned}
\dot{s}&=\ddot{e}_s+\Lambda\dot{e}_s\\
&=\begin{bmatrix}\ddot{x}_s\\\ddot{y}_s\\\ddot{z}_s\end{bmatrix}-\begin{bmatrix}\ddot{x}_d\\\ddot{y}_d\\\ddot{z}_d\end{bmatrix}+\Lambda\begin{bmatrix}\dot{x}_s\\\dot{y}_s\\\dot{z}_s\end{bmatrix}-\Lambda\begin{bmatrix}\dot{x}_d\\\dot{y}_d\\\dot{z}_d\end{bmatrix}\\
&=\begin{bmatrix}\left(\Lambda-\frac{w_x}{Q_x}\right)\dot{x}_s-w_x^2x_s-\ddot{x}_d-\Lambda\dot{x}_d-w_x^2\cos\theta f_{ps}\\\left(\Lambda-\frac{w_y}{Q_y}\right)\dot{y}_s-w_y^2y_s-\ddot{y}_d-\Lambda\dot{y}_d-w_y^2\sin\theta f_{ps}\\\left(\Lambda-\frac{w_z}{Q_z}\right)\dot{z}_s-w_z^2z_s-\ddot{z}_d-\Lambda\dot{z}_d-w_z^2(F_{ps}-A_{ps})\end{bmatrix}+\begin{bmatrix}w_x^2&0&0\\0&w_y^2&0\\0&0&w_z^2\end{bmatrix}\begin{bmatrix}\tau_x\\\tau_y\\\tau_z\end{bmatrix}\\
&=f_s(s)+w\cdot\tau
\end{aligned}\tag{6}$$

As long as the controller guarantees that the filtered error $s$ is bounded, the tracking error $e_s$ is bounded. In order to apply our nonlinear discrete-time controller, the system dynamics (6) need to be discretized to obtain an affine nonlinear discrete-time system [17] which is given by

$$s(k+1)=T\left(F_s(s(k),s(k-1),\ldots)+w\cdot\tau\right)+s(k)\tag{7}$$

where T is the sampling time and $F_s$ is the corresponding nonlinearity in discrete. By rearranging (7), one can get an affine nonlinear discrete-time system (1), with the filtered tracking error as the system state.

## III. Online reinforcement learning controller design

For the purpose of this paper, our objective is to design an online reinforcement learning NN controller for the system (1) such that 1) all the signals in the closed-loop system remain *UUB*; 2) the state $x(k)$ follows a desired trajectory $x_d(k)\in R^n$; and 3) the long-term cost function (2) is minimized so that a near optimal control input can be generated. Here, the "online" means the learning of the controller takes place "in real-time" by interacting with the plant, instead of in an offline manner.

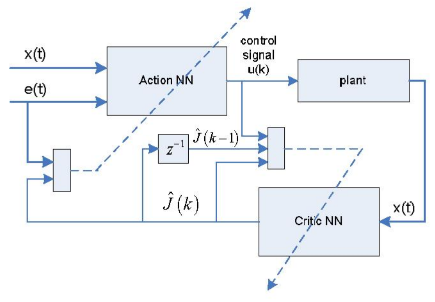

Fig. 2. Online neural dynamic programming based controller structure.

The block diagram of the proposed controller is shown in Fig. 2, where the action NN is providing a near optimal control signal to the plant while the critic NN approximates the long-term cost function. The learning of the two NNs is performed online without any offline learning phase.

In our controller architecture, we consider the action and the critic NN having two layers. The output of the NN can be given by $Y=W^T\phi(V^TX)$, where $V$ and $W$ are the hidden layer and output layer weights respectively. The number of hidden layer nodes is denoted as $N_2$.

A general function $f(x) \in C^{N_3}(S)$ can be written as

$$f(x) = W^T \phi(V^T x) + \varepsilon(x) \tag{8}$$

with $\varepsilon(x)$ a NN functional reconstruction error vector. In our design, $V$ is initially selected at random and held fixed during entire learning process. It is demonstrated in [15] that if the hidden layer weights, $V$, are chosen initially at random and kept constant and if $N_2$ is sufficiently large, the NN approximation error $\varepsilon(x)$ can be made arbitrarily small since the activation function vector forms a basis.

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 states, $x_d(k)$, is bounded over the compact subset of $R^n$. For our nanomanipulation system, the desired value is zero.

## *A. The Action NN Design*

The tracking error at instant *k* is defined as

$$e(k) = x(k) - x_d(k) \tag{9}$$

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

$$e(k+1) = f(x(k)) + g(x(k))u(k) + d(k) - x_d(k+1) \tag{10}$$

To eliminate the tracking error, a desired control law is given by

$$u_d(k) = g^{-1}(x(k))(-f(x(k) + x_d(k+1) + l_1 e(k)) \tag{11}$$

where $l_1 \in R^{n \times n}$ is a design matrix selected such that the tracking error, $e(k)$, is converging 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, an action NN is employed to generate the control signal. From (11) and considering Assumption 1 and 2, the desired control signal can be approximated as

$$u_d(k) = w_a^T \phi_a(v_a^T s(k)) + \varepsilon_a(s(k)) = w_a^T \phi_a(s(k)) + \varepsilon_a(s(k)) \tag{12}$$

where $s(k) = [x^T(k), e^T(k)]^T \in R^{2n}$ is the action NN input vector. The action NN consists of two layers, and $w_a \in R^{n_a \times n}$ and $v_a \in R^{2n \times n_a}$ 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 neurons in the hidden layer. Since $v_a$ is fixed, for simplicity purpose, the hidden layer activation function vector $\phi_a(v_a^T s(k)) \in R^{n_2}$ is 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 actual output can be expressed as

$$u(k) = \hat{w}_a^T(k)\phi_a(v_a^T s(k)) = \hat{w}_a^T(k)\phi_a(s(k)) \tag{13}$$

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

Using the action NN output as the control signal, and substituting (12) and (13) into (10) yields

$$\begin{aligned} e(k+1) &= f(x(k)) + g(x(k))u(k) + d(k) - x_d(k+1) \\ &= l_1 e(k) + g(x(k))(u(k) - u_d(k)) + d(k) \\ &= l_1 e(k) + g(x(k))\left(\tilde{w}_a^T(k)\phi_a(s(k)) - \varepsilon_a(s(k))\right) + d(k) \\ &= l_1 e(k) + g(x(k))\zeta_a(k) + d_a(k) \end{aligned} \tag{14}$$

where

$$\tilde{w}_a(k) = \hat{w}_a(k) - w_a \tag{15}$$

$$\zeta_a(k) = \tilde{w}_a^T(k)\phi_a(s(k)) \tag{16}$$

$$d_a(k) = -g(x(k))\varepsilon_a(s(k)) + d(k) \tag{17}$$

Thus, the closed-loop tracking error dynamics is expressed as

$$e(k+1) = l_1 e(k) + g(x(k))\zeta_a(k) + d_a(k) \tag{18}$$

Next the critic NN design is introduced.

## *B. The Critic NN Design*

As stated above, a near optimal controller should be able to stabilize the closed-loop system by minimizing the cost function. In this paper, a critic NN is employed to approximate the long-term cost function $J(k)$. Since the actual $J(k)$ is unavailable for us at the $k^{\text{th}}$ time instant in an online learning framework, the critic NN is tuned online in order to converge to the actual $J(k)$.

First, the prediction error generated by the critic or the Bellman error [6] is defined as

$$e_c(k) = \gamma \hat{J}(k) - [\hat{J}(k-1) - r(k)] \tag{19}$$

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

$$\hat{J}(k) = \hat{w}_c^T(k)\phi_c(v_c^T x(k)) = \hat{w}_c^T(k)\phi_c(x(k)) \tag{20}$$

where $\hat{J}(k) \in R$ is the critic NN output which is an approximation of $J(k)$. In our design, the critic NN is also a two-layer NN, while $\hat{w}_c(k) \in R^{n_c \times 1}$ and $v_c \in R^{n \times n_c}$ 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. Similar to HDP, the system states $x(k) \in R^n$ are selected as the critic NN input. 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 that enough number of the neurons in the hidden layer, the optimal long-term cost function $J^*(k)$ can be approximated by the critic NN with arbitrarily small approximation error $\varepsilon_c$ (k),

$$J^*(k) = w_c^T \phi_c(v_c^T x(k)) + \varepsilon_c(x(k)) = w_c^T \phi_c(x(k)) + \varepsilon_c(x(k)) \tag{21}$$

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

$$\tilde{w}_c(k) = \hat{w}_c(k) - w_c \tag{22}$$

where the approximation error is given by

$$\zeta_c(k) = \tilde{w}_c^T(k)\phi_c(x(k)) \tag{23}$$

Thus, we

$$\begin{aligned} e_c(k) &= \gamma\hat{J}(k) - \hat{J}(k-1) + r(k) \\ &= \gamma\zeta_c(k) + \gamma J^*(k) - \zeta_c(k-1) - J^*(k-1) \\ &\quad + r(k) - \varepsilon_c(k) + \varepsilon_c(k-1) \end{aligned} \tag{24}$$

Next we discuss the weight tuning algorithms for critic and action NNs.

## C. 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

$$E_c(k) = \frac{1}{2}e_c^T(k)e_c(k) = \frac{1}{2}e_c^2(k) \tag{25}$$

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

$$\hat{w}_c(k+1) = \hat{w}_c(k) + \Delta\hat{w}_c(k) \tag{26}$$

where

$$\Delta\hat{w}_c(k) = \alpha_c\left[-\frac{\partial E_c(k)}{\partial\hat{w}_c(k)}\right] \tag{27}$$

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

Combining (19), (20), (25) with (27), the critic NN weight updating rule can be obtained by using the chain rule as

$$\begin{aligned} \Delta\hat{w}_c(k) &= -\alpha_c\frac{\partial E_c(k)}{\partial\hat{w}_c(k)} = -\alpha_c\frac{\partial E_c(k)}{\partial e_c(k)}\frac{\partial e_c(k)}{\partial\hat{J}(k)}\frac{\partial\hat{J}(k)}{\partial\hat{w}_c(k)} \\ &= -\alpha_c\gamma\phi_c(x(k))e_c(k) \end{aligned} \tag{28}$$

Thus, the critic NN weight updating algorithm is obtained as

$$\hat{w}_c(k+1) = \hat{w}_c(k) - \alpha_c\gamma\phi_c(x(k))\left(\gamma\hat{J}(k) + r(k) - \hat{J}(k-1)\right) \tag{29}$$

## D. Weight Updating for the Action NN

The basis for adapting the action NN is to track the desired trajectory and to lower 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 cost function vector as $\bar{J}(k) = \left[\hat{J}(k) \quad \hat{J}(k) \quad \dots \quad \hat{J}(k)\right]^T \in R^{n\times 1}$. Let

$$\begin{aligned} e_a(k) &= \sqrt{g(x(k))}\zeta_a(k) + \left(\sqrt{g(x(k))}\right)^{-1}\left(\bar{J}(k) - J_d(k)\right) \\ &= \sqrt{g(x(k))}\zeta_a(k) + \left(\sqrt{g(x(k))}\right)^{-1}\bar{J}(k) \end{aligned} \tag{30}$$

where $\zeta_a(k)$ is defined in (16). Given Assumption 1, we define $\sqrt{g(x(k))} \in R^{n\times n}$ 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 $\left(\sqrt{g(x(k))}\right)^T = \sqrt{g(x(k))}$ [10]. 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 weights of the action NN $\hat{w}_a(k)$ are tuned to minimize the error

$$E_a(k) = \frac{1}{2}e_a^T(k)e_a(k) \tag{31}$$

Combining (14), (16), (18), (30), (31) and using the chain rule yields

$$\begin{aligned} \Delta\hat{w}_a(k) &= -\alpha_a\frac{\partial E_a(k)}{\partial\hat{w}_a(k)} = -\alpha_a\frac{\partial E_a(k)}{\partial e_a(k)}\frac{\partial e_a(k)}{\partial\zeta_a(k)}\frac{\partial\zeta_a(k)}{\partial\hat{w}_c(k)} \\ &= -\alpha_a\phi_a(s(k))\left(g(x(k))\zeta_a(k) + \bar{J}(k)\right)^T \\ &= -\alpha_a\phi_a(s(k))\left(e(k+1) - l_1e(k) - d_a(k) + \bar{J}(k)\right)^T \end{aligned} \tag{32}$$

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 assume the $d(k)$ and the mean value of $\varepsilon_a(s(k))$ over the compact subset of $R^{2n}$ to be zero, and obtain the weight updating algorithm for the action NN as

$$\hat{w}_a(k+1) = \hat{w}_a(k) - \alpha_a\phi_a(s(k))\left(e(k+1) - l_1e(k) + \bar{J}(k)\right)^T \tag{33}$$

## IV. Main Theoretic Result

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

$$\|w_a\| \le w_{am}, \text{ and } \|w_c\| \le w_{cm} \tag{34}$$

where $w_{am} \in R^+$ 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

$$\|\phi_a(k)\| \le \phi_{am}, \ \|\phi_c(k)\| \le \phi_{cm} \tag{35}$$

where $\phi_{am}, \phi_{cm} \in R^+$ is the upper bound for the activation functions.

***Assumption 5:*** The NN approximation errors $\varepsilon_a(s(k))$ and $\varepsilon_c(x(k))$ are bounded above over the compact set $S \subset R^n$ by $\varepsilon_{am}$ and $\varepsilon_{cm}$ [11].

***Fact 2:*** With the Assumption 1, 4, the term $d_a(k)$ in (17) is bounded over the compact set $S \subset R^n$ by

$$\|d_a(k)\| \le d_{am} = g_{\max}\varepsilon_{am} + d_m \tag{36}$$

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

***Theorem 1:*** Consider the system given by (1). Let the Assumptions 1 through 4 hold with the disturbance bound $d_m$ a known constant. Let the control input be provided by the action NN (13), with the critic NN (20). Further, let the weights of the action NN and the critic NN be tuned by (29) and (33) respectively. Then the tracking error $e(k)$, and the NN weight estimates of the action and critic NNs, $\hat{w}_a(k)$ and $\hat{w}_c(k)$ are *UUB*, with the bounds specifically given by (A.9) through (A.11) provided the controller design parameters are selected as

(a)
$$0<\alpha_a\left\|\phi_a(k)\right\|^2<\frac{g_{\min}}{g_{\max}^2} \quad (37)$$

(b)
$$0<\alpha_c\left\|\phi_c(x(k))\right\|^2<1 \quad (38)$$

(c)
$$0<l_{\max}<\frac{\sqrt{3}}{3} \quad (39)$$

(d)
$$\gamma>\frac{1}{2} \quad (40)$$

where $\alpha_a$ and $\alpha_c$ are NN adaptation gains, and $\alpha$ is employed to define the *strategic* utility function.
***Proof:*** See Appendix.

## V. Simulation results

To demonstrate the feasibility of the theoretic results, nanomanipulation system using the proposed controller is chosen as an example. Some of the parameters used in this simulation are set as follows: (note: $I_3$ is the identity matrix with dimension of 3)

TABLE 1
SUMMARY OF PARAMETERS USED IN SIMULATION OF NANOMANIPULATION

| Parameter | $w_x$ | $w_y$ | $w_z$ | $Q_x, Q_y, Q_z$ | $\theta$ |
|---|---|---|---|---|---|
| Value | 1570 rad/s | 1570 rad/s | 117.6 rad/s | 20 | 30° |
| Parameter | $R$ | $F$ | $\gamma$ | $\Lambda$ | $l_1$ |
| Value | 0.1 | 0.1 | 0.5 | 100 | $0.1\times I_3$ |
| Parameter | $n_c$ | $n_a$ | $\alpha_c$ | $\alpha_a$ | |
| Value | 20 | 20 | $1\times e^{-8}$ | $1\times e^{-8}$ | |

The simulation is run with time step of $1\times e^{-5}$. The effect of surface roughness in a form of sinusoid function is also introduced into the simulation as disturbance. The objective or the desired trajectory is to realize the movement of the particle along the sample surface with a constant speed. A proper force on the nano particle will indicate that the particle is being pushed by the tip, which could be observed by the movement of the stage along z axis.

Our online learning controller is first applied, with the results shown as in Fig. 4.

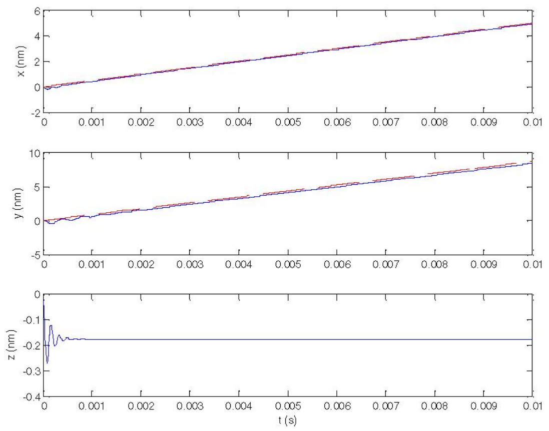

Fig. 4. Simulation results of the online learning controller on nanomanipulation system. Solid line: trajectories of the actual movement of the stage; Dashed line: desired movement of the stage. Note: there is no desired trajectory in the z axis.

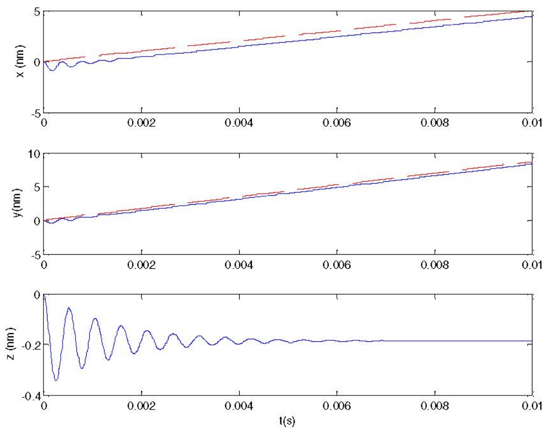

Fig. 5. Simulation results of PD controller on nanomanipulation system. Solid line: trajectories of the actual movement of the stage; Dashed line: desired movement of the stage.

To compare the performance, the system is also simulated with a typical PD controller. The results are shown at Fig. 5. From the comparison of the results, we can find that the online learning controller outperforms the PD controller in stabilizing the stage along the *z* axis, which also implies that an appropriate applied force on the particle is applied within a shorter time. Meanwhile, the cost of generating the input for the online learning controller is calculated to be 397.24, which is much better than that of the PD controller (541.44).

## VI. Conclusion

A novel reinforcement learning-based online neural controller is designed for affine nonlinear systems to deliver a desired performance under bounded disturbance. The proposed NN controller optimizes the long-term cost function by introducing a critic NN. Unlike the many applications where the controller is trained offline or trained by multiple iterations, the control signal in our scheme is updated in an online fashion.

Online learning control designs are especially useful for such complex systems whose dynamics are varying along with time and whose exact models are unreachable. At the same time, nanomanipulation system is a promising application and demands that the process is made automatic. However, its "fragile" dynamics exclude the implementation of iterative based control design, since no two manipulation attempts share a same dynamics. The "optimal" control policy being approached by a learning entity in one trial does not hold for another trial. In this regard, an online learner is more applicable.

To guarantee that a control system must be stable all the time, the UUB of the closed-loop tracking errors and NN weight estimates is verified by using Lyapunov analysis in the presence of bounded disturbances and approximation errors. Finally, the feasibility of our method is strengthened through the simulation results.

## APPENDIX

**Proof of Theorem 1**: Define the Lyapunov candidate as

$$L(k)=\sum_{i=1}^{4}L_i$$
$$=\frac{\gamma_1}{3}\|e(k)\|^2+\frac{\gamma_2}{\alpha_a}tr\left(\tilde{W}_a^T(k)\tilde{W}_a(k)\right)+\frac{\gamma_3}{\alpha_c}tr\left(\tilde{W}_c^T(k)\tilde{W}_c(k)\right) \quad \text{(A.1)}$$
$$+\gamma_4\|\zeta_c(k-1)\|^2$$

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

$$\Delta L_1=\frac{\gamma_1}{3}\left(\|e(k+1)\|^2-\|e(k)\|^2\right)$$
$$=\frac{\gamma_1}{3}\left(\|le(k)+g(x(k))\zeta_a(k)+d_a(k)\|^2-\|e(k)\|^2\right) \quad \text{(A.2)}$$
$$\le-\frac{\gamma_1}{3}\left(1-3l_{\max}^2\right)\|e(k)\|^2+\gamma_1 g_{\max}^2\|\zeta_a(k)\|^2+\gamma_1\|d_a(k)\|^2$$

$$\Delta L_2=\frac{\gamma_2}{\alpha_a}tr\left(\tilde{W}_a^T(k+1)\tilde{W}_a(k+1)-\tilde{W}_a^T(k)\tilde{W}_a(k)\right)$$
$$=\frac{\gamma_2}{\alpha_a}tr\left(-2\tilde{W}_a^T(k)\alpha_a\phi_a(s(k))\left(g(x(k))\zeta_a(k)+\bar{J}(k)+d_a(k)\right)\right.$$
$$\left.+\Delta\hat{W}_a^T(k)\Delta\hat{W}_a(k)\right)$$
$$=-2\gamma_2\zeta_a^T(k)g(x(k))\zeta_a(k)-2\gamma_2\zeta_a^T(k)\left(\bar{J}(k)+d_a(k)\right)$$
$$+\gamma_2\alpha_a\|\phi_a(s(k))\|^2\|g(x(k))\zeta_a(k)+\bar{J}(k)+d_a(k)\|^2$$
$$\le-2\gamma_2 g_{\min}\|\zeta_a(k)\|^2-2\gamma_2\zeta_a^T(k)\left(\bar{J}(k)+d_a(k)\right)$$
$$+\gamma_2\alpha_a\|\phi_a(s(k))\|^2 g_{\max}^2\|\zeta_a(k)\|^2+\gamma_2\alpha_a\|\phi_a(s(k))\|^2$$
$$\times\left(\|\bar{J}(k)+d_a(k)\|^2+2\gamma_2\alpha_a\left(\bar{J}(k)+d_a(k)\right)^T g(x(k))\zeta_a(k)\right)$$

$$=\gamma_2\left\{-g_{\min}\|\zeta_a(k)\|^2-\left(g_{\min}-\alpha_a\|\phi_a(s(k))\|^2 g_{\max}^2\right)\|\zeta_a(k)\|^2\right.$$
$$-2\zeta_a^T(k)\left(I-\alpha_a\|\phi_a(s(k))\|^2 g(x(k))\right)\left(\bar{J}(k)+d_a(k)\right)$$
$$\left.+\alpha_a\|\phi_a(s(k))\|^2\|\bar{J}(k)+d_a(k)\|^2\right\} \quad \text{(A.3)}$$
$$=\gamma_2\left\{-g_{\min}\|\zeta_a(k)\|^2-\left(g_{\min}-\alpha_a\|\phi_a(s(k))\|^2 g_{\max}^2\right)\right.$$
$$\times\left\|\zeta_a(k)+\frac{\left(I-\alpha_a\|\phi_a(s(k))\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\|\phi_a(s(k))\|^2 g_{\max}^2}\right\|^2$$
$$\left.+\frac{1-\alpha_a\|\phi_a(s(k))\|^2 g_{\min}}{g_{\min}-\alpha_a\|\phi_a(s(k))\|^2 g_{\max}^2}\|\bar{J}(k)+d_a(k)\|^2\right\}$$

Set $\gamma_2=\gamma_2'\gamma_2''$, where $\gamma_2''\frac{1-\alpha_a\|\phi_a(s(k))\|^2 g_{\min}}{g_{\min}-\alpha_a\|\phi_a(s(k))\|^2 g_{\max}^2}\le\frac{1}{2}$,

Therefore,

$$\Delta L_2\le-\gamma_2 g_{\min}\|\zeta_a(k)\|^2-\gamma_2\left(g_{\min}-\alpha_a\|\phi_a(s(k))\|^2 g_{\max}^2\right)$$
$$\times\left\|\zeta_a(k)+\frac{\left(I-\alpha_a\|\phi_a(s(k))\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\|\phi_a(s(k))\|^2 g_{\max}^2}\right\|^2+\gamma_2'\frac{\|\bar{J}(k)+d_a(k)\|^2}{2}$$
$$\le-\gamma_2 g_{\min}\|\zeta_a(k)\|^2-\gamma_2\left(g_{\min}-\alpha_a\|\phi_a(s(k))\|^2 g_{\max}^2\right)$$
$$\times\left\|\zeta_a(k)+\frac{\left(I-\alpha_a\|\phi_a(s(k))\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\|\phi_a(s(k))\|^2 g_{\max}^2}\right\|^2+\gamma_2'n\|\zeta_c(k)\|^2$$
$$+\gamma_2'n\|J^*(k)+d_a(k)\|^2 \quad \text{(A.4)}$$

At the same time,

$$\Delta L_3=\frac{\gamma_3}{\alpha_c}tr\left(\tilde{W}_c^T(k+1)\tilde{W}_c(k+1)-\tilde{W}_c^T(k)\tilde{W}_c(k)\right)$$
$$=\frac{\gamma_3}{\alpha_c}tr\left(2\tilde{W}_c^T(k)\Delta\hat{W}_c(k)+\Delta\hat{W}_c^T(k)\Delta\hat{W}_c(k)\right)$$
$$=\frac{\gamma_3}{\alpha_c}tr\left(-2\tilde{W}_c^T(k)\alpha_c\gamma\phi_c(x(k))e_c(k)\right)+\frac{\gamma_3}{\alpha_c}tr\left(\Delta\hat{W}_c^T(k)\Delta\hat{W}_c(k)\right)$$
$$=-2\gamma_3\gamma\zeta_c(k)e_c(k)+\gamma_3\alpha_c\gamma^2e_c^2(k)\|\phi_c(k)\|^2$$
$$=-2\gamma_3e_c(k)\left(e_c(k)-\gamma J^*(k)+\zeta_c(k-1)+J^*(k-1)-r(k)\right.$$
$$\left.+\varepsilon_c(k)-\varepsilon_c(k-1)\right)+\gamma_3\alpha_c\gamma^2e_c^2(k)\|\phi_c(k)\|^2$$
$$=-\gamma_3\left(1-\alpha_c\gamma^2\|\phi_c(k)\|^2\right)e_c^2(k)-\gamma_3e_c^2(k)+2\gamma_3e_c(k)\left(\gamma J^*(k)\right.$$
$$\left.-\zeta_c(k-1)-J^*(k-1)+r(k)-\varepsilon_c(k)+\varepsilon_c(k-1)\right)$$
$$=-\gamma_3\left(1-\alpha_c\gamma^2\|\phi_c(k)\|^2\right)e_c^2(k)-\gamma_3\gamma^2\zeta_c^2(k)$$
$$+\gamma_3\left(\gamma J^*(k)-\zeta_c(k-1)-J^*(k-1)+r(k)-\varepsilon_c(k)+\varepsilon_c(k-1)\right)^2$$
$$\le-\gamma_3\left(1-\alpha_c\gamma^2\|\phi_c(k)\|^2\right)e_c^2(k)-\gamma_3\gamma^2\zeta_c^2(k)+\frac{\gamma_3}{4}\zeta_c^2(k-1)$$
$$+\frac{\gamma_3}{4}\left(\gamma J^*(k)-J^*(k-1)\right)^2+\frac{\gamma_3}{4}r(k)+\frac{\gamma_3}{4}\left(\varepsilon_c(k)-\varepsilon_c(k-1)\right)^2$$

$$\begin{aligned}
&\le -\gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k)-\gamma_3\gamma^2\zeta_c^2(k)+\frac{\gamma_3}{4}\zeta_c^2(k-1)\\
&+\frac{\gamma_3}{4}\left(\gamma J^*(k)-J^*(k-1)\right)^2+\frac{\gamma_3}{4}Q_{\max}\left\|e(k)\right\|^2\\
&+\frac{\gamma_3}{4}\left(\zeta_a(k)+w_a{}^T\phi_a(k)\right)^T R\left(\zeta_a(k)+w_a{}^T\phi_a(k)\right)+\gamma_3\varepsilon_{cm}{}^2\\
&\le -\gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k)-\gamma_3\gamma^2\zeta_c^2(k)+\frac{\gamma_3}{4}\zeta_c^2(k-1)\\
&+\frac{\gamma_3}{4}\left(\gamma J^*(k)-J^*(k-1)\right)^2+\frac{\gamma_3}{4}Q_{\max}\left\|e(k)\right\|^2+\frac{\gamma_3}{8}R_{\max}\left\|\zeta_a(k)\right\|^2\\
&+\frac{\gamma_3}{8}R_{\max}\left\|w_a{}^T\phi_a(k)\right\|^2+\gamma_3\varepsilon_{cm}{}^2
\end{aligned} \tag{A.5}$$

where $Q_{\max}$ and $R_{\max}$ are the maximum eigenvalue of matrix $Q$ and $R$ respectively and

$$\Delta L_4=\gamma_4\left(\left\|\zeta_c(k)\right\|^2-\left\|\zeta_c(k-1)\right\|^2\right) \tag{A.6}$$

Combining (A.1) - (A.6) yields

$$\begin{aligned}
\Delta L(k)\le&-\frac{\gamma_1}{3}\left(1-3l_{\max}^2\right)\left\|e(k)\right\|^2+\gamma_1 g_{\max}^2\left\|\zeta_a(k)\right\|^2+\gamma_1\left\|d_a(k)\right\|^2\\
&-\gamma_2 g_{\min}\left\|\zeta_a(k)\right\|^2-\gamma_2\left(g_{\min}-\alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2\right)\\
&\times\left\|\zeta_a(k)+\frac{\left(I-\alpha_a\left\|\phi_a(s(k))\right\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2}\right\|^2+\gamma_2' n\left\|\zeta_c(k)\right\|^2\\
&+\gamma_2' n\left\|J^*(k)+d_a(k)\right\|^2-\gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k)\\
&-\gamma_3\gamma^2\zeta_c^2(k)+\frac{\gamma_3}{4}\zeta_c^2(k-1)+\frac{\gamma_3}{4}\left(\gamma J^*(k)-J^*(k-1)\right)^2\\
&+\frac{\gamma_3}{4}Q_{\max}\left\|e(k)\right\|^2+\frac{\gamma_3}{8}R_{\max}\left\|\zeta_a(k)\right\|^2\\
&+\frac{\gamma_3}{8}R_{\max}\left\|w_a{}^T\phi_a(k)\right\|^2+\gamma_4\left(\left\|\zeta_c(k)\right\|^2-\left\|\zeta_c(k-1)\right\|^2\right)+\gamma_3\varepsilon_{cm}{}^2\\
=&-\left(\frac{\gamma_1}{3}\left(1-3l_{\max}^2\right)-\frac{\gamma_3}{4}Q_{\max}\right)\left\|e(k)\right\|^2\\
&-\left(\gamma_2 g_{\min}-\gamma_1 g_{\max}^2-\frac{\gamma_3}{8}R_{\max}\right)\left\|\zeta_a(k)\right\|^2\\
&-\left(\gamma_3\gamma^2-\gamma_2' n-\gamma_4\right)\left\|\zeta_c(k)\right\|^2-\left(\gamma_4-\frac{\gamma_3}{4}\right)\left\|\zeta_c(k-1)\right\|^2\\
&-\gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k)-\gamma_2\left(g_{\min}-\alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2\right)\\
&\times\left\|\zeta_a(k)+\frac{\left(I-\alpha_a\left\|\phi_a(s(k))\right\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2}\right\|^2+D_M^2
\end{aligned} \tag{A.7}$$

where

$$\begin{aligned}
D_M^2=&\left(\frac{\gamma_3}{4}+\frac{\gamma_2' n}{2}\right)J_m^2+\frac{\gamma_3}{6}R_{\max}\left\|w_a{}^T\phi_a(k)\right\|^2\\
&+\left(\gamma_1+\frac{\gamma_2' n}{2}\right)\left\|d_a(k)\right\|^2+\gamma_3\varepsilon_{cm}{}^2
\end{aligned} \tag{A.8}$$

For the standard Lyapunov analysis, equation (A.7) and (A.8) implies that $\Delta L\le 0$ as long as the conditions (37) – (40) are satisfied and following holds

$$\left\|e(k)\right\|\ge\frac{2\sqrt{3}D_M}{\sqrt{4\gamma_1\left(1-3l_{\max}^2\right)-3\gamma_3 Q_{\max}}} \tag{A.9}$$

or

$$\left\|\zeta_a(k)\right\|\le\frac{2\sqrt{2}D_M}{\sqrt{8\gamma_2 g_{\min}-8\gamma_1 g_{\max}^2-\gamma_3 R_{\max}}} \tag{A.10}$$

or

$$\left\|\zeta_c(k)\right\|\le\frac{D_M}{\sqrt{\gamma_3\gamma^2-\gamma_2' n-\gamma_4}} \tag{A.11}$$

According to the standard Lyapunov extension theorem [17], 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] R. Luus, "Iterative Dynamic Programming", CRC Press, Boca Raton, FL, 2000.

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

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

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

[6] 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, Mar.2001.

[7] 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 .

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

[9] 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.

[10] 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.

[11] Y. Kim and F..L. Lewis, "Optimal design of CMAC neural network controller for Robot Manipulators," IEEE Trans. Systems, Man, and Cybernetics, vol. 30, no. 1, pp. 22-31, Feb 2000.

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

[13] Q. 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. on Mechatronics, July 2000.

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

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

[17] S. Jagannathan, "Neural Network Control of Nonlinear Discrete-time Systems", Taylor and Francis (CRC Press), Boca Raton, FL 2006.