Adaptive Learning in Tracking Control Based on the Dual Critic Network Design

Zhen Ni, Haibo He, *Senior Member, IEEE*, and Jinyu Wen, *Member, IEEE*

***Abstract*— In this paper, we present a new adaptive dynamic programming approach by integrating a reference network that provides an internal goal representation to help the systems learning and optimization. Specifically, we build the reference network on top of the critic network to form a dual critic network design that contains the detailed internal goal representation to help approximate the value function. This internal goal signal, working as the reinforcement signal for the critic network in our design, is adaptively generated by the reference network and can also be adjusted automatically. In this way, we provide an alternative choice rather than crafting the reinforcement signal manually from prior knowledge. In this paper, we adopt the online action-dependent heuristic dynamic programming (ADHDP) design and provide the detailed design of the dual critic network structure. Detailed Lyapunov stability analysis for our proposed approach is presented to support the proposed structure from a theoretical point of view. Furthermore, we also develop a virtual reality platform to demonstrate the real-time simulation of our approach under different disturbance situations. The overall adaptive learning performance has been tested on two tracking control benchmarks with a tracking filter. For comparative studies, we also present the tracking performance with the typical ADHDP, and the simulation results justify the improved performance with our approach.**

***Index Terms*— Adaptive critic design (ACD), adaptive dynamic programming (ADP), internal goal, lyapunov stability analysis, online learning, reinforcement learning, tracking control, virtual reality.**

I. Introduction

DEVELOPING brain-like intelligence has become a cutting-edge research topic in the computational intelligence field. While the latest research on adaptive dynamic programming (ADP)/adaptive critic design (ACD) [1]–[6] has shown great success on machine intelligence and applications, there are still many grand challenges in reaching truly brain-like intelligence. One of the most important questions is how to design a general system that can learn and optimize adaptively and efficiently like the human brain to achieve the final goal over time [7], [8]. In this paper, we propose a novel structure based on heuristic dynamic programming (HDP) to tackle this problem.

In recent years, extensive efforts have been dedicated to developing machine intelligence, and ADP is one of the most critical approaches to (hopefully) bring such a level of intelligence closer to reality [8]–[10]. Generally speaking, ADP can be categorized into three typical structures [11], [12].

1) HDP, which was proposed to provide a control action sequence with the action network and use the critic network to approximate the value function. For instance, in online HDP version such as [13], the critic network was designed to critique the generated action by propagating a temporal difference (TD) between two consecutive estimates of the value function.
2) Dual heuristic dynamic programming (DHDP), the key point of which is to employ the critic network to approximate the derivatives of the value function with respect to the state vectors.
3) Globalized dual heuristic dynamic programming (GDHP), which takes advantage of both HDP and DHP by employing the critic network to approximate both the value function and its derivatives with respect to the state vectors.

Moreover, various versions have been developed based on these typical structures, such as the action-dependent (AD) version [13]–[15] by taking the control action as one of the inputs for the critic network, and the model-dependent version [16], [17] by taking the estimates of the model network as part of the inputs for the critic network.

Currently, many implementations of these ADP designs have been tested on both mathematical benchmarks and engineering applications, such as the infinite-time optimal tracking control with greedy HDP algorithm in [18]–[20], the HDP controller for nonlinear discrete time tracking problem in [21]–[25], and the engine torque and exhaust air–fuel ratio tracking control based on AD HDP in [26]. In [27]–[29], the looper system control in the iron industry was improved with the Levenberg–Marquardt (LM) algorithm instead of back-propagation as the weights updating rules in neural networks, and in [16] and [30]–[33] the turbogenerator/power system control was improved with HDP and DHP controllers, and many others [34]–[39].

Although there are many successful applications with ADP approaches, many of them use either a binary reinforcement signal ("0" and "−1") or some specific discrete reinforcement

Manuscript received November 29, 2011; revised November 28, 2012; accepted February 10, 2013. Date of publication March 7, 2013; date of current version April 5, 2013. This work was supported by the National Science Foundation under grant CAREER ECCS 1053717 and grant CNS 1117314, the Army Research Office under grant W911NF-12-1-0378, the Defense Advanced Research Projects Agency under grant FA8650-11-1-7148, and the National Natural Science Foundation of China under grant 51228701.

Z. Ni and H. He are with Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881 USA (e-mail: ni@ele.uri.edu; he@ele.uri.edu).

J. Wen is with the College of Electrical, Electronic and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: jinyu.wen@hust.edu.cn).

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

Digital Object Identifier 10.1109/TNNLS.2013.2247627

signals ("0," "−0.4," and "−1") [13], [21], [22]. Some others crafted the reinforcement signal manually with past experience or prior knowledge. For instance, in [11], the square of the difference between the reference signal and the actual output of the system is set as the reinforcement signal. In [40], the reinforcement signal is defined with different weights for various components, with some of the error components even having nonlinear coefficients. Therefore, an intuitive question is: can the reinforcement signal be assigned without any (or minimum) prior knowledge or human intervention and be adjusted adaptively when the operation environment changes?

In this paper, we propose a novel ADP structure with dual critic networks that provide the internal goal representation adaptively according to the system's behaviors to tackle this question. Specifically, we introduce one reference network to be on top of the typical ADP design, through which we maintain the advantage of a model-free AD structure [13]. In this way, we build one reference network on top of the critic network to work as an integrated dual critic network structure, where the reference network provides the critic network with an internal reinforcement signal. Unlike the binary reinforcement signal from the external environment, this internal signal is continuous and bounded. It not only works as one of the input vectors of the critic network but also contributes to the parameter tuning for the critic network. Moreover, it can be adjusted adaptively and automatically by the reference network according the system states and the control action. As this internal signal is not crafted manually or with any prior knowledge, we believe that our proposed dual critic network design can provide useful suggestions about how to adaptively and automatically assign a proper reinforcement signal for the general system to achieve the final goal over time adaptively. We would like to note that, although there are many successful applications on tracking control with adaptive/fuzzy controllers [41]–[43], many such approaches require an accurate system model to be able to achieve the control or tracking performance. However, in real-world complex problems, it is not uncommon that such an accurate system model is very difficult, or even impossible sometimes, to obtain due to the complexity of the system. Therefore, we consider our proposed dual critic model-free ADP approach to be particularly useful for such complex systems.

Motivated by our previous work [7], [44], we extend our idea to the tracking control problems with several major new contributions. First, we integrate a tracking filter in this paper to handle the tracking problem rather than the balancing problem. Second, we develop the Lyapunov stability analysis to provide the theoretical support of our method, which is important to understand the convergence of this method. Third, we develop a virtual reality (VR) platform to demonstrate the performance of our method, with active interaction of the external environment. Finally, we conduct many more experiments, including two types of noises/disturbances, for the system to demonstrate its performance. We would like to note that the terminology of "reference network" we use in this paper is similar to the "goal network" and "goal generator network" we discussed in several of our recent papers [15], [45], and [25]. The underlying principle of both the reference network and the goal network/goal generator network is the same, i.e., to provide a detailed internal goal representation to facilitate learning, optimization, and control.

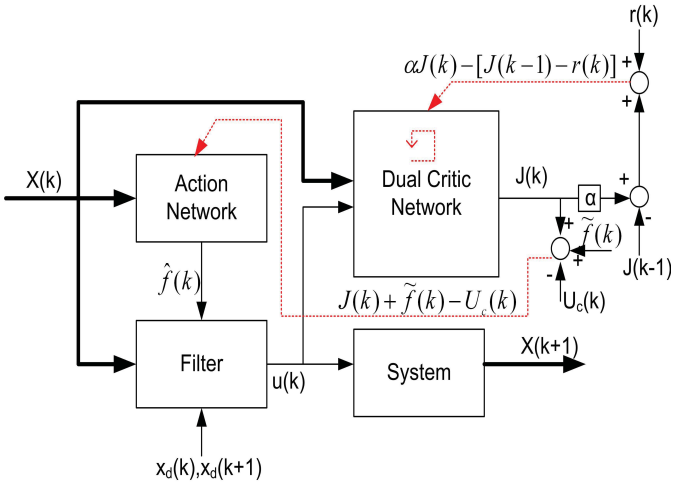


Fig. 1. Architecture design of the proposed HDP approach with a tracking filter.

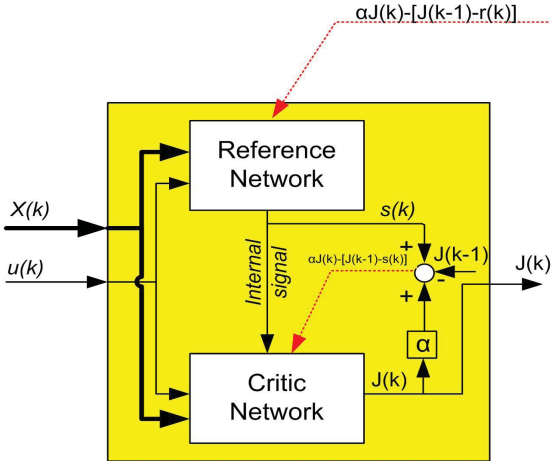


Fig. 2. Description of the dual critic network.

The rest of this paper is organized as follows. Section II provides the architecture description of the dual critic network and the tracking filter design. Section III presents the implementation of the dual critic network HDP design with a tracking filter, together with its associated learning algorithms. Detailed experiment setup and simulation results are given in Sections IV and V, respectively. For both cases, we provide comparative studies between our approach and the typical HDP approach under the same settings. Detailed Lyapunov stability analysis is presented in Section VI. Finally, Section VII concludes this work with some discussions on future research directions. The detailed pseudo-code for the implementation of our method is given in the Appendix.

II. HDP With a Dual Critic Network for Nonlinear Tracking Control

The schematic diagram of our proposed idea is presented in Fig. 1. The action network is kept the same as in [13], [21]. For the critic network, we integrate with one reference network, and therefore there are two networks in the critic

network block as presented in Fig. 2. The tracking filter is added here to show the performance on the tracking control problem. In the rest of this section, we will introduce the dual critic network block and the tracking filter.

A. Dual Critic Network in ADP Design

The motivation of this dual critic network design is twofold: one is to provide an internal goal representation for the critic network; the other is to help approximate the cost-to-go in detail since the internal reinforcement signal works as one of the input vectors for the critic network.

From the system-level view in Fig. 1, we can see that the parameters in the dual critic network block can not only be tuned by an external signal but also be adjusted by itself. Specifically, the reference network in the top of the block is tuned by the error function with the external reward signal, while the critic network at the bottom of the block is tuned by the error function with the internal reinforcement signal. As presented in Fig. 2, the reference network observes a regular reward signal $r(k)$ (usually a binary value) from external environment and provides the critic network with a detailed internal reinforcement signal $s(k)$ (usually a continuous value) by justifying the system state vectors $X(k)$ and control action $u(k)$. In order to approximate the value function $J(k)$ well, the critic network keeps the same inputs as the reference network in addition to the internal signal $s(k)$. Moreover, $s(k)$ also contributes to the error function of the critic network, as the dash line shown inside the block. Since the $s(k)$ can be automatically adjusted according to the state vectors $X(k)$ and the control action $u(k)$, we regard it as an adaptive reinforcement signal. In summary, the key idea of our dual critic design is to use the reference network to automatically and adaptively generate the internal goal signal, rather than hand-crafted in the traditional ADP approaches, to guide the decision-making process for the optimal action at any time instance to accomplish the final goal. This reference network can also actively interact with the critic network and action network, either directly (for critic network) or indirectly (for action network), to support the action selection in a principled way. We will further present the detailed learning architecture and algorithm for such a dual structure design in Section III-B.

B. Tracking Filter

In order to demonstrate the improvement of our proposed dual-critic controller, we would like to test it on nonlinear tracking control problem with a tracking filter, as presented in Fig. 1. The inner structure of the tracking filter is presented in Fig. 3, which was originally from [46] and later developed in [21] and [22]. To be clear, we introduce the design of the tracking filter as follows.

The nonlinear system function is defined in a general form as

$$x(k+1) = f(x(k)) + u(k) + d(k) \tag{1}$$

where $f(x(k))$ is the nonlinear system function, $x(k) = [\,x_1(k)\ x_2(k)\ x_3(k)\ \cdots\ x_n(k)\,]$, and $x_i(k) \in \Re$ is the state value for the ith dimension at time instance k. $u(k)$ is the control action, and $d(k)$ is the disturbance bounded in $[-d_m, d_m]$, where d_m is a constant value.

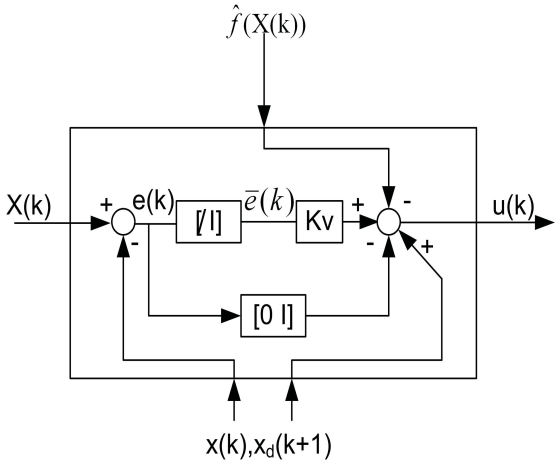


Fig. 3. Description of the tracking filter.

The nonlinear system function $f(x(k))$ is assumed to be unknown in the simulation and can be approximated by the action network here. The approximation value is denoted as $\hat{f}(x(k))$, which works as one of the inputs of the filter. Moreover, the inputs of the filter also include the current state vector $X(k)$ and the desired trajectory value $x_d(k)$ and $x_d(k+1)$, as presented in Fig. 3. The error $e(k)$ is defined as the difference between the current state value and the desired value as follows:

$$e(k) = x(k) - x_d(k) \tag{2}$$

and the filtered tracking error $\bar{e}(k)$ is defined as

$$\bar{e}(k) = e(k) + \lambda_1 e_{n-1}(k) + \cdots + \lambda_{n-1} e_1(k) \tag{3}$$

where $e_{n-1}(k), \ldots, e_1(k)$ are the past error values, which means that $e_{n-i}(k) = e_n(k-i)$, $i = 0, 1, \ldots, n-1$, $n \in \Re$. For brevity, we define $\wedge = [\lambda_{n-1}, \lambda_{n-2}, \ldots, \lambda_1]$, where $\lambda_i \in \Re$. Therefore, (3) can be rewritten as

$$\bar{e}(k) = [\wedge \quad I]e(k). \tag{4}$$

Again, we can rewrite (3) for time instance $k+1$ as

$$\begin{aligned}\bar{e}(k+1) = {} & e(k+1) + \lambda_1 e_{n-1}(k+1) + \cdots \\ & + \lambda_{n-1} e_1(k+1).\end{aligned} \tag{5}$$

Substituting (1) into (5), we get

$$\begin{aligned}\bar{e}(k+1) = {} & f(x(k)) - x_d(k+1) + \lambda_1 e_{n-1}(k+1) + \cdots \\ & + \lambda_{n-1} e_1(k+1) + u(k) + d(k) \\ = {} & f(x(k)) - x_d(k+1) + \lambda_1 e_n(k) + \cdots \\ & + \lambda_{n-1} e_2(k) + u(k) + d(k).\end{aligned} \tag{6}$$

Similar to [21] and [22], we define that the control sequence

$$\begin{aligned}u(k) = {} & x_d(k+1) - \hat{f}(x(k)) + k_v \bar{e}(k) - \lambda_1 e_n(k) - \\ & \cdots - \lambda_{n-1} e_2(k).\end{aligned} \tag{7}$$

Substituting (7) into (6), we get

$$\bar{e}(k+1) = K_v \bar{e}(k) - \tilde{f}(x(k)) + d(k) \tag{8}$$

where $\tilde{f}(x(k))$ is the nonlinear system function approximation error given by

$$\tilde{f}(x(k)) = f(x(k)) - \hat{f}(x(k)) \tag{9}$$

and K_v is the gain value. Assuming that $\tilde{f}(x(k))$ is bounded, the system will be stable if $0 < K_{\text{vmax}} < 1$, where K_{vmax} is the maximum eigenvalue of K_v [22].

III. Implementation of HDP With Dual Critic Network for Nonlinear Tracking Control

In this section, we provide detailed procedures on how to implement our proposed dual critic network HDP approach with a tracking filter. We also define the external reinforcement signal and the internal reinforcement signal here. A brief pseudocode (Algorithm 1) is provided in this section to clearly demonstrate the implementation of the dual critic network. Moreover, a detailed algorithm-level implementation (Algorithm 2) is presented in the Appendix to show the implementation of our proposed approach. Multilayer perceptron (MLP) neural network has been one of the most popular techniques to approximate the nonlinear function in the ADP community [11], [27], [12]. Therefore, we follow this trend and adopt MLP in our ADP design as well.

A. External and Internal Reinforcement Signal

As mentioned previously, there are two types of reinforcement signal in our proposed approach. One is the external reinforcement signal, which comes from the environment, and the other is internal reinforcement signal, which comes from the reference network and works as an internal goal that guides the system's behavior specifically. We will discuss these two signals in detail in the following.

External reinforcement signal $r(k)$ is defined according to the current filtered tracking error $\bar{e}(k)$ as

$$r(k) = \left[r_1(k)\ r_2(k)\ , \ldots ,\ r_m(k) \right] \in \Re^m \tag{10}$$

with

$$r_i(k) = \begin{cases} 0, & if\ \|\bar{e}_i(k)\| \le c \\ -1, & if\ \|\bar{e}_i(k)\| \ge c \end{cases}, \qquad i = 1, 2, 3, \ldots, m \tag{11}$$

where $\|\cdot\|$ represents the Euclidean vector 2-norm and c is the constant threshold for the filtered tracking error. The binary value 0 represents a good tracking performance while -1 means a poor one.

The internal reinforcement signal $s(k)$ is the output of the reference network bounded in $[-1, 1]$ and can be adaptively adjusted as the system states change. At the feed-forward stage, the internal signal works as one of the inputs for the critic network. At the feed-backward stage, the parameters in the critic network are tuned by the error function with $s(k)$. Therefore, the internal reinforcement signal $s(k)$ closely connects the reference network and the critic network as a whole block.

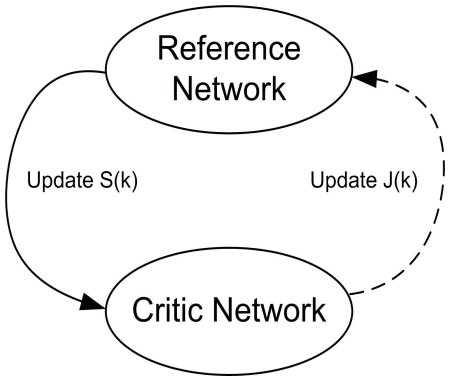


Fig. 4. Learning schematic in dual critic network.

B. Learning and Optimization of Dual Critic Network

Compared to the typical HDP design in [13], the learning and optimization of the critic network here is associated with the reference network as presented in Fig. 4. At the forward stage, the reference network obtains the inputs of the state vector $X(k)$ and the control action $u(k)$ and provides an internal reinforcement signal $s(k)$ for the critic network. Then the critic network updates the cost-to-go signal $J(k)$. At the backward stage, the reference network will first be tuned by the error function (15) with $r(k)$ and the updated cost-to-go signal $J(k)$. After this is done, the reference network will provide the updated internal reinforcement signal $s(k)$ for the critic network, which will then be adjusted with the error function (12). The learning process will repeat until the terminal conditions are satisfied. The pseudocode in Algorithm 1 shows exactly this learning procedure.

The error function of the critic network is defined as follows:

$$e_c(k) = \alpha J(k) - [J(k-1) - s(k)]; \quad E_c(k) = \frac{1}{2} e_c^2(k) \tag{12}$$

where

$$J(k) = \omega_c^{(1)}(k) \cdot \phi\left(\omega_c^{(2)}(k) \cdot x_c(k)\right) \tag{13}$$

and $\omega_c^{(1)}(k)$ and $\omega_c^{(2)}(k)$ refer to the weights of the input to the hidden layer and the hidden to the output layer in critic network, respectively. $x_c(k)$ is the input vector of the critic network and it contains the state vector $X(k)$, the control action $u(k)$, and the internal signal $s(k)$. ϕ stands for the sigmoid function that refines the output into the range of $[-1, 1]$. $s(k)$ is defined as

$$s(k) = \phi\left(\omega_r^{(1)}(k) \cdot \phi\left(\omega_r^{(2)}(k) \cdot x_r(k)\right)\right) \tag{14}$$

where $\omega_r^{(1)}(k)$ and $\omega_r^{(2)}(k)$ refer the weights of the input to the hidden layer and the hidden to the output layer in reference network, respectively. $x_r(k)$ is the input vector of the reference network and it contains the state vector $X(k)$ and the control action $u(k)$.

The error function of the reference network is defined as

$$e_r(k) = \alpha J(k) - [J(k-1) - r(k)]; \quad E_r(k) = \frac{1}{2} e_r^2(k). \tag{15}$$

Algorithm 1 Outline of Implementation of Dual Critic Network HDP

$/ * \mathbf{s} \Leftarrow RefNet(\mathbf{x}, \mathbf{u}, \mathbf{w}_r)$, internal goal representation with the reference network;
$RefNet$: the reference network;
$\mathbf{w}_r$: weights of the $RefNet$;
$\mathbf{s}$: internal goal signal;

$\mathbf{J} \Leftarrow CritNet(\mathbf{x}, \mathbf{u}, \mathbf{s}, \mathbf{w}_c)$, total cost-to-go signal approximated by the critic network;
$CritNet$: the critic network;
$\mathbf{w}_c$: weights of the $CritNet$;
$\mathbf{J}$:total cost-to-go signal, the output of the critic network; */

1) **while** $\neg TerminalContion$ **do**
2) Initiate $\mathbf{x}$, $\mathbf{w}_r$, $\mathbf{w}_c$;
3) **repeat**
4) Obtain the updated action $\mathbf{u}$ and apply to the system;
5) Obtain the updated state $\mathbf{X}$ and immediate external reward r;
6) Update the error function E_r based on (15);
7) Employ backpropagation rules (20)-(23) to minimize E_r till the $TerminalContion1$;
8) Update the error function E_c based on (12);
9) Employ backpropagation rules (27)-(30) to minimize E_c till the $TerminalContion2$;
10) **until** $CurrentState \notin threshold$
11) **end while**

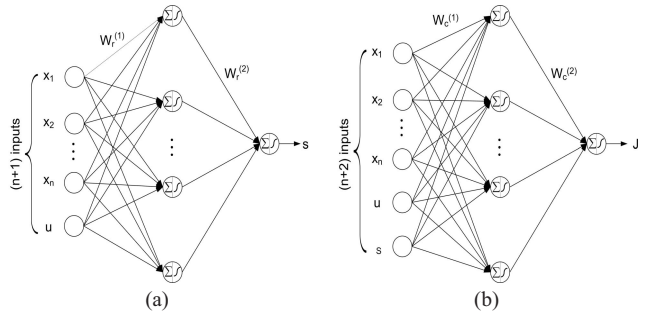


Fig. 5. (a) Neural network structure of the reference network. (b) Neural network structure of the critic network.

Given that the state vector $X(k)$ has n elements and the control action $u(k)$ is a single control unit, the inputs for the reference network and the critic network will be $(n+1)$ and $(n+2)$, as presented in Fig. 5(a) and (b), respectively. The chain backpropagation rule is employed for the neural networks to learn and adapt their weights.

1) *Reference Network*: The reference network is introduced here to provide an internal goal representation for the critic network. The internal goal $s(k)$ is defined as

$$s(k) = \frac{1 - \exp^{-l(k)}}{1 + \exp^{-l(k)}} \tag{16}$$

$$l(k) = \sum_{i=1}^{N_{rh}} w_{r_i}^{(2)}(t) y_i(k) \tag{17}$$

$$y_i(k) = \frac{1 - exp^{-z_i(k)}}{1 + exp^{-z_i(k)}}, \quad i = 1, \ldots, N_{rh} \tag{18}$$

$$z_i(k) = \sum_{j=1}^{n+1} w_{r_{i,\ j}}^{(1)}(t) x_{rj}(k), \quad i = 1, \ldots, N_{rh} \tag{19}$$

where z_i is the input of the ith hidden node and y_i is the corresponding output of this hidden node after the sigmoid function, l is the input to the output node, N_{rh} is the number of hidden neurons in the reference network, and x_{rj} is the input vector of the reference network, which has $(n+1)$ input nodes as presented in Fig. 5(a).

The procedure of backpropagation rule applied to the reference network is illustrated below.
1) $\Delta w_r^{(2)}$: Reference network weights adjustment from the hidden layer to the output layer

$$\Delta w_{r_i}^{(2)}(k) = \eta_r(k) \left[-\frac{\partial E_r(k)}{\partial w_{r_i}^{(2)}(k)} \right] \tag{20}$$

where $\eta_r(k)$ is the learning rate of the reference network at time instance k, and

$$\begin{aligned} \frac{\partial E_r(k)}{\partial w_{r_i}^{(2)}(k)} &= \frac{\partial E_r(k)}{\partial J(k)} \frac{\partial J(k)}{\partial s(k)} \frac{\partial s(k)}{\partial l(k)} \frac{\partial l(k)}{\partial w_{r_i}^{(2)}(k)} \\ &= \alpha e_r(k) \cdot \frac{1}{2}(1 - (s(k))^2) \cdot y_i(k) \\ &\quad \cdot \sum_{i=1}^{N_{rh}} [w_{c_i}^{(2)}(k) \frac{1}{2}(1 - p_i^2(k)) w_{c_{i,\ n+2}}^{(1)}(k)]. \end{aligned} \tag{21}$$

2) $\Delta w_r^{(1)}$: Reference network weight adjustment from the input layer to the hidden layer

$$\Delta w_{r_{i,\ j}}^{(1)}(k) = \eta_r(k) \left[-\frac{\partial E_r(k)}{\partial w_{r_{i,\ j}}^{(1)}(k)} \right] \tag{22}$$

$$\begin{aligned} \frac{\partial E_r(k)}{\partial w_{r_{i,\ j}}^{(1)}(t)} &= \frac{\partial E_r(k)}{\partial J(k)} \frac{\partial J(k)}{\partial s(k)} \frac{\partial s(k)}{\partial l(k)} \frac{\partial l(k)}{\partial y_i(k)} \frac{\partial y_i(k)}{\partial z_i(k)} \frac{\partial z_i(k)}{\partial w_{r_{i,\ j}}^{(1)}(k)} \\ &= \alpha e_r(k) \cdot \frac{1}{2}(1 - y_i^2(k)) \cdot x_{rj}(k) \cdot \frac{1}{2}(1 - (s(k))^2) \\ &\quad \cdot w_{r_i}^{(2)}(k) \cdot \sum_{i=1}^{N_{rh}} \left[w_{c_i}^{(2)}(k) \frac{1}{2}(1 - p_i^2(k)) w_{c_{i,\ n+2}}^{(1)}(k) \right]. \end{aligned} \tag{23}$$

Once the internal goal $s(k)$ is updated in reference network, we can adapt the weight tuning in the critic network.

2) *Critic Network*: In the literature, the critic network is applied to approximate the cost function, and its inputs normally contain the state vector $X(k)$ and the control unit $u(k)$. Here we add one more input with the internal goal $s(k)$ and hope that $s(k)$ can provide the critic network with detailed goal representation that contributes to the system's decision making. The cost-to-go signal $J(k)$ is defined as

$$J(k) = \sum_{i=1}^{N_{ch}} w_{c_i}^{(2)}(k) p_i(k) \tag{24}$$

$$p_i(k) = \frac{1 - exp^{-q_i(k)}}{1 + exp^{-q_i(k)}}, \quad i = 1, \ldots, N_{ch} \tag{25}$$

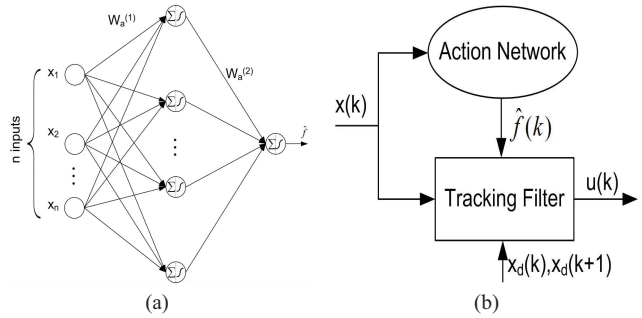


Fig. 6. (a) Neural network structure of the action network. (b) Control action generator.

$$q_i(k) = \sum_{j=1}^{n+2} w_{c_i,\ j}^{(1)}(k) x_{cj}(k), \qquad i = 1, \ldots, N_{ch} \tag{26}$$

where q_i and p_i are the input and output of the ith hidden node in the critic network, respectively, and x_{cj} is input vector of the critic network with $n+2$ nodes, as presented in Fig. 5(b).

The procedure of the backpropagation rule applied to the critic network is provided in the following.

1) $\Delta w_c^{(2)}$: Critic network weight adjustment from the hidden layer to the output layer

$$\Delta w_{c_i}^{(2)}(k) = \eta_c(k) \left[-\frac{\partial E_c(k)}{\partial w_{c_i}^{(2)}(k)} \right] \tag{27}$$

where $\eta_c(k)$ is the learning rate of the critic network at time instance k

$$\frac{\partial E_c(k)}{\partial w_{c_i}^{(2)}(k)} = \frac{\partial E_c(k)}{\partial J(k)} \frac{\partial J(k)}{\partial w_{c_i}^{(2)}(k)} = \alpha e_c(k) \cdot p_i(k). \tag{28}$$

2) $\Delta w_c^{(1)}$: Critic network weight adjustment from the input layer to the hidden layer

$$\Delta w_{c_i,\ j}^{(1)}(k) = \eta_c(k) \left[-\frac{\partial E_c(k)}{\partial w_{c_i,\ j}^{(1)}(k)} \right] \tag{29}$$

$$\begin{aligned} \frac{\partial E_c(k)}{\partial w_{c_i,\ j}^{(1)}(k)} &= \frac{\partial E_c(t)}{\partial J(k)} \frac{\partial J(k)}{\partial p_i(k)} \frac{\partial p_i(k)}{\partial q_i(k)} \frac{\partial q_i(k)}{\partial w_{c_i,\ j}^{(1)}(k)} \\ &= \alpha e_c(k) \cdot w_{c_i}^{(2)}(k) \cdot \frac{1}{2}(1 - p_i^2(k)) x_{cj}(k). \end{aligned} \tag{30}$$

C. Interaction of the Action Network and the Tracking Filter

The control action in this paper is generated by the tracking filter, as shown in Fig. 6(b). The action network here is to approximate the nonlinear system function $f(x(k))$, and the approximation error $\tilde{f}(x(k))$ is added to the error function of the action network. Note that $\tilde{f}(x(k))$ cannot be obtained directly from (9) since $f(x(k))$ is assumed unknown in this paper. The model network is commonly used in the ADP/ACD designs for tracking control [18], [26]. Here we apply the same technique to predict the state vector $\hat{x}(k+1)$ and get $\hat{\tilde{e}}(k+1)$. Then, $\tilde{f}(x(k))$ can be obtained from (8).

The error function of the action network is defined as follows:

$$e_a = J(k) + \tilde{f}(k); \qquad E_a(k) = \frac{1}{2} e_a^2(k). \tag{31}$$

The nonlinear system function $\hat{f}(x(k))$ can be obtained as follows:

$$\hat{f}(x(k)) = \frac{1 - \exp(-\upsilon(k))}{1 + \exp(-\upsilon(k))} \tag{32}$$

$$\upsilon(k) = \sum_{i=1}^{N_{ah}} w_{a_i}^{(2)}(k) g_i(k) \tag{33}$$

$$g_i(k) = \frac{1 - \exp(-h_i(k))}{1 + \exp(-h_i(k))}, \quad i = 1, \ldots, N_{ah} \tag{34}$$

$$h_i(k) = \sum_{j=1}^{n} w_{a_{i,j}}^{(1)}(k) x_{aj}(k), \quad i = 1, \ldots, N_{ah} \tag{35}$$

where h_i and g_i are the input and output of the ith hidden node in the action network, and υ is the input for the output node. $\hat{f}$ is the output of the action network, and N_{ah} is the total number of hidden nodes in the action network. And x_{aj} is the input vector of the action network presented in Fig. 6(a). Note that the weights tuning of the action network should consider the tracking filter as well.

1) $\Delta w_a^{(2)}$: Action network weights adjustment from the hidden layer to the output layer

$$\Delta w_{a_i}^{(2)}(k) = \eta_a(k) \left[-\frac{\partial E_a(k)}{\partial w_{a_i}^{(2)}(k)} \right] \tag{36}$$

where $\eta_a(k)$ is the learning rate of the action network at time instance k. And

$$\begin{aligned} \frac{\partial E_a(k)}{\partial w_{a_i}^{(2)}(k)} &= \frac{\partial E_a(k)}{\partial J(k)} \frac{\partial J(k)}{\partial u(k)} \frac{\partial u(k)}{\partial \upsilon(k)} \frac{\partial \upsilon(k)}{\partial w_{a_i}^{(2)}(k)} \\ &= e_a(k) \left[-\frac{1}{2} \left(1 - \hat{f}^2(k) \right) \right] g_i(k) \end{aligned} \tag{37}$$

$$\cdot \left\{ \sum_{i=1}^{N_{ch}} w_{c_i}^{(2)}(k) \left[\frac{1}{2} \left(1 - p_i^2(k) \right) \right] w_{c_{i,(n+1)}}^{(1)}(k) \right\}. \tag{38}$$

2) $\Delta w_a^{(1)}$: Action network weights adjustment from the input layer to the hidden layer

$$\Delta w_{a_{i,\ j}}^{(1)}(k) = \eta_a(k) \left[-\frac{\partial E_a(k)}{\partial w_{a_{i,\ j}}^{(1)}(k)} \right] \tag{39}$$

$$\begin{aligned} \frac{\partial E_a(k)}{\partial w_{a_{ij}}^{(1)}(k)} &= \frac{\partial E_a(k)}{\partial J(k)} \frac{\partial J(k)}{\partial u(k)} \frac{\partial u(k)}{\partial \upsilon(k)} \frac{\partial \upsilon(k)}{\partial g_i(k)} \\ &\quad \frac{\partial g_i(k)}{\partial h_i(k)} \frac{\partial h_i(k)}{\partial w_{a_{ij}}^{(1)}(k)} \end{aligned} \tag{40}$$

$$\begin{aligned} &= \sum_{l=1}^{N_{ah}} \left(w_{c_l}^{(2)}(k) w_{c_{l,(n+1)}}^{(1)}(k) \left[\frac{1}{2} \left(1 - p_l^2(k) \right) \right] \right) e_a(k) \\ &\times \left\{ w_{a_i}^{(2)}(k) x_{aj}(k) \left[\frac{1}{2} \left(1 - \hat{f}^2(k) \right) \right] \left[\frac{1}{2} \left(1 - g_i^2(k) \right) \right] \right\}. \end{aligned} \tag{41}$$

TABLE I
SUMMARY OF THE PARAMETERS USED IN THE SIMULATION STUDY ONE

Para.	η_c	η_a	η_r	Kv	λ	c	*
Value	$5e-3$	$8e-3$	$2.5e-3$	0.1	0.2	$1.5e-3$	*
Para.	N_c	N_a	N_r	T_c	T_a	T_r	α
Value	40	150	50	$1e-4$	$1e-5$	$1e-5$	0.95

We note that, similar to [13] and [27], the normalization of the weights will be employed during the learning and adaptation for all the networks used here. The weights are confined to the proper range by

$$\mathbf{w}_r(k+1) = \frac{\mathbf{w}_r(k) + \Delta\mathbf{w}_r(k)}{a}, \quad \left\{a = \max(|a_{ij}|) \,|\, \forall a_{ij} \in \mathbf{w}_r(k) + \Delta\mathbf{w}_r(k)\right\} \tag{42}$$

$$\mathbf{w}_c(k+1) = \frac{\mathbf{w}_c(k) + \Delta\mathbf{w}_c(k)}{b}, \quad \left\{b = \max(|b_{ij}|) \,|\, \forall b_{ij} \in \mathbf{w}_c(k) + \Delta\mathbf{w}_c(k)\right\} \tag{43}$$

$$\mathbf{w}_a(k+1) = \frac{\mathbf{w}_a(k) + \Delta\mathbf{w}_a(k)}{c}, \quad \left\{c = \max(|c_{ij}|) \,|\, \forall c_{ij} \in \mathbf{w}_a(k) + \Delta\mathbf{w}_a(k)\right\}. \tag{44}$$

IV. SIMULATION STUDY ONE

In this section, we present two numerical simulations based on the same system function. The motivation is to compare the tracking performance with our proposed approach and with the typical approach in [21], which is originally from [13]. The system function is defined by the general nonlinear form as

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= f(x(k)) + u(k) + d(k) \end{aligned} \tag{45}$$

where

$$f(x(k)) = -\frac{4}{11} \cdot \left(\frac{x_1}{1+x_2^2}\right) + \frac{2}{5}x_2(k) \tag{46}$$

with $f(x(k))$ assumed to be unknown in the tracking process. Instead, the approximation value $\hat{f}(x(k))$ can be obtained from the action network. And $d(k)$ is the disturbance here.

A. Example One

The objective of Example 1 is to track the sinusoidal signal with some harmonic signal with x_2. The desired signal function is defined as $x_{2d} = \sin(\omega kT)\cos(2\omega kT + \tau)$, where $\omega = 0.2$ rad/s and $\tau = \pi/2$. We set the sample interval $T = 50$ ms and the total simulation time to be 150 s. In the literature, noise or disturbance is normally added in the simulation to see how robustness of the proposed approach will be, like in [22], [47], and [48]. In this paper, we adopt a similar technique as in [22]. Disturbance $d(k) = 1.5$ is introduced at $k = 1200$, corresponding to $t = 60$ s. Otherwise, it is set to be 0. The input vector X is defined as $X_{in}(k) = [e(k-1) \quad e(k) \quad x_{2d}(k-1) \quad x_{2d}(k)]$, where $e(k) = x_2(k) - x_{2d}(k)$. She structures of the action network, the reference network, and the critic network are $4-4-1$ (i.e., the network has four input nodes, four hidden nodes and one output node), $5-4-1$, and $6-4-1$, respectively. The parameters we used in the simulation are summarized as in Table I.

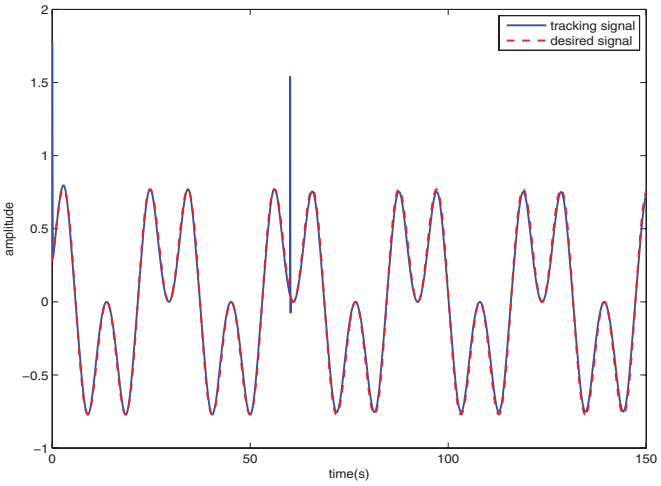


Fig. 7. Typical tracking performance with our proposed approach.

where

α : Discount factor.
c : Threshold on mean square error.
η_c : Initial learning rate of the critic network.
η_a : Initial learning rate of the action network.
η_r : Initial learning rate of the reference network.
N_c : Internal cycle of the critic network.
N_a : Internal cycle of the action network.
N_r : Internal cycle of the reference network.
T_c : Internal training error threshold for the critic network.
T_a : Internal training error threshold for the action network.
T_r : Internal training error threshold for the reference network.

We note that the learning rates will drop once the tracking performance is "good" over time. Specifically, we will compare the mean square error (MSE) as expressed in (47) with a certain threshold c

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x - x_d)^2 \tag{47}$$

where x is the state vector, x_d is the desired tracking signal, and N is a preset integer.

If the MSE is less than the threshold, then the learning rate will be divided by a certain number and the new threshold can be calculated by dividing another number correspondingly. This kind of evaluation will be repeated during the whole process of the tracking control. The detailed implementation is presented in Algorithm 2 (line 24 to 32).

For a comparative study, we have conducted the simulation with both approaches under the same parameters and environment settings. The weights in the neural networks used in both approaches are randomly selected from $[-1, 1]$. The starting point of the state vector is $(0, 1.5)$, which is the same for both approaches. The typical tracking performance with our proposed approach and the typical HDP approach are

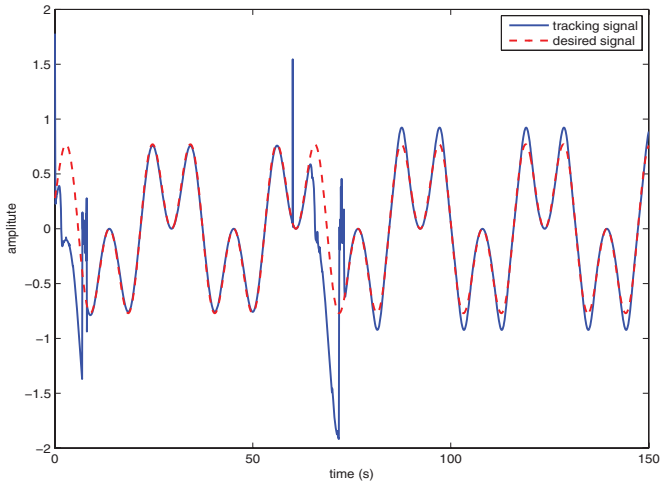


Fig. 8. Typical tracking performance with HDP approach.

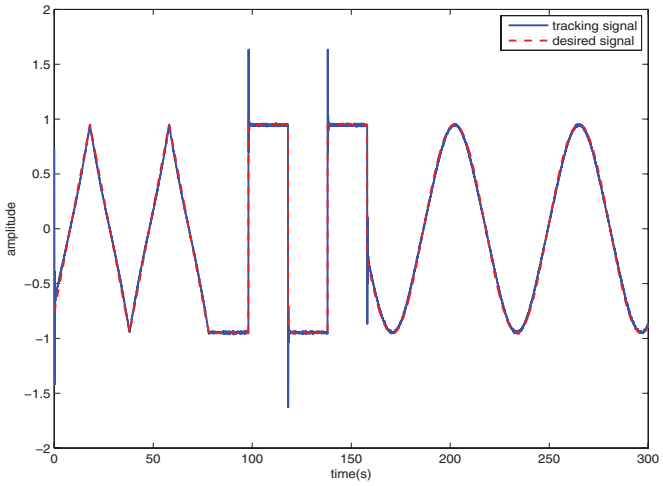


Fig. 9. Typical tracking performance with our proposed approach.

presented in Figs. 7 and 8, respectively. From Fig. 7, we can see that the tracking signal (solid line) can exactly follow the desired signal (dash line) within 1 s. In addition, the tracking signal can also quickly go back to the right track after the disturbance at 60 s. On the other hand, the tracking signal with the typical HDP approach can only follow the desired signal after 10 s. Moreover, the tracking signal needs more time (13 s) to go back to track the desired signal after the disturbance. From this example, we can see that our proposed approach shows not only faster learning process than the typical HDP approach but also better robustness against disturbance.

B. Example Two

In order to show the adaptiveness of the proposed approach, we conducted another numerical example to track a signal that would change from a saw signal to a square signal and finally to a sinusoid signal. We used the same system function and environment settings as in Section IV-A, except that $d(k)$ was set to be white Gaussian noise with a standard deviation of 0.005 for all k. The object is to track the desired signal with the state vector x_2. The desired tracking signal is defined as

$$x_{2d} = \begin{cases} A \cdot \left(1 - \left|t - \frac{T_0}{2}\right|\right), & 0 < t < T_0 \\ A \cdot \left(1 - \left|t - \frac{3T_0}{2}\right|\right), & T_0 < t < 2T_0 \\ -A, 2T_0 < t < 3T_0 \text{ or } 4T_0 < t < 5T_0 \\ A, 3T_0 < t < 4T_0 \text{ or } 5T_0 < t < 6T_0 \\ A \cdot \sin(\omega t), 6T_0 < t < 7.5T_0 \end{cases} \quad (48)$$

where $t = kT$, k is the step number, and T is the sample time ($T = 50$ ms). $A = 0.95$ is the amplitude of the signal. $T_0 = 40$ s is taken as the time internal for which each signal lasts (i.e., the signal will change after T_0).

The difficulty of this task is that the controller needs to learn to track the desired signal, which will change over time, under the white Gaussian noise. Fig. 9 shows the typical tracking performance with our proposed approach, and one can clearly see the good transient tracking performance when the signal changes. But for the typical HDP approach, we

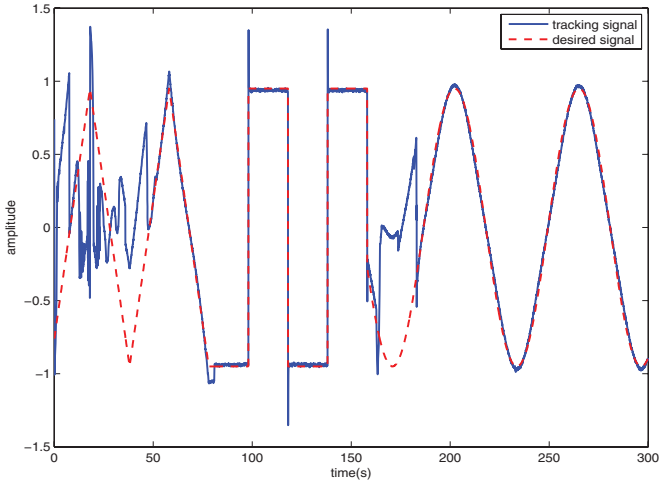


Fig. 10. Typical tracking performance with HDP approach.

can see in Fig. 10 that the controller takes about 50 s to learn to follow the saw signal and also spends much time to learn when the desired signal changes to a rectangular signal. In addition, the controller also requires about 20 s to catch up after the desired signal changes to sinusoid signal. This indicates that our proposed approach shows better adaptiveness to this tracking problem than the typical HDP approach under noisy condition.

V. Simulation Study Two

Instead of testing on two numerical cases, here we would like to evaluate our proposed approach on a continuous benchmark, i.e., the ball-and-beam tracking problem [49]. There are many versions of this benchmark, and in this paper we adopt the model presented in Fig. 11. The system contains a long beam that can be tilted by a servo or electric motor with a ball rolling back and forth on top of the beam. In this system, the driver is located at the center of the beam. The angle of the beam to the horizontal axis is measured by an incremental encoder, and the position of the ball can be obtained with cameras mounted on the top of system. Our proposed approach

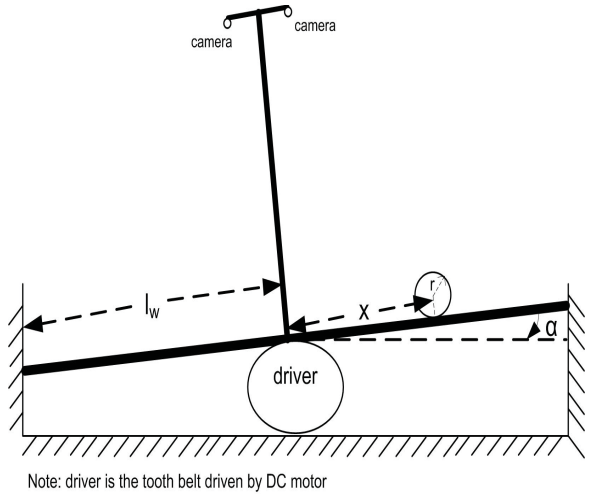


Fig. 11. Schematic diagram of the ball-and-beam system.

will learn to track the desired signal with the position of the ball.

A. Problem Formulation

From [49], we can obtain the equations of motion from the Lagrange equation as follows:

$$\left(m+\frac{I_b}{r^2}\right)\ddot{x}'+(mr^2+I_b)\frac{1}{r}\ddot{\alpha}-mx'\dot{\alpha}^2=mg(\sin\alpha) \quad (49)$$

$$[m(x')^2+I_b+I_\omega]\ddot{\alpha}+(2m\dot{x}'x'+bl^2)\dot{\alpha}+Kl^2\alpha + (mr^2+I_b)\frac{1}{r}\ddot{x}'-mgx'(\cos\alpha)=ul(\cos\alpha) \quad (50)$$

where

$m = 0.0162$ kg, is the mass of the ball;

$r = 0.02$ m, is the roll radius of the ball;

$I_b = 4.32\times10^{-5} kg\cdot m^2$, is the inertia moment of the ball;

$b = 1\ N_s/m$, is the friction coefficient of the drive mechanics;

$l = 0.48$ m, is the radius of force application;

$l_\omega = 0.5$ m, is the radius of the beam;

$K = 0.001N/m$, the stiffness of the drive mechanics;

g : 9.8 N/kg, is the gravity;

$I_\omega = 0.14025$ kg$\cdot$m^2, is the moment of inertia of the beam; and

u is the force of the drive mechanics.

In order to simplify the system model function, we define that $x_1 = x'$ represents the position of the ball, $x_2 = \dot{x}'$ represents the velocity the ball, $x_3 = \alpha$ is the angle of the beam with respect to the horizontal axis, and $x_4 = \dot{\alpha}$ is the angular velocity of the beam. Therefore, the state vector can be defined as $X = [x_1\ x_2\ x_3\ x_4]$. In this way, the system function (49) and (50) can be transformed into the following forms:

$$\left(m+\frac{I_b}{r^2}\right)\dot{x}_2+(mr^2+I_b)\frac{1}{r}\dot{x}_4=mx_1x_4^2+mg(\sin x_3) \quad (51)$$

$$(mr^2+I_b)\frac{1}{r}\dot{x}_2+[mx_1^2+I_b+I_\omega]\dot{x}_4=(ul+mgx_1)\times\cos x_3-(2mx_2x_1+bl^2)x_4-Kl^2x_3. \quad (52)$$

TABLE II
SUMMARY OF THE PARAMETERS USED IN THE SIMULATION STUDY TWO

Para.	η_c	η_a	η_r	$K\upsilon$	λ	c	*
Value	0.02	0.01	0.05	0.05	0.1	$5e-5$	*
Para.	N_c	N_a	N_r	T_c	T_a	T_r	α
Value	100	200	200	$1e-4$	$1e-5$	$1e-5$	0.95

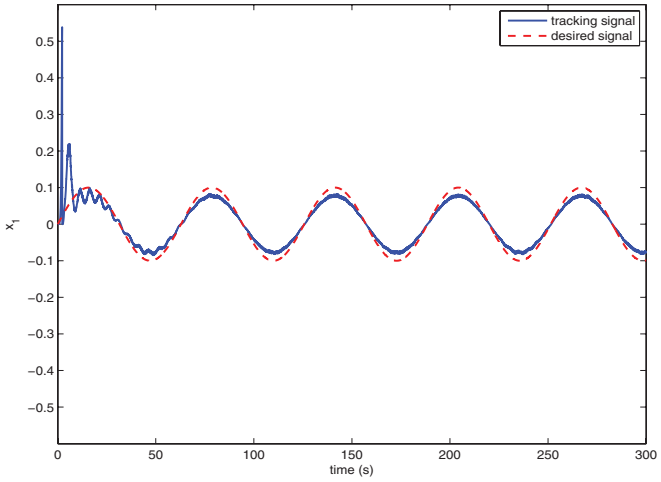


Fig. 12. Typical tracking performance with our proposed approach.

To make it clearer, we rewrite (51) and (52) with the specific value of all the parameters mentioned above into the approximate nonlinear state-space equations as follows:

$$\dot{x}_1 = x_2 \quad (53)$$

$$\dot{x}_2 = 1.717\sin(x_3) \quad (54)$$

$$\dot{x}_3 = x_4 \quad (55)$$

$$\dot{x}_4 = -0.241x_4+0.157x_1\cos(x_3)+0.5\cos(x_3)\cdot u. \quad (56)$$

The objective is to track the sinusoid signal $x_{1d} = 0.1\sin(\omega t)$ with the position of ball (x_1), where $\omega = 0.1$. This task requires the controller to not only keep the balance of the ball on the beam but also to track the desired signal using the position of the ball (x_1). That is to say, if x_1 is out of the bound ($[-0.48, 0.48]$ m), or x_3 exceeds the angular velocity tolerance ($[0.24, 0.24]$ rad/s), we will reset the ball to the initial starting point ($[0\ 0\ 0\ 0]$). Since the learning process of the neural network is continuous, we will assume that the weights can be carried on when the task is reset. Keen readers may also find that this is a continuous-time benchmark rather the discrete-time case above. Here we apply the common technique that is extensively used in the literature [13], [27], [44] of calling the continuous system model function with the *ode*45 function in MATLAB with the step size 0.02 s. The parameters used in this case are summarized in Table II, and 5% uniform noise is also added to the sensor of x_1 to show the tracking performance under noisy conditions.

Figs. 12 and 13 present the tracking performance with our proposed approach and the typical HDP approach [21], respectively. Fig. 12 clearly shows that the ball is out of bounds at the very beginning, but can quickly track the desired signal within one period. The oscillation of the tracking signal (solid line) in the first period shows the learning process of the

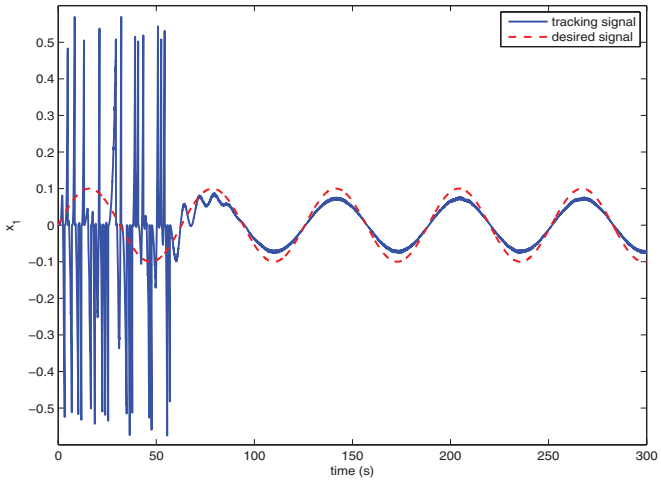


Fig. 13. Typical tracking performance with HDP approach.

TABLE III
SUMMARY OF TOTAL TRACKING ERROR FOR THE SIMULATION CASES

Number	Example 1	Example 2	Ball and Beam
DualCritHDP	7.454	9.292	3.761
HDP	223.9	712.8	54.49

controller with our proposed approach. In Fig. 13, we can see that the ball is out of bounds many times before it can track the desired signal. In other words, the controller spends the whole first period to learn to control the ball and it learns after about 80 s. The simulation results show that the controller with our proposed approach has better noise tolerance than with the typical HDP approach.

To provide a more accurate assessment of the tracking performance, here we summarize the quantitative measurements in terms of tracking error for all examples in study 1 (i.e., examples 1 and 2) and study 2 (i.e., the ball and beam benchmark). Here we adopt the evaluation function in [18] defined as PER $= \sum_0^N e^T(k)e(k)$, where $e(k)$ is the tracking error in Fig. 3 and N refers to the number of steps in the simulation. The PER for Sections IV-A, IV-B, and V-A are summarized in Table III.

From Table III, one can see that our proposed dual critic HDP can achieve much lower PER (total tracking error) than with the typical HDP approach [21]. The results also confirm that our proposed structure with the informative and adaptive reinforcement signal can outperform the typical HDP structure in terms of tracking accuracy.

B. VR Demonstration of the Dual Critic Network Design

In this paper, we further apply our algorithm on VR environment to show real-time simulation of our approach during interaction with the external environment. VR can enable powerful human–computer interactions, and it is interesting to observe how the proposed algorithm works in real-time simulation without the requirement of setting up the real physical system. Here we would like to demonstrate the tracking performance of our proposed approach on the ball-and-beam benchmark [50], [51]. And we also add disturbances to see how robust our proposed approach can perform.

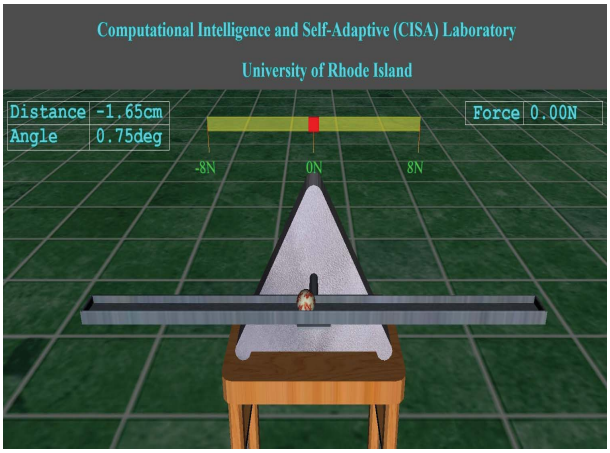


Fig. 14. Schematic of the VR platform.

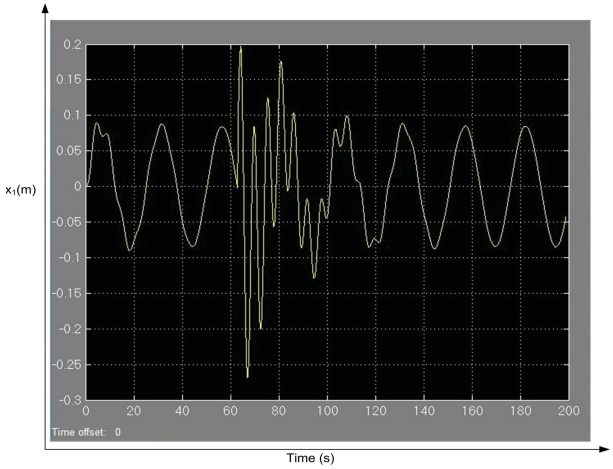


Fig. 15. Typical tracking performance with our proposed approach in VR/simulation platform.

The VR platform is developed as in Fig. 14, where one can see that the ball-and-beam system is at the center of the scene and the state vectors (x_1 and x_3) are displayed in the upper-left table of the figure. In order to be more realistic, we add a disturbance option (upper center) in the simulation. That is to say, the user can apply any disturbance between -8 to 8 N on the ball whenever needed, and the corresponding force will be applied to the system, as also displayed on the upper-right table.

Real-time simulation result on the tracking problem with our proposed approach is presented in Fig. 15, where one can clearly see the online learning process. In other words, the first period of signal is like a distorted sine wave, while the signal in second period almost tracks the desired sine signal. At about 64 s, we added a 2.1 N disturbance on the ball and we can see that the controller spends about two periods to learn to get back to the right track.

VI. LYAPUNOV STABILITY ANALYSIS

We now proceed to study the stability characteristics of our proposed method. Similar to [52], we employ ϕ_a for the output of the hidden neurons in the action network s.t. $\phi_a(k) = [\phi_{a,1}(k)\ \phi_{a,2}(k) \cdots \phi_{a,N_{ah}}(k)]^T$ and then

$\|\phi_a(k)\|^2 = \phi_a^T(k)\phi_a(k)$. For the critic and reference networks, we adopt similar notations s.t. ϕ_c and ϕ_r are the output of the hidden neurons in the critic network and reference network, respectively. The estimated weights in the networks are denoted as $\hat{\omega}_a$, $\hat{\omega}_c$, and $\hat{\omega}_r$, while the expected weights are denoted as ω_a, ω_c, and ω_r. Therefore, the differences are defined as $\tilde{\omega}_a = \hat{\omega}_a - \omega_a$, $\tilde{\omega}_c = \hat{\omega}_c - \omega_c$, and $\tilde{\omega}_r = \hat{\omega}_r - \omega_r$. We note that, although $\hat{\omega}_a$, $\hat{\omega}_c$, and $\hat{\omega}_r$ are unknown parameters, we assume they have upper bounds in this analysis because, when they exceed the preset upper bounds, we will normalize the weights according to (42)–(44).

Define the Lyapunov function candidate as

$$V = V_1 + V_2 + V_3 + V_4 + V_5 \tag{57}$$

where

$$V_1 = \frac{1}{\gamma_1}\bar{e}^T(k)\bar{e}(k) \tag{58}$$

$$V_2 = \frac{1}{\eta_c} tr(\tilde{\omega}_c^T(k)\tilde{\omega}_c(k)) \tag{59}$$

$$V_3 = \frac{1}{2}\|\zeta_c(k-1)\|^2 \tag{60}$$

$$V_4 = \frac{1}{\gamma_2\eta_a} tr(\tilde{\omega}_a^T(k)\tilde{\omega}_a(k)) \tag{61}$$

$$V_5 = \frac{1}{\gamma_3\eta_r} tr(\tilde{\omega}_r^T(k)\tilde{\omega}_r(k)). \tag{62}$$

In (60), $\zeta_c(k-1) = (\hat{\omega}_c(k-1) - \omega_c)^T\phi_1(k-1) = \tilde{\omega}_c^T(k-1)$ $\phi_1(k-1)$ and $\gamma_i > 0$ for $i = 1, 2, 3$.

The first difference of the Lyapunov function candidate can be written as

$$\Delta V = \Delta V_1 + \Delta V_2 + \Delta V_3 + \Delta V_4 + \Delta V_5. \tag{63}$$

For ΔV_1, we have

$$\bar{e}(k+1) = K_v\bar{e}(k) - \tilde{f}(x(k)) + d(k) \tag{64}$$

where

$$\begin{aligned}
\tilde{f}(x(k)) &= f(x(k)) - \hat{f}(x(k)) \\
f(x(k)) &= \omega_a^T(k)\phi_a(k) + \varepsilon_a(x(k)) \\
\tilde{f}(x(k)) &= (\hat{\omega}_a(k) - \omega_a(k))^T\phi_a(k) + \varepsilon_a(x(k)) \\
&= \tilde{\omega}_a^T(k)\phi_a(k) + \varepsilon_a(x(k)).
\end{aligned} \tag{65}$$

Therefore

$$\begin{aligned}
\Delta V_1 &= \frac{1}{\gamma_1}\left(\bar{e}^T(k+1)\bar{e}(k+1) - \bar{e}^T(k)\bar{e}(k)\right) \\
&= \frac{1}{\gamma_1}\big[\left(K_v\bar{e}(k) + \zeta_a(k) + \varepsilon_a(k) + d(k)\right)^T \\
&\quad \cdot (K_v\bar{e}(k) + \zeta_a(k) + \varepsilon_a(k) + d(k)) - \bar{e}^T(k)\bar{e}(k)\big] \\
&\le \frac{1}{\gamma_1}\big[3\left(K_v^2\|\bar{e}(k)\|^2 + \|\zeta_a(k)\|^2 + \|\varepsilon_a(k) + d(k)\|^2\right) \\
&\quad - \|\bar{e}(k)\|^2\big] \\
&\le \frac{3}{\gamma_1}\Bigg(\left(K_{\text{vmax}}^2 - \frac{1}{3}\right)\|\bar{e}(k)\|^2 + \|\zeta_a(k)\|^2 \\
&\quad + \|\varepsilon_a(k) + d(k)\|^2\Bigg)
\end{aligned} \tag{66}$$

where K_{vmax} is the maximum eigenvalue of K_v.

For ΔV_2, we have

$$\begin{aligned}
\hat{\omega}_c(k+1) = \hat{\omega}_c(k) - \eta_c\alpha\phi_c(k)(\alpha\hat{\omega}_c^T(k)\phi_c(k) + s(k) \\
-\hat{\omega}_c^T(k-1)\phi_c(k-1))^T
\end{aligned} \tag{67}$$

and the corresponding $\tilde{\omega}_c(k+1)$ can be expressed as

$$\begin{aligned}
\tilde{\omega}_c(k+1) &= \tilde{\omega}_c(k) - \eta_c\alpha\phi_c(k)\big(\alpha\hat{\omega}_c^T(k)\phi_c(k) + s(k) \\
&\quad -\hat{\omega}_c^T(k-1)\phi_c(k-1)\big)^T \\
&= \left(I - \eta_c\alpha^2\phi_c(k)\phi_c^T(k)\right)\tilde{\omega}_c(k) - \eta_c\alpha\phi_c(k) \\
&\quad \left(\alpha\omega_c^T\phi_c(k) + s(k) - \hat{\omega}_c^T(k-1)\phi_c(k-1)\right)^T.
\end{aligned} \tag{68}$$

Then

$$\begin{aligned}
\Delta V_2 &= \frac{1}{\eta_c} tr(\tilde{\omega}_c^T(k+1)\tilde{\omega}_c(k+1) - \tilde{\omega}_c^T(k)\tilde{\omega}_c(k)) \\
&= \frac{1}{\eta_c} tr\big(\tilde{\omega}_c^T(k)A^TA\tilde{\omega}_c(k) - \tilde{\omega}_c^T(k)\tilde{\omega}_c(k) \\
&\quad + B\alpha^2\eta_c^2\phi_c^T(k)\phi_c(k)B^T - \tilde{\omega}_c^T(k)A^T\eta_c\alpha\phi_c(k)B^T \\
&\quad - B\eta_c\alpha\phi_c^T(k)A\tilde{\omega}_c(k)\big)
\end{aligned} \tag{69}$$

where $A = I - \eta_c\alpha^2\phi_c(k)\phi_c^T(k)$ and $B = \alpha\omega_c^T\phi_c(k) + s(k) - \hat{\omega}_c^T(k-1)\phi_c(k-1)$.

We note

$$\begin{aligned}
\tilde{\omega}_c^T(k)A^TA\tilde{\omega}_c(k) &- \tilde{\omega}_c^T(k)\tilde{\omega}_c(k) \\
&= \tilde{\omega}_c^T(k)[(I - \eta_c\alpha^2\phi_c\phi_c^T)^T \\
&\quad \times (I - \eta_c\alpha^2\phi_c\phi_c^T)]\tilde{\omega}_c(k) - \tilde{\omega}_c^T(k)\tilde{\omega}_c(k) \\
&= -\eta_c\alpha^2\|\zeta_c(k)\|^2 - \eta_c\alpha^2\tilde{\omega}_c^T(k)\phi_c\phi_c^T \\
&\quad \times (I - \eta_c\alpha^2\phi_c\phi_c^T)\tilde{\omega}_c(k)
\end{aligned} \tag{70}$$

where $\zeta_c(k) = \tilde{\omega}_c^T(k)\phi_c(k)$.

Therefore

$$\begin{aligned}
\Delta V_2(k) = &-\alpha^2\|\zeta_c(k)\|^2 - \alpha^2\left(1 - \eta_c\alpha^2\|\phi_c(k)\|^2\right)\|\zeta_c(k)\|^2 \\
&+\eta_c\alpha^2\|\phi_c(k)\|^2 \\
&\cdot \left\|\alpha\omega_c^T\phi_c(k) + s(k) - \hat{\omega}_c^T(k-1)\phi_c(k-1)\right\|^2 \\
&-2tr[\alpha\left(I - \eta_c\alpha^2\|\phi_c(k)\|^2\right)\zeta_c(k) \\
&\cdot\left(\alpha\omega_c^T\phi_c(k) + s(k) - \hat{\omega}_c^T(k-1)\phi_c(k-1)\right)^T].
\end{aligned} \tag{71}$$

We would like to seek the upper bound of (71) by applying Cauchy–Schwarz inequality for the fourth term. Therefore, we have

$$\begin{aligned}
\Delta V_2(k) \le &-\alpha^2\|\zeta_c(k)\|^2 - \alpha^2(1 - \eta_c\alpha^2\|\phi_c(k)\|^2)\cdot\|\zeta_c(k)\|^2 \\
&+ \left\|\alpha\omega_c^T\phi_c(k) + s(k) - \hat{\omega}_c^T(k-1)\phi_c(k-1)\right\|^2 \\
&-\alpha^2(1 - \eta_c\alpha^2\|\phi_c(k)\|^2) \\
&\cdot \left\|\zeta_c(k) + \omega_c^T\phi_c(k) + \alpha^{-1}s(k) - \alpha^{-1}\hat{\omega}_c^T(k-1)\right. \\
&\left.\times\phi_c(k-1)\right\|^2.
\end{aligned} \tag{72}$$

From (72), we can further get

$$\begin{aligned}\Delta V_2(k) \le & -\alpha^2 \|\zeta_c(k)\|^2 + \frac{1}{2}\|\zeta_c(k-1)\|^2 \\ & +2\left\|\alpha\omega_c^T\phi_c(k) + s(k) - \hat{\omega}_c^T(k-1)\phi_c(k-1)\right\|^2 \\ & -\alpha^2\left(1-\eta_c\alpha^2\|\phi_c(k)\|^2\right) \\ & \cdot\left\|\zeta_c(k) + \omega_c^T\phi_c(k) + \alpha^{-1}s(k) - \alpha^{-1}\hat{\omega}_c^T(k-1)\right. \\ & \left.\times\phi_c(k-1)\right\|^2. \end{aligned} \tag{73}$$

For $\Delta V_3(k)$, we have

$$\Delta V_3(k) = \frac{1}{2}(\|\zeta_c(k)\|^2 - \|\zeta_c(k-1)\|^2). \tag{74}$$

For $\Delta V_4(k)$, we have

$$\begin{aligned}\hat{\omega}_a(k+1) = & \hat{\omega}_a(k) - \eta_a\phi_a(k)\hat{\omega}_c^T(k)C_a(k) \\ & \cdot\left(\hat{\omega}_c^T(k)\phi_c(k) + K_v\bar{e}(k) - \bar{e}(k+1)\right)^T \end{aligned} \tag{75}$$

where $C_a(k)$ is an $N_{ch}\times 1$ vector and its elements can be defined as $C_a(k) = (1/2)(1-\phi_c(k))\omega_{c,n+1}(k)$. And then

$$\begin{aligned}\tilde{\omega}_a(k+1) = & \tilde{\omega}_a(k) - \eta_a\phi_a(k)\hat{\omega}_c^T(k)C_a(k) \\ & \cdot\left(\hat{\omega}_c^T(k)\phi_c(k) + K_v\bar{e}(k) - \bar{e}(k+1)\right)^T. \end{aligned} \tag{76}$$

With tracking error dynamics, we can rewrite (76) as

$$\begin{aligned}\tilde{\omega}_a(k+1) = & \tilde{\omega}_a(k) - \eta_a\phi_a(k)\hat{\omega}_c^T(k)C_a(k) \\ & \cdot\left(\hat{\omega}_c^T(k)\phi_c(k) + \zeta_a(k) - \varepsilon_a(k) - d(k)\right)^T. \end{aligned} \tag{77}$$

Therefore, we have

$$\begin{aligned}\Delta V_4(k) = & \frac{1}{\gamma_2\eta_a}tr\left[\tilde{\omega}_a^T(k+1)\tilde{\omega}_a(k+1) - \tilde{\omega}_a^T(k)\tilde{\omega}_a(k)\right] \\ = & \frac{1}{\gamma_2}tr\Big[-2\hat{\omega}_c^T(k)C_a(k)\zeta_a(k) \\ & \cdot\left(\hat{\omega}_c^T(k)\phi_c^T(k) + \zeta_a(k) - \varepsilon_a(k) - d(k)\right)^T \\ & +\eta_a\phi_a^2(k)\cdot\left\|\hat{\omega}_c^T(k)C_a(k)\right\|^2 \\ & \cdot\left\|\hat{\omega}_c^T(k)\phi_c(k) + \zeta_a(k) - \varepsilon_a(k) - d(k)\right\|^2\Big]. \end{aligned} \tag{78}$$

In order to further simplify the formula, we can rewrite (78) as follows:

$$\begin{aligned}\Delta V_4(k) = & \frac{1}{\gamma_2}tr[-(1-\eta_a\phi_a^2(k))\|D\|^2\|E\|^2 \\ & +\|D\cdot E - \zeta_a(k)\|^2 - \|\zeta_a(k)\|^2] \end{aligned} \tag{79}$$

where

$$D = \hat{\omega}_c^T(k)C_a(k) \tag{80}$$

$$E = \hat{\omega}_c^T(k)\phi_c(k) + \zeta_a(k) - \varepsilon_a(k) - d(k). \tag{81}$$

Applying the Cauchy–Schwarz inequality for (79), we have

$$\Delta V_4(k) \le \frac{1}{\gamma_2}tr[F + 2\cdot\|D\cdot E\|^2 + \|\zeta_a(k)\|^2] \tag{82}$$

where

$$F = -(1-\eta_a\|\phi_a(k)\|^2)\|D\|^2\|E\|^2. \tag{83}$$

For $\Delta V_5(k)$, we have

$$\begin{aligned}\tilde{\omega}_r(k+1) = & \tilde{\omega}_r(k) - \eta_r\phi_r(k)\hat{\omega}_c^T(k)C_r(k)(\alpha\hat{\omega}_c^T(k)\phi_c(k) \\ & +r(k) - \hat{\omega}_c^T(k-1)\phi_c(k-1))^T \end{aligned} \tag{84}$$

where $C_r(k)$ is an $N_{ch}\times 1$ vector and its elements can be defined as $C_r(k) = (1/2)(1-\phi_c(k))\omega_{c,n+2}(k)$.

Therefore

$$\begin{aligned}\Delta V_5(k) = & \frac{1}{\gamma_3\eta_r}tr\left[\tilde{\omega}_r^T(k+1)\tilde{\omega}_r(k+1) - \tilde{\omega}_r^T(k)\tilde{\omega}_r(k)\right] \\ = & \frac{1}{\gamma_3}tr\Big[-2\hat{\omega}_r^T(k)C_r(k)\zeta_r(k)(\alpha\hat{\omega}_c^T(k)\phi_c(k) \\ & +r(k) - \hat{\omega}_c^T(k-1)\phi_c(k-1))^T \\ & +\eta_r\phi_r(k)\left\|\hat{\omega}_r^T(k)C_r(k)\right\|^2\|\alpha\hat{\omega}_c^T(k)\phi_c(k) \\ & +r(k) - \hat{\omega}_c^T(k-1)\phi_c(k-1)\|^2\Big]. \end{aligned} \tag{85}$$

We can also simplify (85) by rewriting it as

$$\Delta V_5(k) = \frac{1}{\gamma_3}tr[H + \|G\cdot I - \zeta_r(k)\|^2 - \|\zeta_r(k)\|^2] \tag{86}$$

where

$$G = (\alpha\hat{\omega}_c^T(k)\phi_c(k) + r(k) - \hat{\omega}_c^T(k-1)\phi_c(k-1))^T \tag{87}$$

$$H = -(1-\eta_r\|\phi_r(k)\|^2)\|G\|^2\|I\|^2 \tag{88}$$

$$I = \hat{\omega}_r^T(k)C_r(k). \tag{89}$$

Applying the Cauchy–Schwarz inequality for (86), we have

$$\Delta V_5(k) \le \frac{1}{\gamma_3}tr\left[H + 2\cdot\|G\cdot I\|^2 + \|\zeta_r(k)\|^2\right]. \tag{90}$$

Substituting (66), (73), (74), (82), and (90) into (57), we can get the first difference of the Lyapunov function candidate as

$$\begin{aligned}\Delta V(k) \le & \frac{3}{\gamma_1}\left(\left(K_{\text{vmax}}^2 - \frac{1}{3}\right)\|\bar{e}(k)\|^2 + \|\zeta_a(k)\|^2 + \|\varepsilon_a(k)\right. \\ & \left.+d(k)\|^2\right) - \alpha^2\|\zeta_c(k)\|^2 + 2\|\alpha\omega_c^T\phi_c(k) + s(k) \\ & -\hat{\omega}_c^T(k-1)\phi_c(k-1)\|^2 - \alpha^2\left(1-\eta_c\alpha^2\|\phi_c(k)\|^2\right) \\ & \cdot\|\zeta_c(k) + \omega_c^T\phi_c(k) + \alpha^{-1}s(k) \\ & -\alpha^{-1}\hat{\omega}_c^T(k-1)\phi_c(k-1)\|^2 \\ & +\frac{1}{2}\|\zeta_c(k-1)\|^2 + \frac{1}{2}(\|\zeta_c(k)\|^2 - \|\zeta_c(k-1)\|^2) \\ & +\frac{1}{\gamma_2}tr\left[F + \left(2\cdot\|D\cdot E\|^2\right) + \|\zeta_a(k)\|^2\right] \\ & +\frac{1}{\gamma_3}tr\left[H + \left(2\cdot\|G\cdot I\|^2\right) + \|\zeta_r(k)\|^2\right]. \end{aligned} \tag{91}$$

In order to express (91) in a clear way, we rewrite it as

$$\begin{aligned}\Delta V(k) \le & \frac{3}{\gamma_1}\left(\left(K_{vmax}^2-\frac{1}{3}\right)\|\bar{e}(k)\|^2\right.\\ &\left.+\|\zeta_a(k)\|^2+\|\varepsilon_a(k)+d(k)\|^2\right)\\ &-\left(\alpha^2-\frac{1}{2}\right)\|\zeta_c(k)\|^2-\alpha^2\left(I-\eta_c\alpha^2\|\phi_c(k)\|^2\right)\\ &\cdot\|\zeta_c(k)+\omega_c^T\phi_c(k)+\alpha^{-1}s(k)\\ &-\alpha^{-1}\hat{\omega}_c^T(k-1)\phi_c(k-1)\|^2\\ &-\frac{1}{\gamma_2}(1-\eta_a\|\phi_a(k)\|^2)\|D\|^2\|E\|^2\\ &-\frac{1}{\gamma_3}(1-\eta_r\|\phi_r(k)\|^2)\|G\|^2\|I\|^2+Z^2 \end{aligned} \tag{92}$$

where Z^2 is defined as

$$\begin{aligned}Z^2=&\frac{1}{\gamma_2}\|\zeta_a(k)\|^2+\frac{1}{\gamma_3}\|\zeta_r(k)\|^2\\ &+2\left\|\alpha\omega_c^T\phi_c(k)+s(k)-\hat{\omega}_c^T(k-1)\phi_c(k-1)\right\|^2\\ &+\frac{1}{\gamma_2}\left(2\cdot\|D\cdot E\|^2\right)+\frac{1}{\gamma_3}\left(2\cdot\|G\cdot I\|^2\right).\end{aligned} \tag{93}$$

And the upper bound for Z^2 is

$$\begin{aligned}Z_2\le&\frac{2}{\gamma_2}\left\|\omega_{am}^T\phi_{am}\right\|^2+\frac{2}{\gamma_3}\left\|\omega_{rm}^T\phi_{rm}\right\|^2\\ &+6(\alpha^2+1)\cdot\left\|\omega_{cm}^T\phi_{cm}\right\|^2+6s_m^2+\frac{6}{\gamma_2}\left\|\omega_{cm}^TC_{am}\right\|^2\\ &\cdot\left(\left\|\omega_{cm}^T\phi_{cm}\right\|^2+2\left\|\omega_{am}^T\phi_{am}\right\|^2+\|\varepsilon_{am}+d_m\|^2\right)\\ &+\frac{6}{\gamma_3}\left\|\omega_{rm}^TC_{rm}\right\|^2\cdot\left((\alpha^2+1)\cdot\left\|\omega_{cm}^T\phi_{cm}\right\|^2+\|r_m\|^2\right)\\ =&\,6\cdot\left(\alpha^2+1+\frac{1}{\gamma_2}\left\|\omega_{cm}^TC_{am}\right\|^2\right.\\ &\left.+\frac{1}{\gamma_3}(\alpha^2+1)\cdot\left\|\omega_{rm}^TC_{rm}\right\|^2\right)\\ &\cdot\left\|\omega_{cm}^T\phi_{cm}\right\|^2+\frac{2}{\gamma_2}\left(1+6\left\|\omega_{cm}^TC_{am}\right\|^2\right)\left\|\omega_{am}^T\phi_{am}\right\|^2\\ &+\frac{2}{\gamma_3}\left\|\omega_{rm}^T\phi_{rm}\right\|^2+6s_m^2+\frac{6}{\gamma_2}\left\|\omega_{cm}^TC_{am}\right\|^2\\ &\cdot\|\varepsilon_{am}+d_m\|^2+\frac{6}{\gamma_3}\left\|\omega_{rm}^TC_{rm}\right\|^2\cdot\|r_m\|^2\\ =&\,Z_m^2\end{aligned} \tag{94}$$

where $\omega_{cm}, \omega_{am}, \omega_{rm}, \phi_{cm}, \phi_{am}, \phi_{rm}, C_{am}, C_{rm}, s_m$, and r_m are the upper bounds of $\omega_c, \omega_a, \omega_r, \phi_c, \phi_a, \phi_r, C_a, C_r, s$, and r, respectively.

Equation (94) further implies that $\Delta V(k) \le 0$ if the following conditions hold:

$$0 < K_{\text{vmax}} < \frac{\sqrt{3}}{3} \tag{95}$$

$$\frac{\sqrt{2}}{2} < \alpha < 1 \tag{96}$$

$$\eta_c\alpha^2\|\phi_c(k)\|^2<1,\quad \eta_a\|\phi_a(k)\|^2<1,\quad \eta_r\|\phi_r(k)\|^2<1 \tag{97}$$

and

$$\|\bar{e}(k)\| > \sqrt{\frac{\gamma_1}{1-3K_{v\max}}}\,\|Z_m\| \tag{98}$$

or

$$\|\zeta_c(k)\| > \sqrt{\frac{1}{(\alpha^2-\frac{1}{2})}}\,\|Z_m\|. \tag{99}$$

According to a standard Lyapunov extension theorem [53], [54], this demonstrates that the auxiliary error and the error in the weights estimates are uniformly ultimately bounded. And this further implies that the weights estimates are bounded correspondingly.

VII. Conclusion

This paper proposed a novel ADP structure that combines one reference network with the original critic network into a dual critic network design. The reference network provided the critic network with an internal goal representation that helped it to approximate the total cost-to-go signal in detail. Unlike the discrete reward signal from the external environment, this internal goal signal can be adjusted adaptively with regard to the system state and the control action. Therefore, past experience or prior knowledge is not a necessity to assign reinforcement signal value here. The dual critic network has a weight-turning path not only from the outside but also from the inside. Compared with the typical HDP design under the same simulation environment settings, our proposed dual critic network HDP can achieve better tracking performance on time-costing and the accumulated tracking error. In addition to various simulation studies and a VR platform development, we also presented detailed theoretical analysis in terms of Lyapunov stability analysis for our method.

As a new ADP approach, there are many interesting future research topics along this direction. For instance, in this paper we only used one reference network in our design. We are extending this dual design to see how it works with multiple reference networks on top of the critic network, similar to the hierarchical neural network structure. Some promising results have been achieved and reported in [39] and [45]. Also, the weight-turning rules in our design are based on the classic chain backpropagation and we are interested to see how much improvement can be achieved if we implemented the LM algorithm or Kalman filter into our neural networks.

Appendix
Pseudo Code

Algorithm 2 Algorithm-Level Implementation of Dual Critic Network of ADP for Tracking Control

/* $\hat{f}(\mathbf{x}) \Leftarrow \mathbf{ActNet}(\mathbf{x}, \mathbf{w_a})$, nonlinear function approximation with the action network;
ActNet: the action network;
$\mathbf{x}$: state vector;
$\mathbf{w}_a$: weights of *ActNet*;
$\hat{f}(\mathbf{x})$: nonlinear system function approximation, the output of *ActNet*;
$\mathbf{u} \Leftarrow \mathit{Filter}\left(\mathbf{x}_d, \mathbf{x}, \hat{f}(\mathbf{x})\right)$, control action calculation;

Filter: the tracking filter;
$\mathbf{x}_d$: the desired reference signal;
$\mathbf{u}$: control action;
$\mathbf{s} \Leftarrow$ *RefNet* $(\mathbf{x}, \mathbf{u}, \mathbf{w}_r)$, internal goal representation with the reference network;
RefNet: the reference network;
$\mathbf{w}_r$: weights of the *RefNet*;
$\mathbf{s}$: internal goal signal;
$\mathbf{J} \Leftarrow$ *CritNet* $(\mathbf{x}, \mathbf{u}, \mathbf{s}, \mathbf{w}_c)$, total cost-to-go signal approximated by the critic network;
CritNet: the critic network;
$\mathbf{w}_c$: weights of the *CritNet*;
$\mathbf{J}$:total cost-to-go signal, the output of the critic network; */
/*Note: the parameters N_r, T_r, η_r, N_c, T_c, η_c, N_a, T_a, and η_a are all defined in Table I; */

Initiate $\mathbf{x(0)}$
Uniformly randomize $\mathbf{w}_a(0)$, $\mathbf{w}_r(0)$, $\mathbf{w}_c(0)$ in $[-1,\ 1]$
$\hat{f}(\mathbf{x}(0)) \Leftarrow$ *ActNet* $(\mathbf{x}(0), \mathbf{w}_a(0))$
$\mathbf{u(0)} \Leftarrow$ *Filter* $\left(\mathbf{x}_d, \mathbf{x(0)}, \hat{f}(\mathbf{x(0)})\right)$
$\mathbf{s}(0) \Leftarrow$ *RefNet* $(\mathbf{x}(0), \mathbf{u(0)}, \mathbf{w}_r(0))$
$J(0) \Leftarrow$ *CritNet* $(\mathbf{x}(0), \mathbf{u(0)}, \mathbf{s(0)}, \mathbf{w}_c(0))$
$Jprev = J(0)$
for 1 to MaxStep **do**;
//weights are carried on through the whole learning process;
CurrentState $\Leftarrow$ $(\mathbf{x(k-1)}, \mathbf{u(k-1)})$; //obtain current state vectors from the external environment
$\mathbf{w}_a(k) = \mathbf{w}_a(k-1)$;
$\mathbf{w}_c(k) = \mathbf{w}_c(k-1)$;
$\mathbf{w}_r(k) = \mathbf{w}_r(k-1)$;
$\hat{f}(\mathbf{x}(k)) \Leftarrow$ *ActNet* $(\mathbf{x}(k), \mathbf{w}_a(k))$;
$\mathbf{u(k)} \Leftarrow$ *Filter* $\left(\mathbf{x}_d, \mathbf{x(k)}, \hat{f}(\mathbf{x(k)})\right)$;
$\mathbf{s}(k) \Leftarrow$ *RefNet* $(\mathbf{x}(k), \mathbf{u(k)}, \mathbf{w}_r(k))$;
$J(k) \Leftarrow$ *CritNet* $(\mathbf{x}(k), \mathbf{u(k)}, \mathbf{s(k)}, \mathbf{w}_c(k))$;
Obtain the tracking error $\bar{e}(k)$ via (3)
if $||\bar{e}(k)|| < c$ **then**
  $r(k) = 0$; // reward
else
  $r(k) = -1$; // punishment
end if //corresponding to step 19
if $Step \geq 100$ **then**
  calculate MSE via equation (47);
  if $MSE <$ *threshold* **then**
    $\eta_r(k)$,$\eta_c(k)$,$\eta_a(k)$ are divided by 6, respectively
    $threshold = threshold/2$; //update
  elseif $MSE \geq 4 * threshold$ **then**
    $\eta_r(k)$,$\eta_c(k)$,$\eta_a(k)$ and *threshold* are reset to the initial values, respectively;
  end if; //corresponding to step 26
end if; //corresponding to step 24
$E_r(k) = \frac{1}{2}(\alpha J(k) - (J(k-1) - r(k)))^2$;
$cyc = 0$;
while $(E_r(k) > T_r \& cyc > N_r)$ **do**
// update the weights recursively;
  $\mathbf{w}_r(k) = \mathbf{w}_r(k) + \Delta\mathbf{w}_r(k)$ via (20) and (23);
// update the $\mathbf{s}(k)$, $J(k)$, $E_r(k)$, cyc correspondingly
  $\mathbf{s}(k) \Leftarrow$ *RefNet* $(\mathbf{x}(k), \mathbf{u(k)}, \mathbf{w}_r(k))$;
  $J(k) \Leftarrow$ *CritNet* $(\mathbf{x}(k), \mathbf{u(k)}, \mathbf{s(k)}, \mathbf{w}_c(k))$;
  $E_r(k) = \frac{1}{2}(\alpha J(k) - (J(k-1) - r(k)))^2$;
  $cyc = cyc + 1$;
end while // online learning of the reference network
$E_c(k) = \frac{1}{2}(\alpha J(k) - (J(k-1) - s(k)))^2$;
$cyc = 0$;
while $(E_c(k) > T_c \& cyc > N_c)$ **do**
  $\mathbf{w}_c(k) = \mathbf{w}_c(k) + \Delta\mathbf{w}_c(k)$ via (27) and (30);
  $J(k) \Leftarrow$ *CritNet* $(\mathbf{x}(k), \mathbf{u(k)}, \mathbf{s(k)}, \mathbf{w}_c(k))$;
  $E_c(k) = \frac{1}{2}(\alpha J(k) - (J(k-1) - s(k)))^2$;
  $cyc = cyc + 1$;
end while // online learning of the critic network
$E_a(k) = \frac{1}{2}(\alpha J(k) + \hat{f}(x(k)) - U_c)^2$;
$cyc = 0$;
while $(E_a(k) > T_a \& cyc > N_a)$ **do**
  $\mathbf{w}_a(k) = \mathbf{w}_a(k) + \Delta\mathbf{w}_a(k)$ via (36) and (41);
// update the $\hat{f}(x(k))$, $u(k)$, $\mathbf{s}(k)$, $J(k)$, $E_r(k)$, cyc correspondingly
  $\hat{f}(\mathbf{x}(k)) \Leftarrow$ *ActNet* $(\mathbf{x}(k), \mathbf{w}_a(k))$;
  $\mathbf{u(k)} \Leftarrow$ *Filter* $\left(\mathbf{x}_d, \mathbf{x(k)}, \hat{f}(\mathbf{x(k)})\right)$;
  $\mathbf{s}(k) \Leftarrow$ *RefNet* $(\mathbf{x}(k), \mathbf{u(k)}, \mathbf{w}_r(k))$;
  $J(k) \Leftarrow$ *CritNet* $(\mathbf{x}(k), \mathbf{u(k)}, \mathbf{s(k)}, \mathbf{w}_c(k))$;
  $E_a(k) = \frac{1}{2}(\alpha J(k) + \hat{f}(x(k) - U_c))^2$;
  $cyc = cyc + 1$;
end while // online learning of the action network
end for //corresponding to step 8.

REFERENCES

[1] F. Lewis and D. Liu, eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Piscataway, NJ, USA: IEEE Press, 2013.

[2] J. Si, A. G. Barto, W. B. Powell, and D. C. Wunsch, eds., *Handbook of Learning and Approximate Dynamic Programming*. Piscataway, NJ, USA: IEEE Press, 2004.

[3] D. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA, USA: Athena Scientific, 1995.

[4] W. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New York, USA: Wiley, 2007.

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

[6] J. Lee and J. Lee, "Approximate dynamic programming strategies and their applicability for process control: A review and future directions," *Int. J. Control Autom. Syst.*, vol. 2, no. 3, pp. 263–278, Sep. 2004.

[7] H. He, *Self-Adaptive Systems for Machine Intelligence*. New York, USA: Wiley, 2011.

[8] P. Werbos, "Reinforcement learning and approximate dynamic programming (RLADP)—foundations, common misconceptions and challenges ahead," in *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. New York, USA: Wiley, 2013, pp. 3–30.

[9] P. J. Werbos, "Intelligence in the brain: A theory of how it works and how to build it," *Neural Netw.*, vol. 22, no. 3, pp. 200–212, Apr. 2009.

[10] P. J. Werbos, "ADP: The key direction for future research in intelligent control and understanding brain intelligence," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 898–900, Aug. 2008.

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

[12] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 39–47, May 2009.

[13] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, Mar. 2001.

[14] R. Enns and J. Si, "Helicopter trimming and tracking control using direct neural dynamic programming," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 929–939, Jul. 2003.

[15] Z. Ni, H. He, D. Zhao, and D. Prokhorov, "Reinforcement learning control based on multi-goal representation using hierarchical heuristic dynamic programming," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2012, pp. 1–8.

[16] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 764–773, May 2002.

[17] J. W. Park, R. G. Harley, and G. K. Venayagamoorthy, "Adaptive-critic-based optimal neurocontrol for synchronous generators in a power system using MLP/RBF neural networks," *IEEE Trans. Ind. Appl.*, vol. 39, no. 5, pp. 1529–1540, Sep.–Oct. 2003.

[18] H. Zhang, Q. Wei, and Y. Luo, "A novel infinte-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 937–942, Aug. 2008.

[19] H. Zhang, L. Cui, X. Zhang, and Y. Luo, "Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2226–2236, Dec. 2011.

[20] H. Zhang, R. Song, Q. Wei, and T. Zhang, "Optimal tracking control for a class of nonlinear discrete-time systems with time delays based on heuristic dynamic programming," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1851–1862, Dec. 2011.

[21] L. Yang, J. Si, K. S. Tsakalis, and A. A. Rodriguez, "Direct heuristic dynamic programming for nonlinear tracking conrol with filtered tracking error," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1617–1622, Dec. 2009.

[22] P. He and S. Jagannathan, "Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 37, no. 2, pp. 425–436, Apr. 2007.

[23] P. He and S. Jagannathan, "Reinforcement-based neuro-output feedback control of nonlinear discrete-time systems with input constraints," *IEEE Trans. Systems Man, Cybern. B, Cybern.*, vol. 35, no. 1, pp. 150–154, Feb. 2005.

[24] S. Jagannathan and P. He, "Neural network-based state feedback control of nonlinear discrete-time system in non-strict feedback form," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 2073–2087, Dec. 2008.

[25] Z. Ni, X. Fang, H. He, D. Zhao, and X. Xu, "Real-time tracking control on adaptive critic design with uniformly ultimately bounded condition," in *Proc. IEEE Symp. Adapt. Dynamic Program. Reinforcement Learn., IEEE Symp. Ser. Comput. Intell.*, Apr. 2013, to be published.

[26] D. Liu, H. Javaherian, O. Kovalenko, and T. Huang, "Adaptive critic learning techniques for engine torque and air-fuel ratio control," *IEEE Trans. Systems Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 988–993, Aug. 2008.

[27] J. Fu, H. He, and X. Zhou, "Adaptive learning and control for MIMO system based on adaptive dynamic programming," *IEEE Trans. Neural Netw.*, vol. 22, no. 7, pp. 1133–1148, Jul. 2011.

[28] J. Fu, H. He, Q. Liu, and Z. Ni, "An adaptive dynamic programming approach for closely-coupled mimo system control," in *Proc. 8th Int. Symp. Neural Netw.*, vol. 6677. Jun. 2011, pp. 1–10.

[29] Z. Ni, H. He, D. V. Prokhorov, and J. Fu, "An online actor-critic learning approach with levenberg-marquardt algorithm," in *Proc. Int. Joint Conf. Neural Netw.*, Aug. 2011, pp. 2333–2340.

[30] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Dual heuristic programming excitation neurocontrol for generators in a multimachine power system," *IEEE Trans. Ind. Appl.*, vol. 39, no. 2, pp. 382–394, Apr.–May 2003.

[31] W. Qiao, G. Venayagamoorthy, and R. Harley, "DHP-based wide-area coordinating control of a power system with a large wind farm and multiple FACTS devices," in *Proc. IEEE Int. Conf. Neural Netw.*, Aug. 2007, pp. 2093–2098.

[32] S. Ray, G. K. Venayagamoorthy, B. Chaudhuri, and R. Majumder, "Comparison of adaptive critics and classical approaches based wide area controllers for a power system," *IEEE Trans. Syst. Man, Cybern., B, Cybern.*, vol. 38, no. 4, pp. 1002–1007, Aug. 2008.

[33] Y. Tang, H. He, Z. Ni, J. Wen, and X. Sui, "Reactive power control of grid-connected wind farm based on adaptive dynamic programming," *Neurocomputing*, to be published.

[34] D. Liu, Y. Zhang, and H. G. Zhang, "A self-learning call admission control scheme for CDMA cellular networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 5, pp. 1219–1228, Sep. 2005.

[35] H. G. Zhang, Y. H. Luo, and D. Liu, "Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1490–1503, Sep. 2009.

[36] F.-Y. Wang, N. Jin, D. Liu, and Q. Wei, "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound," *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 24–36, Jan. 2011.

[37] P. H. Eaton, and D. V. Prokhorov, and D. C. Wunsch, "Neurocontroller alternatives for 'fuzzy' ball-and-beam systems with nonuniform nonlinear friction," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 423–435, Mar. 2000.

[38] M. Fairbank, E. Alonso, and D. V. Prokhorov, "Simple and fast calculation of the second-order gradients for globalized dual heuristic dynamic programming in neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 10, pp. 1671–1676, Oct. 2012.

[39] H. He, Z. Ni, and D. Zhao, "Data-driven learning and control with multiple critic networks," in *Proc. 10th World Congr. Int. Control Autom.*, Jul. 2012, pp. 523–527.

[40] G. G. Lendaris and J. C. Neidhoefer, "Guidance in the use of adaptive critics for control," in *Handbook of Learning and Approximate Dynamic Programming*. Piscataway, NJ, USA: IEEE Press, 2004, pp. 97–118.

[41] H. Wong, M. de Queiroz, and V. Kapila, "Adaptive tracking control using synthesized velocity from attitude measurements," *Automatica*, vol. 37, no. 6, pp. 947–953, Jun. 2001.

[42] G. Tao, S. Joshi, and X. Ma, "Adaptive state feedback and tracking control of systems with actuator failures," *IEEE Trans. Autom. Control*, vol. 46, no. 1, pp. 78–95, Jan. 2001.

[43] H. Zhang and L. Cai, "Nonlinear adaptive control using the fourier integral and its application to cstr systems," *IEEE Trans. Systems, Man, Cybern. B, Cybern.*, vol. 32, no. 3, pp. 367–372, Jun. 2002.

[44] H. He, Z. Ni, and J. Fu, "A three-network architecture for on-line learning and optimization based on adaptive dynamic programming," *Neurocomputing*, vol. 78, no. 1, pp. 3–13, Feb. 2012.

[45] H. He, Z. Ni, and D. Zhao, "Learning and optimization in hierarchical adaptive critic design," in *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Piscataway, NJ, USA: IEEE Press, 2013, pp. 78–95.

[46] F. L. Lewis, S. Jagannathan, and A. Yesildirek, "Neural network control of robot manipulators and nonlinear systems," *Neural Network Robot Control*. New York, USA: Taylor & Francis, 1999, pp. 173–219.

[47] F. Yuan, L. Feldkamp, G. Puskorius, and L. Davis, "A simple solution to the bioreactor benchmark problem by application of Q-learning," in *Proc. World Congr. Neural Netw.*, Washington, DC, USA, Jul. 1995, pp. 326–331.

[48] S. N. Balakrishnan and V. Biega, "Adaptive critic based neural networks for control (low order system applications)," in *Proc. Amer. Control Conf.*, Seattle, WA, USA, Jun. 1995, pp. 335–339.

[49] T. L. Chien, C. C. Chen, Y. C. Huang, and W.-J. Lin, "Stability and almost disturbance decoupling analysis of nonlinear system subject to feedback linearization and feedforward neural network controller," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1220–1230, Jul. 2008.

[50] X. Fang, H. He, Z. Ni, and Y. Tang, "Learning and control in virtual reality for machine intelligence," in *Proc. 3rd Int. Conf. Intell. Control Inf. Process.*, Jul. 2012, pp. 63–67.

[51] [Online]. Available: http://www.youtube.com/watch?v=OeZEDBz6ki0&feature=youtu.be

[52] F. Liu, J. Sun, J. Si, W. Guo, and S. Mei, "A boundedness result for the direct heuristic dynamic programming," *Neural Netw.*, vol. 32, pp. 229–235, Aug. 2012.

[53] J. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.

[54] H. Khalil, *Nonlinear Systems*, 3rd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2001.

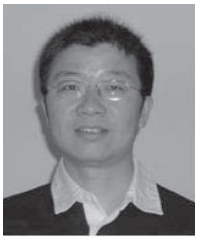
Zhen Ni received the B.S. degrees from the Department of Control Science and Engineering, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2010, and the M.S. degree from the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, USA, in 2012, where he is currently pursuing the Ph.D. degree.

His current research interests include computational intelligence and reinforcement learning, specifically in adaptive dynamic programming and optimal/adaptive control.

Haibo He (SM'11) received the B.S. and M.S. degrees in electrical engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1999 and 2002, respectively, and the Ph.D. degree in electrical engineering from Ohio University, Athens, OH, USA, in 2006.

He was an Assistant Professor with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, USA, from 2006 to 2009. He is currently an Associate Professor with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, USA. His current research interests include adaptive dynamic programming (ADP), machine learning, computational intelligence, hardware design for machine intelligence, and various applications such as smart grid and renewable energy systems. He has authored one research book (Wiley), edited six conference proceedings (Springer), and authored or co-authored over 100 peer-reviewed journal and conference papers. His research has been covered by national and international media such as IEEE Smart Grid Newsletter, The Wall Street Journal, and Providence Business News.

Dr. He is currently an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and the IEEE TRANSACTIONS ON SMART GRID. He was the recipient of the National Science Foundation (NSF) CAREER Award in 2011 and the Providence Business News (PBN) Rising Star Innovator Award in 2011.



Jinyu Wen (M'10) is the Vice Dean of the College of Electrical and Electronic Engineering, Huazhong University of Science and Technology, Wuhan, China. He leads the Smart Grid Operation & Control Research Group, China State Key Laboratory of Advanced Electromagnetic Engineering and Technology. His current research interests include smart grid, power system operation and control, renewable energy systems, and energy storage.

He is an active member of the IEEE Power & Energy Society and Industry Application Society.