01 Jul 2007

Online Reinforcement Learning-Based Neural Network Controller Design for 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, "Online Reinforcement Learning-Based Neural Network Controller Design for Affine Nonlinear Discrete-Time Systems," *Proceedings of the American Control Conference, 2007. ACC'07*, Institute of Electrical and Electronics Engineers (IEEE), Jul 2007.
The definitive version is available at https://doi.org/10.1109/ACC.2007.4282709

Proceedings of the 2007 American Control Conference
Marriott Marquis Hotel at Times Square
New York City, USA, July 11-13, 2007

Online Reinforcement Learning-based Neural Network Controller Design for Affine Nonlinear Discrete-time Systems

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 general multi-input and multi-output affine unknown nonlinear discrete-time systems in the presence of bounded disturbances. Adaptive critic designs consist of two entities, an action network that produces optimal solution 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 and the action network is adapted simultaneously based on the information from the critic. In our online learning method, one NN is designated as the critic NN, which approximates the Bellman equation. An action NN is employed to derive the 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 and uniformly ultimate boundedness (UUB) of the tracking error and weight estimates is shown. The effectiveness of the controller is evaluated on a two-link robotic arm system.**

I. Introduction

In the literature, there are many approaches proposed for designing stable controllers for nonlinear systems. However, stability is only a bare requirement for the controller design. A further consideration is the optimality based on a cost function which is used to determine the performance. Of the available methods, dynamic programming (DP) has been extensively applied to generate optimal control for nonlinear systems [1]-[2]. However, one of the drawbacks for DP is the computation cost with the increasing dimension of the nonlinear system, which is referred to as the "curse of dimensionality" [2]. Therefore, adaptive methods for DP (e.g., see [3]) have been developed recently. However, most of them are implemented either by offline using iterative schemes or require the nonlinear system dynamics to be known *a priori*. These requirements are often not practical for real-world systems. Additionally, stability of the closed-loop system is not discussed.

On the other hand, 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. However, to obtain a satisfactory reinforcement signal for each action, the approach must visit each system state and apply each action often enough [6], 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 [7]-[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 information that is fed back from the system response. Depending upon whether the NNs approximate the cost function or its derivative, or both are approximated by a critic structure, this family of ACD is classified into three categories: 1) heuristic dynamic programming (HDP); 2) dual heuristic dynamic programming (DHP); and 3) globalized dual heuristic dynamic programming (DHP). It is important to note that when the action value is introduced as an additional input to the critic, then it will be referred to action dependent (AD) version.

In [10], a new NN learning algorithm based on gradient descent rule is introduced without any convergence or stability proof. By contrast, Lyapunov analysis was derived in [11] and [12]. However, approach presented in [12] is specific to robotic systems in continuous-time whereas [10] and [11] employ simplified binary reward or cost function, which is a variant of the 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 entire 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 long-term cost function; an adaptive critic NN to approximate the long-term cost function and to tune the action NN weights. Since the control signal is not used in the critic NN as an additional input, our approach could be seen as a HDP approach.

Besides addressing optimization, contributions of this paper include: 1) the demonstration of the UUB of the overall system is shown even in the presence of NN approximation errors and bounded unknown disturbances unlike in the existing adaptive critic works where the convergence is given under ideal circumstances; 2) the NN weights are tuned online instead of offline; and 3) the proposed approach uses the standard Bellman equation and not a variant of the Bellman equation as in [11].

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 awards ECCS #0327877 and ECCS#0621924.

II. BACKGROUND

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

$$x(k+1) = f_0(x(k), u(k)) = f(x(k)) + g(x(k))u(k) + d(k) \quad (1)$$

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 $\|\cdot\|$ stands for the Frobenius norm [13], which will be used through out this paper. It is also assumed that the state vector $x(k)$ is available.

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 $g_{\min}$, $g_{\max} \in R^+$ represent the minimum and maximum eigenvalues of $g(x(k))$ respectively, such that $0 < g_{\min} \le g_{\max}$.

The long-term cost function is defined as

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

where $J(k)$ stands for $J(x(k), u)$ for simplicity, and u is a control policy. As observed from (2), the long-term cost function is the discounted sum of the immediate cost function or Lagrangian

$$r(k) = q(x(k)) + u^T(k)Ru(k) = (x(k) - x_d(k))^T Q(x(k) - x_d(k)) + u^T(k)Ru(k) \quad (3)$$

where R and Q are positive definite matrices and γ $(0 \le \gamma \le 1)$ is the discount factor for the infinite-horizon problem. In this paper, we are using a widely used standard quadratic cost function defined based on the tracking error $e(k)$, which will be defined later in contrast with [10]. The immediate cost function $r(k)$ can be viewed as the 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 (or its derivative, or both), and generate the control signal minimizing the cost. By using a learning algorithm, the 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 [14].

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

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.

The block diagram of the proposed controller is shown in Fig. 1 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.

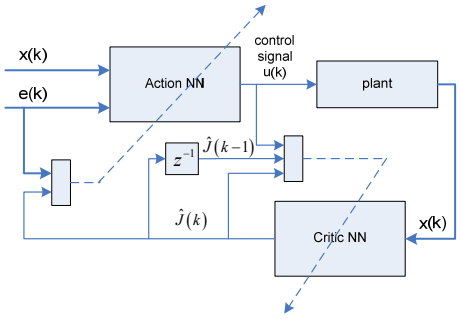

Fig. 1. Online neural dynamic programming based controller structure.

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

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

$$f(x) = W^T\phi(V^T x) + \varepsilon(x) \quad (4)$$

with $\varepsilon(x)$ a NN functional reconstruction error vector. In our design, V is selected initially at random and held fixed during learning. 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 the proposed adaptive controller, the persistence of excitation (PE) condition is relaxed. 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 a smooth bounded function over the compact subset of R^n.

A. The Action NN Design

The tracking error at instant *k* is defined as

$$e(k) = x(k) - x_d(k) \tag{5}$$

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

$$e(k+1) = f(x(k)) + g(x(k))u(k) + d(k) - x_d(k+1) \tag{6}$$

The desired control signal can be given by

$$u_d(k) = g^{-1}(x(k))(-f(x(k) + x_d(k+1) + l_1 e(k)) \tag{7}$$

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 (7) and considering Assumption 1 and 2, the desired control signal can be approximated as

$$u_d(k) = w_a^T \phi_a(v_a^T s(k)) + \varepsilon_a(s(k)) = w_a^T \phi_a(k) + \varepsilon_a(k) \tag{8}$$

where $s(k) = \left[x^T(k), e^T(k)\right]^T \in R^{2n}$ 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 $\varepsilon_a(k) = \varepsilon_a(s(k))$ is the action NN approximation error, and n_a is the number of the neurons in the hidden layer. Since v_a is fixed, for simplicity purpose, the hidden layer activation function vector $\phi_a(v_a^T s(k)) \in R^{n_2}$ is denoted as $\phi_a(k)$.

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

$$u(k) = \hat{w}_a^T(k)\phi_a(v_a^T s(k)) = \hat{w}_a^T(k)\phi_a(k) \tag{9}$$

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 (8) and (9) into (6) yields

$$\begin{aligned} e(k+1) &= f(x(k)) + g(x(k))v(k) + d(k) - x_d(k+1) \\ &= l_1 e(k) + g(x(k))(v(k) - v_d(k)) + d(k) \\ &= l_1 e(k) + g(x(k))(\tilde{w}_a^T(k)\phi_a(k) - \varepsilon_a(k)) + d(k) \\ &= l_1 e(k) + g(x(k))\zeta_a(k) + d_a(k) \end{aligned} \tag{10}$$

where

$$\tilde{w}_a(k) = \hat{w}_a(k) - w_a \tag{11}$$

$$\zeta_a(k) = \tilde{w}_a^T(k)\phi_a(k) \tag{12}$$

$$d_a(k) = -g(x(k))\varepsilon_a(k) + d(k) \tag{13}$$

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

$$e(k+1) = l_1 e(k) + g(x(k))\zeta_a(k) + d_a(k) \tag{14}$$

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^{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 of the critic or the Bellman error [10] is defined as

$$e_c(k) = \gamma \hat{J}(k) - [\hat{J}(k-1) - r(k)] \tag{15}$$

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

$$\hat{J}(k) = \hat{w}_c^T(k)\phi_c(v_c^T x(k)) = \hat{w}_c^T(k)\phi_c(x(k)) \tag{16}$$

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 $\phi_c(k)$ 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 $\varepsilon_c(k)$,

$$J*(k) = w_c^T \phi_c(v_c^T x(k)) + \varepsilon_c(x(k)) = w_c^T \phi_c(k) + \varepsilon_c(k) \tag{17}$$

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

$$\tilde{w}_c(k) = \hat{w}_c(k) - w_c \tag{18}$$

where the approximation error is given by

$$\zeta_c(k) = \tilde{w}_c^T(k)\phi_c(k) \tag{19}$$

Thus, we have

$$\begin{aligned} e_c(k) &= \gamma \hat{J}(k) - \hat{J}(k-1) + r(k) \\ &= \gamma\zeta_c(k) + \gamma J*(k) - \zeta_c(k-1) - J*(k-1) \\ &\quad + r(k) - \varepsilon_c(k) + \varepsilon_c(k-1) \end{aligned} \tag{20}$$

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

$$E_c(k) = e_c^T(k)e_c(k)/2 = e_c^2(k)/2 \tag{21}$$

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

$$\hat{w}_c(k+1) = \hat{w}_c(k) + \Delta\hat{w}_c(k) \tag{22}$$

where

$$\Delta \hat{w}_c(k) = \alpha_c \left[-\frac{\partial E_c(k)}{\partial \hat{w}_c(k)} \right] \tag{23}$$

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

Combining (15), (16), (21) with (23), the critic NN weight updating rule can be obtained by using the chain rule as

$$\begin{aligned} \Delta \hat{w}_c(k) &= -\alpha_c \frac{\partial E_c(k)}{\partial \hat{w}_c(k)} = -\alpha_c \frac{\partial E_c(k)}{\partial e_c(k)} \frac{\partial e_c(k)}{\partial \hat{J}(k)} \frac{\partial \hat{J}(k)}{\partial \hat{w}_c(k)} \\ &= -\alpha_c \gamma \phi_c(k) e_c(k) \end{aligned} \tag{24}$$

Thus, the critic NN weight updating algorithm is obtained as

$$\hat{w}_c(k+1) = \hat{w}_c(k) - \alpha_c \gamma \phi_c(k)(\gamma \hat{J}(k) + r(k) - \hat{J}(k-1)) \tag{25}$$

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 $\bar{J}(k) = \left[\hat{J}(k) \quad \hat{J}(k) \quad \ldots \quad \hat{J}(k) \right]^T \in R^{n \times 1}$. Let

$$\begin{aligned} e_a(k) &= \sqrt{g(x(k))}\zeta_a(k) + \left(\sqrt{g(x(k))}\right)^{-1} (\bar{J}(k) - J_d(k)) \\ &= \sqrt{g(x(k))}\zeta_a(k) + \left(\sqrt{g(x(k))}\right)^{-1} \bar{J}(k) \end{aligned} \tag{26}$$

where $\zeta_a(k)$ is defined in (12). 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 $\left(\sqrt{g(x(k))}\right)^T = \sqrt{g(x(k))}$ [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 weights of the action NN $\hat{w}_a(k)$ are tuned to minimize the error

$$E_a(k) = e_a^T(k) e_a(k) / 2 \tag{27}$$

Combining (10), (12), (14), (26), (27) and using the chain rule yields

$$\begin{aligned} \Delta \hat{w}_a(k) &= -\alpha_a \frac{\partial E_a(k)}{\partial \hat{w}_a(k)} = -\alpha_a \frac{\partial E_a(k)}{\partial e_a(k)} \frac{\partial e_a(k)}{\partial \zeta_a(k)} \frac{\partial \zeta_a(k)}{\partial \hat{w}_c(k)} \\ &= -\alpha_a \phi_a(k)(g(x(k))\zeta_a(k) + \bar{J}(k))^T \\ &= -\alpha_a \phi_a(k)(e(k+1) - l_1 e(k) - d_a(k) + \bar{J}(k))^T \end{aligned} \tag{28}$$

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 and obtain the weight updating algorithm for the action NN as

$$\hat{w}_a(k+1) = \hat{w}_a(k) - \alpha_a \phi_a(k)(e(k+1) - l_1 e(k) + \bar{J}(k))^T \tag{29}$$

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

$$\|w_a\| \le w_{am}, \text{ and } \|w_c\| \le w_{cm} \tag{30}$$

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

$$\|\phi_a(k)\| \le \phi_{am}, \ \|\phi_c(k)\| \le \phi_{cm} \tag{31}$$

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

Assumption 5: The NN approximation errors $\varepsilon_a(k)$ and $\varepsilon_c(k)$ are bounded above over the compact set $S \subset R^n$ by ε_{am} and ε_{cm} [13].

Fact 2: With the Assumption 1, 4, the term $d_a(k)$ in (13) is bounded over the compact set $S \subset R^n$ by

$$\|d_a(k)\| \le d_{am} = g_{\max}\varepsilon_{am} + d_m \tag{32}$$

Combining Assumption 1, 3 and 4, with Facts 1 and 2, the main result of this paper is introduced as 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 (9), with the critic NN (16). Further, let the weights of the action NN and the critic NN be tuned by (24) and (28) 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) $$0 < \alpha_a \|\phi_a(k)\|^2 < g_{\min} / g_{\max}^2 \tag{33}$$

(b) $$0 < \alpha_c \|\phi_c(k)\|^2 < 1 \tag{34}$$

(c) $$0 < l_{\max} < \sqrt{3}/3 \tag{35}$$

(d) $$\gamma > 1/2 \tag{36}$$

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

Proof: See Appendix.

Remark 1: The action and critic NN weights can be initialized at zero or random. This means that there is no explicit off-line learning phase needed.

Remark 2: It is important to note that persistency of excitation condition is not utilized in order to show the boundedness of the NN weights and certainty equivalence principle is not employed.

V. Simulation Results

In the implementation, the two-link planar robot arm system discussed in [13] is considered and the dynamics are

discretized as an affine nonlinear discrete-time system with standard zero-order-hold discretization techniques [14]. The continuous-time manipulator dynamics is as follows [13]

$$\begin{bmatrix} \alpha+\beta+2\eta\cos q_2 & \beta+\eta\cos q_2 \\ \beta+\eta\cos q_2 & \beta \end{bmatrix}\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}+\begin{bmatrix} -\eta(2\dot{q}_1\dot{q}_2+\dot{q}_2^2)\sin q_2 \\ \eta\dot{q}_1^2\sin q_2 \end{bmatrix} + \begin{bmatrix} \alpha e_1\cos q_1+\eta e_1\cos(q_1+q_2) \\ \eta e_1\cos(q_1+q_2) \end{bmatrix}=\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{37}$$

where $\alpha=(m_1+m_2)a_1^2$, $\beta=m_2a_2^2$, $\eta=m_2a_1a_2$, $e_1=g/a_1$, $g=9.8\ m/s^2$, the acceleration of gravity; m_1, m_2 denote point mass of the links at distal end; a_1, a_2 represent length of the links; q_1, q_2 denote rotational angle of the joints; and τ_1, τ_2 represent torque applied on the joints.

In the controller designs, joint angles q_1 and q_2 are the states while τ_1 and τ_2 are the control inputs. To fulfill Assumption 1, we define the filtered tracking error as $s=\dot{e}+\Lambda e$, and use them as the system states with Λ is a positive definite design parameter matrix. Common usage is to select Λ diagonal with large positive entries. Then the filtered error s is bounded as long as the controller guarantees that the tracking error e is bounded. The filtered tracking error is also treated as the system state for the pendulum example in this paper. The parameters used in our simulation are tabulated as below:

TABLE 1
SUMMARY OF PARAMETERS USED IN SIMULATION OF 2-LINK ROBOT ARM

Parameter	m_1	m_2	a_1	a_2	R	F	γ
Value	0.8	2.3	1	1	2	1	0.5
Parameter	Λ	l_1	n_c	n_a	α_c	α_a	
Value	10	0.8	10	10	$1\times e^{-4}$	$1\times e^{-5}$	

In the simulation, the time step is set as 1 ms. To be more realistic, the system was added with a uniformly distributed random disturbance. The bound of the disturbance d_m is selected to be 0.1. The initial states of the system are set at $q_1(0)=q_2(0)=10^\circ$. Our goal is to manipulate the robot arm back to zero condition with the lowest cost and simulation will be stopped when the rotational angles converge to zero.

To obtain a measurement of the cost to accomplish the task, the simulation is run 100 times. At the beginning of each running instance, the NN weights are initialized randomly to exclude any information about the system obtained from previous learning. During the simulation, the cost function as defined in (2) is accumulated online for 10 seconds with parameters shown in Table 1. The average cost of the 100 trials for the algorithm is calculated as 79.94. For comparison, an adaptive NN controller is also implemented [13] with same parameters, and the average cost of 100 trial running is computed to be 129.92. From the simulation results (not shown), it can be noted that the performance of the proposed controller design is highly satisfactory.

VI. CONCLUSIONS

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, the control input is updated in an online fashion. To guarantee that a control system must be stable all of 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.

APPENDIX

Proof of Theorem 1: Define the Lyapunov candidate as

$$\begin{aligned} L(k)&=\sum_{i=1}^{4}L_i=\gamma_1\|e(k)\|^2/3+\gamma_2 tr(\tilde{W}_a^T(k)\tilde{W}_a(k))/\alpha_a \\ &\quad+\gamma_3 tr(\tilde{W}_c^T(k)\tilde{W}_c(k))/\alpha_c+\gamma_4\|\zeta_c(k-1)\|^2 \end{aligned} \tag{A.1}$$

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

$$\begin{aligned} \Delta L_1&=\gamma_1(\|e(k+1)\|^2-\|e(k)\|^2)/3 \\ &=\gamma_1(\|le(k)+g(x(k))\zeta_a(k)+d_a(k)\|^2-\|e(k)\|^2)/3 \\ &\le-\gamma_1(1-3l_{\max}^2)\|e(k)\|^2/3+\gamma_1 g_{\max}^2\|\zeta_a(k)\|^2+\gamma_1\|d_a(k)\|^2 \end{aligned} \tag{A.2}$$

$$\begin{aligned} \Delta L_2&=\gamma_2 tr(\tilde{W}_a^T(k+1)\tilde{W}_a(k+1)-\tilde{W}_a^T(k)\tilde{W}_a(k))/\alpha_a \\ &=-2\gamma_2\zeta_a^T(k)g(x(k))\zeta_a(k)-2\gamma_2\zeta_a^T(k)(\bar{J}(k)+d_a(k)) \\ &\quad+\gamma_2\alpha_a\|\phi_a(k)\|^2\|g(x(k))\zeta_a(k)+\bar{J}(k)+d_a(k)\|^2 \\ &\le-2\gamma_2 g_{\min}\|\zeta_a(k)\|^2-2\gamma_2\zeta_a^T(k)(\bar{J}(k)+d_a(k)) \\ &\quad+\gamma_2\alpha_a\|\phi_a(k)\|^2 g_{\max}^2\|\zeta_a(k)\|^2+\gamma_2\alpha_a\|\phi_a(k)\|^2 \\ &\quad\times\left(\|\bar{J}(k)+d_a(k)\|^2+2\gamma_2\alpha_a(\bar{J}(k)+d_a(k))^T g(x(k))\zeta_a(k)\right) \\ &=\gamma_2\left\{-g_{\min}\|\zeta_a(k)\|^2-\left(g_{\min}-\alpha_a\|\phi_a(k)\|^2 g_{\max}^2\right)\|\zeta_a(k)\|^2\right. \\ &\quad-2\zeta_a^T(k)(I-\alpha_a\|\phi_a(k)\|^2 g(x(k)))(\bar{J}(k)+d_a(k)) \\ &\quad\left.+\alpha_a\|\phi_a(k)\|^2\|\bar{J}(k)+d_a(k)\|^2\right\} \\ &=\gamma_2\left\{-g_{\min}\|\zeta_a(k)\|^2-(g_{\min}-\alpha_a\|\phi_a(k)\|^2 g_{\max}^2)\right. \\ &\quad\times\left\|\zeta_a(k)+\frac{I-\alpha_a\|\phi_a(k)\|^2 g(x(k))}{g_{\min}-\alpha_a\|\phi_a(k)\|^2 g_{\max}^2}\right\|^2 \\ &\quad\left.+\frac{1-\alpha_a\|\phi_a(k)\|^2 g_{\min}}{g_{\min}-\alpha_a\|\phi_a(k)\|^2 g_{\max}^2}\|\bar{J}(k)+d_a(k)\|^2\right\} \end{aligned} \tag{A.3}$$

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

$$\begin{aligned}\Delta L_2 &\le -\gamma_2 g_{\min}\left\|\zeta_a(k)\right\|^2 - \gamma_2(g_{\min}-\alpha_a\left\|\phi_a(k)\right\|^2 g_{\max}^2)\\ &\times\left\|\zeta_a(k)+\frac{\left(I-\alpha_a\left\|\phi_a(k)\right\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\left\|\phi_a(k)\right\|^2 g_{\max}^2}\right\|^2 + \gamma_2'\frac{\left\|\overline{J}(k)+d_a(k)\right\|^2}{2} \qquad (A.4)\\ &\le -\gamma_2 g_{\min}\left\|\zeta_a(k)\right\|^2 - \gamma_2\left(g_{\min}-\alpha_a\left\|\phi_a(k)\right\|^2 g_{\max}^2\right)\\ &\times\left\|\zeta_a(k)+\frac{\left(I-\alpha_a\left\|\phi_a(k)\right\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\left\|\phi_a(k)\right\|^2 g_{\max}^2}\right\|^2 + \gamma_2' n\left\|\zeta_c(k)\right\|^2\\ &+\gamma_2' n\left\|J^*(k)+d_a(k)\right\|^2\end{aligned}$$

At the same time,

$$\begin{aligned}\Delta L_3 &= \gamma_3 tr(\tilde{W}_c^T(k+1)\tilde{W}_c(k+1)-\tilde{W}_c^T(k)\tilde{W}_c(k))/\alpha_c\\ &= -2\gamma_3\gamma\zeta_c(k)e_c(k)+\gamma_3\alpha_c\gamma^2 e_c^2(k)\left\|\phi_c(k)\right\|^2\\ &= -2\gamma_3 e_c(k)\left(e_c(k)-\gamma J^*(k)+\zeta_c(k-1)+J^*(k-1)-r(k)\right.\\ &\left.+\varepsilon_c(k)-\varepsilon_c(k-1)\right)+\gamma_3\alpha_c\gamma^2 e_c^2(k)\left\|\phi_c(k)\right\|^2\\ &= -\gamma_3(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2)e_c^2(k)-\gamma_3 e_c^2(k)+2\gamma_3 e_c(k)\left(\gamma J^*(k)\right.\\ &\left.-\zeta_c(k-1)-J^*(k-1)+r(k)-\varepsilon_c(k)+\varepsilon_c(k-1)\right)\\ &= -\gamma_3(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2)e_c^2(k)-\gamma_3\gamma^2\zeta_c^2(k)\\ &+\gamma_3(\gamma J^*(k)-\zeta_c(k-1)-J^*(k-1)+r(k)-\varepsilon_c(k)+\varepsilon_c(k-1))^2\\ &\le -\gamma_3((1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2)e_c^2(k)-\gamma^2\zeta_c^2(k)+\zeta_c^2(k-1)/4\\ &+(\gamma J^*(k)-J^*(k-1))^2/4+r(k)/4+(\varepsilon_c(k)-\varepsilon_c(k-1))^2/4)\\ &\le -\gamma_3(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2)e_c^2(k)-\gamma_3\gamma^2\zeta_c^2(k)+\gamma_3\zeta_c^2(k-1)/4\\ &+\gamma_3(\gamma J^*(k)-J^*(k-1))^2/4+\gamma_3 Q_{\max}\left\|e(k)\right\|^2/4\\ &+\gamma_3 R_{\max}\left\|\zeta_a(k)\right\|^2/8+\gamma_3 R_{\max}\left\|w_a^T\phi_a(k)\right\|^2/8+\gamma_3\varepsilon_{cm}{}^2\end{aligned} \qquad (A.5)$$

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

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

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

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

where

$$\begin{aligned}D_M^2 &= (\gamma_2' n/2+\gamma_1)\left\|d_a(k)\right\|^2+\left(\gamma_3/4+\gamma_2' n/2\right)J_m^2\\ &+\gamma_3 R_{\max}w_{am}^2\phi_{am}^2/6+\gamma_3\varepsilon_{cm}{}^2\end{aligned} \qquad (A.8)$$

For the standard Lyapunov analysis, equation (A.7) and (A.8) implies that $\Delta L \le 0$ as long as the conditions (32) – (35) are satisfied and following holds

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

or

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

or

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

According to the Lyapunov extension theorem [13], 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] R. Luus, "Iterative Dynamic Programming", CRC Press, Boca Raton, FL, 2000.

[4] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuron like adaptive elements that can solve difficult learning control problems," IEEE Trans. Syst., Man, Cybern., vol. 13, pp. 834–847, 1983.

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

[6] 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.

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

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

[9] W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds., "Neural Networks for Control", Cambridge, MA, MIT Press, 1990.

[10] 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.

[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] 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.

[13] S. Jagannathan, "Neural Network Control of Nonlinear Discrete-time Systems," Taylor & Francis, FL, 2007.

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