Reinforcement Learning Control for a Robotic Manipulator with Unknown Deadzone

Yanan Li
Institute for Infocomm Research
Agency for Science, Technology and Research, Singapore
liy@i2r.a-star.edu.sg

Shengtao Xiao and Shuzhi Sam Ge
Department of Electrical and Computer Engineering
National University of Singapore, Singapore
{*xiao_shengtao, samge*}*@nus.edu.sg*

***Abstract*— In this paper, an actor critic neural network control is developed for a robotic manipulator. Both system uncertainties and unknown deadzone are considered in the tracking control design. Stability of the closed-loop system is analyzed via the Lyapunov's direct method. The critic neural network is used to estimate the cost-to-go and the actor neural network is used to make the cost-to-go converge. Simulation studies are conducted to examine the effectiveness of the proposed actor critic neural network control.**

***Index Terms*— Reinforcement learning, robot control, neural networks, deadzone.**

I. Introduction

Deadzone is an unavoidable problem which has to be dealt with in controlling a robotic system. Neglecting its effects may lead to undesirable performance such as large steady state error, poor transient response and large overshoot [1]. Although adaptive neural/fuzzy control is popular in handling nonliearities [2], [3], the selection of control gains is still a challenging issue. In specific, it is easy to tune the gains by trial and error to obtain a good performance through an off-line process, but these well-tuned gains may not guarantee a desired performance if there are variations of system parameters [4]. In this sense, a method which is independent of prescribed control gains will provide feasibility in practice.

Reinforcement learning is an option to cope with the aforementioned issue [5], [6], [7]. A typical framework of reinforcement learning includes an actor agent which generates a control input to the actuator, and an critic agent to evaluate the cost-to-go at current state with current control policy [8], [9], [10]. The actor agent updates its output based on the value from the critic agent. The ultimate goal is to make the cost-to-go converge to its global optimum. In this paper, we develop an actor critic neural network (NN) control for a n-degrees-of-freedom (n-DOF) robotic manipulator with unknown deadzone and uncertain system dynamics. The designed control will be shown to have adaptability and robustness to uncertain system dynamics with unknown deadzone. The main contributions of this paper include:

(i) without knowing the robotic manipulator's dynamics, the deadzone effect of its actuator, and external disturbance, we propose a control to guarantee a desirable tracking performance;
(ii) the control performance and robustness to uncertainties are enhanced by employing reinforcement learning; and
(iii) the uniform boundedness of the closed-loop system is proved by the Lyapunov's direct method.

The organization of this paper is as follows. Section II presents the problem description and preliminaries on function approximation. Details of the control design are shown in Section III. Simulation study is conducted in Section IV to verify the validity of the proposed control. It is followed by conclusion in Section V.

Notation 1: Given a vector $A \in R^{1\times n}$ and a matrix $B \in R^{n\times n}$, $\|A\|^2 = A^TA$ and $\|B\|^2 = \mathrm{tr}(B^TB)$.

II. Problem Formulation

A. Dynamic Model

The dynamics of a n-DOF rigid robotic manipulator with deadzone can be described as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + f_{dis} = D(\tau) \tag{1}$$

where $q \in R^n$ is the coordinate, $\tau \in R^n$ is the control input (joint torque applied) and $D(\tau)$ is the deadzone function to the control input τ. $M(q) \in R^{n\times n}$ is a symmetric positive definite inertia matrix, $C(q,\dot{q}) \in R^{n\times n}$ represents the centripetal and Coriolis torque, $G(q) \in R^n$ is the gravitational force, and $f_{dis} \in R^n$ represents the external disturbance to the manipulator.

Property 1: [11] The matrix $\dot{M}(q) - 2C(q,\dot{q})$ is skew-symmetric.

Assumption 1: The external disturbance f_{dis} is bounded, i.e., $\|f_{dis}\| \le b_f$, where b_f is a positive constant.

According to Eq. (1), if we let $x_1 = q$ and $x_2 = \dot{q}$, the robot dynamics can be expressed as

$$\dot{x}_1 = x_2, \tag{2}$$
$$\dot{x}_2 = M^{-1}[D(\tau) - f_{dis} - G - Cx_2] \tag{3}$$

The following assumptions are made to facilitate the control design [12].

Assumption 2: As illustrated in Fig. 1, the deadzone nonlinearity can be expressed as

$$D(\tau_i) = \begin{cases} h_{r,i}(\tau_i - b_{r,i}) & \tau_i \geq b_{r,i}; \\ 0 & b_{l,i} < \tau_i < b_{r,i}; \\ h_{l,i}(\tau_i - b_{l,i}) & \tau_i \leq b_{l,i}. \end{cases} \tag{4}$$

where $i = 1, 2, \ldots, n$, $h_{r/l,i}(\cdot)$ is an unknown smooth function, and $b_{r,i} > 0$ and $b_{l,i} < 0$ are known constants.

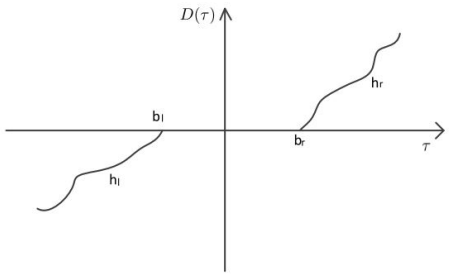


Fig. 1. A deadzone model

Assumption 3: The desired trajectory x_r is continuous and known.

B. Control Objective

A long-term discounted cost function is defined as

$$J(t) = \int_t^{\infty} e^{-\frac{m-t}{\psi}} r(m)\mathrm{d}m \tag{5}$$

where ψ is a time constant for discounting the future cost. $r(t)$ is the instant cost function which is defined as

$$r(t) = (x_1 - x_r)^T Q(x_1 - x_r) + \tau^T R\tau \tag{6}$$

where Q and R are positive definite matrices. The optimal control is achieved when the minimal cost-to-go is obtained.

C. Preliminaries

Radial basis function neural network (RBFNN) is widely used to estimate nonlinear functions due to its good capabilities in function approximation [13]. For a continuous function $h(Z) : R^p \to R$, the following RBFNN is used to approximate it [14]:

$$h_{nn}(Z) = WS(Z) \tag{7}$$

where the input vector $Z \in \Omega \subset R^p$, W is the weight vector such that $W = [w_1, w_2, ..., w_l] \in R^l$ and l is the number of nodes which is greater than 1. $S(Z) = [s_1(Z), s_2(Z), ..., s_l(Z)]^T$ where $s_i(Z)$ can be a Gaussian function as below

$$s_i(Z) = \exp\Big[\frac{-(Z - \mu_i)^T(Z - \mu_i)}{\eta_i^2}\Big], \tag{8}$$

where $i = 1, 2, ..., l, \mu_i = [\mu_{i1}, \mu_{i2}, ..., \mu_{ip}]$ is the center of receptive field, and η_i is the width of the Gaussian function. It is shown that RBFNN (7) is able to estimate any continuous function over a compact set $\Omega_Z \subset R^p$ to any arbitrary accuracy as

$$h(Z) = W^* S(Z) + \epsilon, \ \forall Z \in \Omega_Z \tag{9}$$

where W^* is the optimal constant weight, and ϵ is the estimation error with optimal weights. In particular, the ideal weight vector W^* is defined such that ϵ is minimized for all $Z \in \Omega_Z \subset R^p$, i.e.,

$$W^* \triangleq \arg\min_{W \in R^l} \Big\{ \sup_{Z \in \Omega_Z} |h(Z) - WS(Z)| \Big\} \tag{10}$$

Lemma 1: [13] There exist ideal constant weights W^* such that $|\epsilon| \leq \epsilon^*$ with $\epsilon^* > 0$ for all $Z \in \Omega_Z$.

Lemma 2: [15] A Lyapunov candidate function $V(t)$ is bounded given that initial condition $V(0)$ is bounded, $V(t) \geq 0$ is continuous and the below equation holds:

$$\dot{V}(t) \leq -\kappa V(t) + b \tag{11}$$

where κ and b are two positive constants.

III. Control Design

As illustrated in Fig. 2, two NNs are employed to approximate the desired control and cost-to-go, respectively. The actor network is responsible for generating a control input to the plant which should minimize the cost-to-go and the critic network is to evaluate the current state information and approximate the cost-to-go.

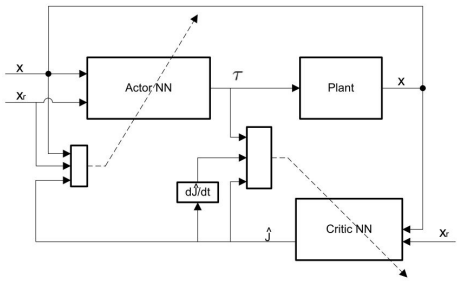


Fig. 2. Structure of reinforcement learning

A. Critic Network

As mentioned above, the critic network is used to approximate the total cost-to-go at current state. Let $J = W_c^* S_c(Z_c) + \epsilon_c$ and $\hat{J} = \hat{W}_c S_c(Z_c)$, where $Z_c = z_1 = x_1 - x_r$.

From the definition (5), the estimation error of the cost-to-go function is [16]:

$$\delta(t) = r(t) - \frac{1}{\psi}\hat{J}(t) + \dot{\hat{J}}(t) \tag{12}$$

As the time constant $\psi \to \infty$, the cost-to-go becomes an infinite horizon problem and Eq. (12) becomes

$$\begin{aligned} \delta(t) &= r(t) + \dot{\hat{J}}(t) \\ &= r(t) + \nabla \hat{J} \dot{Z}_c \end{aligned} \tag{13}$$

which is exactly the same as Hamiltonian defined in [5], [6], [7]. ∇ represents the gradient along Z_c.

Let $E_c = \frac{1}{2}\delta^T\delta$. The updating law for the critic network is designed as $\dot{\hat{W}}_c = -\sigma_c \frac{\partial E_c}{\partial \hat{W}_c}$, more exactly,

$$\begin{aligned} \dot{\hat{W}}_c &= -\sigma_c \delta(t)[-\frac{1}{\psi}\frac{\partial \hat{J}}{\partial \hat{W}_c} + \frac{\partial}{\partial \hat{W}_c}(\frac{\partial \hat{J}}{\partial Z_c}\dot{Z}_c)] \\ &= -\sigma_c(r(t) + \hat{W}_c^T\Lambda)\Lambda \end{aligned} \tag{14}$$

where $\sigma_c > 0$ is the learning rate to the critic neural network and $\Lambda = -\frac{S_c}{\psi} + \nabla S_c \dot{Z}_c$.

B. Actor Network

According to the definition of z_1, we have

$$\begin{aligned} \dot{z}_1 &= \dot{x}_1 - \dot{x}_r \\ &= z_2 + \alpha_1 - \dot{x}_r \end{aligned} \tag{15}$$

where $z_2 = x_2 - \alpha_1$, with α_1 being the virtual control to z_1.

Consider a Lyapunov function candidate $V_1 = \frac{1}{2}z_1^T z_1$. The derivative of V_1 with respect to time is

$$\begin{aligned} \dot{V}_1 &= z_1^T \dot{z}_1 \\ &= z_1^T(z_2 + \alpha_1 - \dot{x}_r) \end{aligned} \tag{16}$$

If we let $\alpha_1 = \dot{x}_r - K_1 z_1$ with $K_1 > 0$, the above derivative of the Lyapunov function candidate will then become

$$\dot{V}_1 = -z_1^T K_1 z_1 + z_1^T z_2 \tag{17}$$

According to the definition of z_2, we have

$$\begin{aligned} \dot{z}_2 &= \dot{x}_2 - \dot{\alpha}_1 \\ &= M^{-1}[D(\tau) - f_{dis} - G - Cx_2] - \dot{\alpha}_1 \end{aligned} \tag{18}$$

Define $\Delta\tau = \tau - D(\tau)$ and consider $V_2 = V_1 + \frac{1}{2}z_2^T M z_2$, so the derivative of V_2 is

$$\begin{aligned} \dot{V}_2 &= -z_1^T K_1 z_1 + z_1^T z_2 + z_2^T\big(\tau - \Delta\tau \\ &\quad -f_{dis} - G - Cx_2 - M\dot{\alpha}_1 + \frac{1}{2}\dot{M}z_2\big) \end{aligned} \tag{19}$$

Applying Property 1, we have

$$\begin{aligned} \dot{V}_2 &= -z_1^T K_1 z_1 + z_1^T z_2 + z_2^T\big(\tau - \Delta\tau - f_{dis} \\ &\quad -G - C\alpha_1 - M\dot{\alpha}_1\big) \end{aligned} \tag{20}$$

The desired control $\tau_d = -z_1 - K_2 z_2 + \Delta\tau + f_{dis} + G + C\alpha_1 + M\dot{\alpha}_1$, where K_2 is a positive constant. However, we do not have information about the deadzone $\Delta\tau$, and system parameters M, C, and G. Therefore, the actor network is used to estimate

$$\tau_d = -z_1 - K_2 z_2 + f_{dis} + W_a^* S_a(Z_a) + \epsilon_a \tag{21}$$

where $Z_a = [x_1, x_2, x_r, \dot{x}_r, \ddot{x}_r, \tau, \tau - b_{r/l}]$, W_a^* is the optimal neural weight and ϵ_a is the estimation error of the actor network. The control law is given by

$$\tau = -z_1 - K_2 z_2 - b_f \text{sgn}(z_2^T) + \hat{W}_a S_a(Z_a) \tag{22}$$

where $\hat{W}_a$ is the estimated neural weight and $\text{sgn}(z_2^T)$ is a vector by applying the sign function to each component of z_2^T.

Let $\tilde{W}_a = \hat{W}_a - W^*$, the instant estimation error is defined as

$$\zeta_a = \tilde{W}_a S_a(Z_a) \tag{23}$$

The objective of the updating law of the actor network is to make the estimation error ζ_a and cost-to-go $\hat{J}$ converge. Define the error involved in the actor network as

$$e_a = \zeta_a(t) + K_J(\hat{J}(t) - J_d(t)) \tag{24}$$

where $J_d(t) = [0, 0..., 0]^T \in R^{n\times 1}$ represents the desired cost-to-go in the future and K_J is a positive constant to be designed. Equivalently, the updating law of $\hat{W}_a$ is to minimize

$$E_a = \frac{1}{2}e_a^T e_a \tag{25}$$

Thus, we obtain

$$\begin{aligned} \dot{\hat{W}}_a' &= -\sigma_a \frac{\partial E_a}{\partial e_a}\frac{\partial e_a}{\partial \zeta_a}\frac{\partial \zeta_a}{\partial \hat{W}_a} \\ &= -\sigma_a(\zeta_a + K_J\hat{J})S_a \end{aligned} \tag{26}$$

where $\sigma_a > 0$ is the learning rate for the actor network. Since ζ_a is unavailable, we develop the following updating law

$$\dot{\hat{W}}_a = -\sigma_a(\hat{W}_a S_a(Z_a) + K_J\hat{J})S_a \tag{27}$$

C. Stability Analysis

Define $V_c = \frac{1}{2}\tilde{W}_c^T\tilde{W}_c$, so its derivative is

$$\dot{V}_c = -\sigma_c\tilde{W}_c^T(r(t)+\hat{W}_c\Lambda)\Lambda \quad (28)$$

Since $r(t) = \frac{J}{\psi} - \dot{J}$, we have

$$\begin{aligned} r(t) &= W_c^{*T}\frac{S_c}{\psi} + \epsilon_c - \nabla(W_c^{*T}S_c+\epsilon_c)\dot{Z}_c \\ &= -W_c^{*T}\Lambda + \varepsilon_c \end{aligned} \quad (29)$$

where $\varepsilon_c = \epsilon_c + \nabla\epsilon_c\dot{Z}_c$, which is bounded, i.e., $\|\varepsilon_c\| \le \varepsilon_{c,max}$. Substituting Eq. (29) into Eq. (28), we obtain

$$\dot{V}_c \le -\frac{\sigma_c\Lambda^T\Lambda}{2}\tilde{W}_c^T\tilde{W}_c + \frac{\sigma_c}{2}\varepsilon_c^T\varepsilon_c \quad (30)$$

Consider a Lyapunov function candidate as below

$$V = \frac{1}{2}z_1^Tz_1 + \frac{1}{2}z_2^TMz_2 + \frac{1}{2}\tilde{W}_a^T\tilde{W}_a + \frac{1}{2}\tilde{W}_c^T\tilde{W}_c \quad (31)$$

Considering the control input (22), updating law (27) and (14), we obtain the derivative of V as below

$$\begin{aligned} \dot{V} &= -z_1^TK_1z_1 - z_2^TK_2z_2 + z_2^T(\tilde{W}_a^TS_a - \epsilon_a \\ &\quad -b_f\text{sgn}(z_2^T) - f_{dis}) - \sigma_a\tilde{W}_aS_a(\hat{W}_aS_a + K_J\hat{J}) \\ &\quad -\sigma_c\tilde{W}_c^T(-W_c^{*T}\Lambda + \hat{W}_c\Lambda + \varepsilon_c)\Lambda \end{aligned} \quad (32)$$

Considering the inequality (30), and substituting

$$\hat{J} = W_c^*S_c + \tilde{W}_cS_c \quad (33)$$

$$\hat{J}^T\hat{J} \le 2(W_c^*S_c)^TW_c^*S_c + 2(\tilde{W}_cS_c)^T\tilde{W}_cS_c \quad (34)$$

into Eq. (32), we have

$$\begin{aligned} \dot{V} &\le -z_2^T(K_2 - \frac{3}{2})z_2 - \frac{\sigma_a-1}{2}\|S_a\|^2\|\tilde{W}_a\|^2 \\ &\quad -z_1^TK_1z_1 - \frac{\sigma_c\Lambda^T\Lambda - 2\sigma_aK_J^2\|S_c\|^2}{2}\|\tilde{W}_c\|^2 \\ &\quad +\frac{1}{2}\|\epsilon_a\|^2 + \frac{\sigma_c}{2}\|\varepsilon_{c,max}\|^2 + \frac{\sigma_a}{2}\|S_a\|^2\|W_a^*\|^2 \\ &\quad +\sigma_aK_J^2\|S_c\|^2\|W_c^*\|^2 \quad (35) \\ &\le -\kappa V + b \quad (36) \end{aligned}$$

where

$$\begin{aligned} \kappa &= \min\Big(K_1, K_2 - \frac{3}{2}, \frac{\sigma_a-1}{2}b_S^2, \\ &\quad \frac{\sigma_cb_\Lambda^2 - 2\sigma_aK_J^2}{2}\Big) \quad (37) \\ b &= \frac{1}{2}\|\epsilon_a\|^2 + \frac{\sigma_c}{2}\|\varepsilon_{c,max}\|^2 + \frac{\sigma_a}{2}b_a^2 \\ &\quad +\sigma_aK_J^2b_c^2 \quad (38) \end{aligned}$$

In above equations, $b_S \le \|S_a\|$ and $b_\Lambda \le \|\Lambda\|$ which are guaranteed by satisfying the persistent excitation condition, and $b_a \ge \|W_a^*\|$ and $b_c \ge \|W_c^*\|$. To ensure $\kappa > 0$, the following conditions must be fulfilled:

$$\begin{aligned} &K_1 > 0,\ K_2 - \frac{3}{2} > 0,\ \frac{\sigma_a-1}{2} > 0, \\ &\frac{\sigma_cb_\Lambda^2 - 2\sigma_aK_J^2}{2} > 0 \end{aligned} \quad (39)$$

Applying Lemma 1, we obtain the main result of this paper, which is summarized in the following theorem.

Theorem 1: Consider the robotic manipulator (1) with an unknown disturbance, a deadzone which fulfills Assumption 2, and a known and smooth reference trajectory. The proposed control (22), with updating laws (27) and (14) with bounded initial conditions, guarantees that the closed-loop system signals (z_1, z_2, $\tilde{W}_a$, and $\tilde{W}_c$) are semiglobally bounded.

IV. Simulation

The proposed control is examined through simulation studies for a 2-DOF robotic manipulator. Details of the inertial matrix M, centripetal and Coriolis matrix C and gravitational force G can be found in [11]. Parameters used for the robotic system are defined in Table 1, where M_u is the uncertain coefficient to the measured mass.

Table 1: Parameters of a 2-DOF robotic manipulator

Parameter	Description	Value
m_1	Mass of link 1	$2.0 \times M_u$kg
m_2	Mass of link 2	$0.85 \times M_u$kg
l_1	Length of link 1	0.35m
l_1	Length of link 2	0.31m
I_1	Inertia of link 1	$\frac{1}{4}m_1l_1^2$
I_2	Inertia of link 2	$\frac{1}{4}m_2l_2^2$

Since the control output is from the actor network, neural weights can be trained by iterations. An improved tracking performance can be expected when the network is trained for several iterations. The objective is to make the output $y = x_1$ follow a desired trajectory: $q_{1d} = \sin(t)$ and $q_{2d} = \sin(t)$. The initial values of y are defined as $y_0 = [-0.1, 0.1]^T$ and $\dot{y}_0 = [1, -1]^T$. The disturbance to the robotic manipulator is randomly generated with the magnitude smaller than 1. The asymmetrical and unknown deadzone is defined to have $b_r = 2.5$ and $b_l = -4.5$ with

$$h_r(v) = 2(v-b_r)(\sin(v)+1) \quad (40)$$

$$h_l(v) = (v-b_l)^3 \quad (41)$$

Given the situation that we have no information about the deadzone functions, system dynamics and disturbance, the proposed control (22) is expected to keep the manipulator in desired boundries. A total of $N = 2^8$ nodes are used for the actor NN $W_aS_a(Z_a)$. The centers are evenly distributed in:

$[-1,1]\times[-1,1]\times[-10,10]\times[-10,10]\times[-1,1]\times[-1,1]\times[-1,1]\times[-1,1]\times[0]\times[0]\times[0]\times[0]\times[0]\times[0]$. 3^2 nodes are used for the critic NN $W_cS_c(Z_c)$. Centers of S_c are evenly distributed in the space of $[-5,5]\times[-5,5]$ with each parameter being -5, 0, or 5. The learning rate $\sigma_a = 25$ and $\sigma_c = 10$. Variances of the RBFNNs are set to be 100 and 500 for S_a and S_c, respectively. Initial weights for the first iteration are defined as $\hat{W}_{a,N,i} = 0$ $(i = 1,2;\ N = 1,2,3,4,...2^8)$ and $\hat{W}_{c,N,j} = 0$ $(j = 1,2;\ N = 1,2,3,4,...3^2)$. The control gains $K_1 = 60$, $K_2 = 10$, and $K_J = 0.2$.

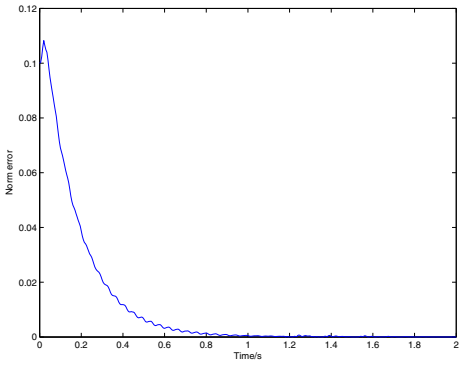
After each iteration, the final neural weights will be used as the initial neural weights for the next iteration. Simulation is also conducted with the adaptive NN control with offline tuning in [17]. The adaptive NN control is tuned to its achievable best performance by trial and error. Several M_u are chosen to study the performance of the proposed control under mass uncertainty with same control parameters.

The tracking performance (the norm of position error) under the proposed actor critic NN control is shown in Figs. 3 and 4. In the first iteration, tracking errors tend to converge to a small value while fluctuation is observed. After a few iterations, the fluctuation has been reduced significantly, which indicates an enhanced tracking performance. When $M_u = 1$, after the 10^{th} iteration, the neural weights are used for trajectory tracking of the 2-DOF manipulator. The control inputs, as shown in Fig. 5, are rather smooth, which further verify the validity of the proposed method.

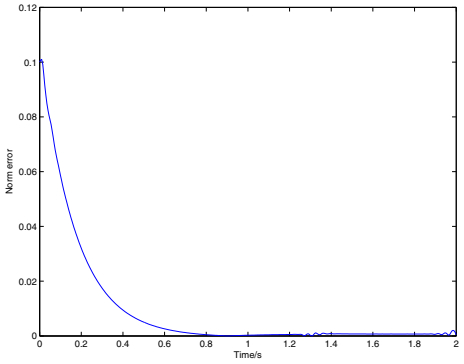
Fig. 6 shows the tracking performance of the adaptive RBFNN control with different M_u. The tracking performance is slightly better in comparison with actor critic NN control when $M_u = 1$. However, the method in [17] is limited when uncertainties appear. In Fig. 6.b, the control performance with fine-tuned parameters for the case of $M_u = 1$ is not satisfactory in the case of $M_u = 13$. One way to improve the performance for $M_u = 13$ is to re-tune the control gains by trial and error, which is tedious and may be unpractical as the system may vary frequently in operation.

V. Conclusion

In this paper, an actor critic NN control has been proposed to solve the problem of off-line tuning faced by conventional adaptive NN control. Tracking errors of the system under the proposed actor critic NN control are proved to be semiglobally uniformly bounded and converge to a compact set. Performance of the proposed control has been examined through simulation of a 2-DOF manipulator with unknown random disturbance and deadzone.



(a) 1^{st} iteration



(b) 10^{th} iteration

Fig. 3. Norm of position error under the proposed method: $M_u = 1$.

References

[1] W. Zhonghua, Y. Bo, C. Lin, and Z. Shusheng, "Robust adaptive deadzone compensation of DC servo system," in *IEE Proceedings of Control Theory and Applications*, vol. 153, pp. 709–713, IET, 2006.

[2] R. Selmic and F. Lewis, "Deadzone compensation in motion control systems using neural networks," *IEEE Transactions on Automatic Control*, vol. 45, no. 4, pp. 602–613, 2000.

[3] R. Selmic, V. Phoha, and F. Lewis, "Intelligent compensation of actuator nonlinearities," in *Proceedings of IEEE Conference on Decision and Control*, vol. 4, pp. 4327–4332, IEEE, 2003.

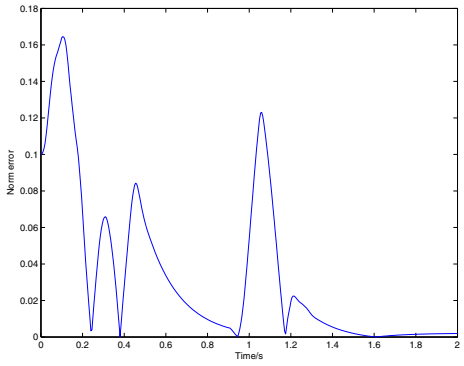
[4] Y. Li and S. S. Ge, "Impedance learning for robots interacting with unknown environments," *IEEE Transactions on Control Systems Technology*, doi: 10.1109/TCST.2013.2286194.

[5] D. Vrabie and F. Lewis, "Adaptive optimal control algorithm for continuous-time nonlinear systems based on policy iteration," in *Proceedings of IEEE Conference on Decision and Control, 2008*, pp. 73–79, IEEE, 2008.

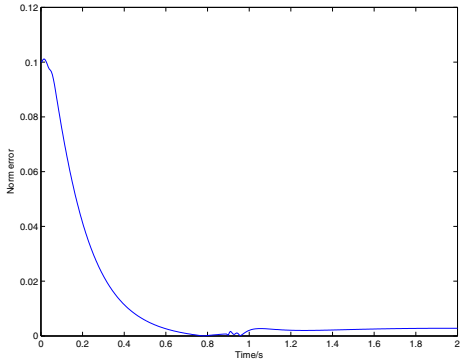
[6] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.

[7] K. G. Vamvoudakis and F. L. Lewis, "Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[8] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE Transactions on Neural Networks,*, vol. 12, no. 2, pp. 264–276, 2001.



(a) 1^{st} iteration



(b) 10^{th} iteration

Fig. 4. Norm of position error under the proposed method: $M_u = 13$.

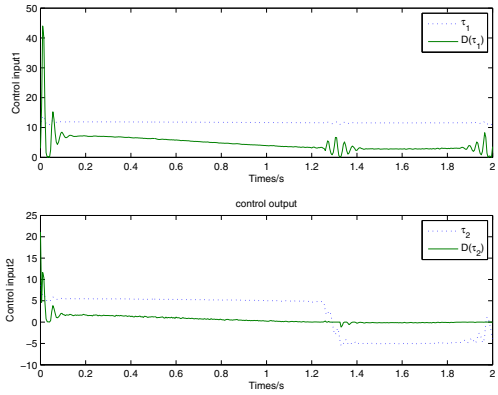
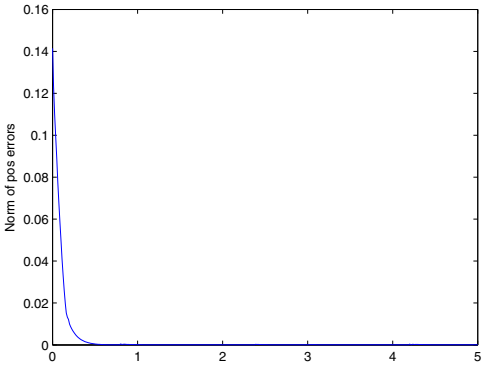
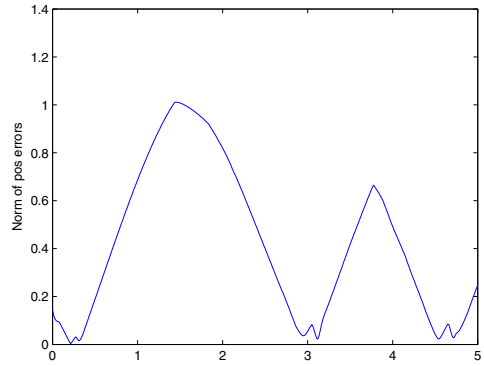


Fig. 5. Control input under the proposed method



(a) $M_u = 1$



(b) $M_u = 13$

Fig. 6. Norm of position error under adaptive NN control in [17].

[9] Q. Yang and S. Jagannathan, "Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 377–390, 2012.

[10] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 32–50, 2009.

[11] S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control of Robotic Manipulators*, vol. 19. World Scientific Pub Co Inc, 1998.

[12] T. Zhang and S. S. Ge, "Adaptive neural control of MIMO nonlinear state time-varying delay systems with unknown dead-zones and gain signs," *Automatica*, vol. 43, no. 6, pp. 1021–1033, 2007.

[13] S. S. Ge and C. Wang, "Adaptive NN control of uncertain nonlinear pure-feedback systems," *Automatica*, vol. 38, no. 4, pp. 671–682, 2002.

[14] R. Sanner and J. Slotine, "Gaussian networks for direct adaptive control," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 837–863, 1992.

[15] S. S. Ge and C. Wang, "Adaptive neural control of uncertain MIMO nonlinear systems," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 674–692, 2004.

[16] K. Doya, "Reinforcement learning in continuous time and space," *Neural Computation*, vol. 12, no. 1, pp. 219–245, 2000.

[17] S. S. Ge, W. He, and S. Xiao, "Adaptive neural network control for a robotic manipulator with unknown deadzone," in *Proceedings of the Chinese Control Conference (CCC), 2013*, (Xi'an, China), pp. 2997–3002, 2013.