A Reinforcement Learning Neural Network for Robotic Manipulator Control

Yazhou Hu
huyazhou@sia.cn
State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, P.R.C., and University of Chinese Academy of Sciences, Beijing 100049, P.R.C.

Bailu Si
sibailu@sia.ac.cn
State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, Shenyang, P.R.C.

We propose a neural network model for reinforcement learning to control a robotic manipulator with unknown parameters and dead zones. The model is composed of three networks. The state of the robotic manipulator is predicted by the state network of the model, the action policy is learned by the action network, and the performance index of the action policy is estimated by a critic network. The three networks work together to optimize the performance index based on the reinforcement learning control scheme. The convergence of the learning methods is analyzed. Application of the proposed model on a simulated two-link robotic manipulator demonstrates the effectiveness and the stability of the model.

1 Introduction

In recent years, with the increasing demand of automatic manipulation and manufacturing in industry, there has been a substantial growth of the deployment of robotic manipulators in engineering applications, such as nuclear power plants (Kim, Seo, Lee, Choi, & Moon, 2015), rehabilitation (Ka, Ding, & Cooper, 2016), haptics and virtu of reality (Zarenia & Sepehri, 2015), cognition (Azeem, Iqbal, Toivanen, & Samad, 2012; Naveed, Iqbal, & Ur Rehman, 2012), motion assistance (Naitchabane, Hoppenot, & Colle, 2016), target detection (Iqbal, Pasha, Baizid, Khan, & Iqbal, 2013; Iqbal, Pasha, Nabi, Khan, & Tqbal, 2013), and other areas (Iqbal, Rehmansaad, Malik, & Mahmood Tahir, 2014; Baizid, Chellali, Yousnadj, Iqbal, & Bentaleb, 2010). To function as an intelligent system, it is unavoidable for a robotic manipulator to deal with the problems of unknown parameters and nondifferentiable nonlinearity—the dead zone (Tao & Kokotovic, 1994). It has been pointed out that the issues of unknown parameters and the dead zone may lead to large steady-state error, poor transient response, large overshoot, and other undesirable performance (Wang, Yan, Cai, & Zhang, 2006). Wang (2010) developed a recursive algorithm to estimate the unknown manipulator parameters and the payload parameters. Neural network models are also proposed by robotic researchers as intelligent control algorithms (Min, Lu, Xu, Duan, & Chen, 2017; Wang, Sun, Pan, & Yu, 2017) or for spatial perception (Leitner, Harding, Frank, Förster, & Schmidhuber, 2013). Compared with other intelligent systems, there are nonlinearities, such as dead zones, in robotic manipulators, and it is difficult to analyze those nonlinearities and build their exact mathematical models (Gao & Selmic, 2006). Neural networks have been widely used in control design for robotic manipulators in recent years (He, Dong, & Sun, 2015) due to their ability to model the dynamics and the nonlinearities of a robotic manipulator. Chen, Wen, Liu, and Wang (2014) approximated the uncertain nonlinear dynamics of a multiagent time delay system by radial basis function neural networks. Kim and Lewis (2000) used neural network models to build controllers for a robotic manipulator in the presence of modeling uncertainties and frictional forces. Liu, Kadirkamanathan, and Billings (1999) estimated the unknown nonlinearity of a dynamical system by a novel neural network architecture, referred to as a variable neural network. In Huang and Tan (2012) and Seidl, Lam, Putnam, and Lorenz (1993), nonlinear models of friction and backlash hysteresis were approximated by neural networks.

Except neural network methods, the theory of adaptive control has drawn much attention in developing control algorithms for robotic manipulators. An adaptive control method was integrated with a sliding mode technique and a neural network to identify unknown parameters (Sun, Pei, Pan, Zhou, & Zhang, 2011; Purwar, Kar, & Jha, 2005). Schindele and Aschemann (2009) considered model-based compensation of nonlinearities for robotic manipulators. The adaptive force/motion tracking control was investigated by Li, Ge, Adams, and Wijesoma (2008) for nonholonomic mobile manipulators with unknown parameters and disturbances under uncertain holonomic constraints. Adaptive optimal control without weight transport was proposed by Chinta and Tweed (2012). A series of algorithms were designed in Tong, Wang, Li, and Chen (2013), Chen, Jiao, Li, and Li (2010), Li, Cao and Ding (2011), Chen and Jiao (2010), Liu, Tong, Wang, Li, and Chen (2011), Tang, Liu, and Tong (2014) for nonlinear systems with uncertain functions and a dead zone, based on intelligent control methods.

Furthermore, it has become popular to combine adaptive control theory with neural networks for better controller design. Sanner and Slotine (1995) extended the stable adaptive tracking controller designs, employing neural networks, to classes of multivariable mechanical systems, that is, robot manipulators. Liu, Chen, Zhang, and Chen (2015) proposed an adaptive neural network controller for dual-arm coordination of a humanoid robot. Chen, Liu, and Wen (2014) proposed adaptive fuzzy neural networks to estimate a nonlinear stochastic system with unknown functions. An adaptive controller, based on a higher-order neural network was employed for a class of uncertain multiple-input-multiple-output (MIMO) nonlinear systems in Liu, Chen, Wen, and Tong (2011) and Liu, Tang, Tong, Chen, and Li (2015). An adaptive neural network control method was proposed by Liu, Tang, Tong, and Chen (2015) for MIMO nonlinear systems in strict feedback form. Rossomando, Soria, and Carelli (2013) proposed a neural adaptive compensator and a trajectory tracking controller for a unicycle-like mobile robot.

Reinforcement learning (RL) is another approach to tackle these control issues. With the development of representation learning techniques, many control methods based on RL have solved large-scale problems (Schmidhuber, 2015). An actor agent, applying an action or a control policy to the actuator (the environment), and a critic component (the agent), assessing the cost-to-go at current state with current control policy, are contained in a typical RL actor-critic structure (Yang & Jagannathan, 2012). The RL output signal is updated via the value of the critic component. It is the ultimate goal that the cost-to-go is converged to its global optimum. A variety of intelligent control algorithms have been developed based on RL to solve control problems (Gosavi, 2014; Sharma & Gopal, 2010; Runa & Sharma, 2015; Lin, 2009). Compared with conventional algorithms, the advantage of RL is that it is not necessary to obtain a perfect dynamic model of the system being controlled. A critic-actor-based–RL neural network control method was proposed to achieve adaptive control during poststroke fine hand motion rehabilitation training by Huang, Naghdy, Du, and Naghdy (2015). Gu, Holly, Lillicrap, and Levine (2016) discussed a deep neural network that learned to approximate the Q-function based on off-policy training to solve complex 3D manipulation tasks. Yahya, Kalakrishnan, Chebotar, and Levine (2016) explored distributed and asynchronous policy learning as a means to achieve generalization and improve training performance on challenging, real-world manipulator tasks. RL coupled with adaptive dynamic programming (ADPRL) has also become popular and can be classified into (Prokhorov & Wunsch, 1997; Wang, Zhang, & Liu, 2009; Si, Barto, Powell, & Wunsch, 2004) heuristic dynamic programming (HDP), action-dependent HDP (ADHDP), dual heuristic dynamic programming (DHP), action-dependent DHP (ADDHP), globalized DHP (GDHP), and ADGDHP (action-dependent GDHP).

In this letter, a neural network model for RL is presented to control a rigid robotic manipulator with unknown parameters and a dead zone. The model has three neural networks: a state network, action network, and critic network. The state network is used to estimate the state information of the robotic manipulator, the critic network is proposed to evaluate the performance of the network, and the action network learns a policy to optimize the performance index. The reliability of this network controller is proven by simulations of a two-link robotic manipulator. It should be noted that the proposed model is applicable when the dynamics and the parameters of the robot manipulator are not known but the output information of the robot manipulator can be measured by sensors. Thus, this letter provides an adaptive control policy for a complex dynamical system, without a priori knowledge of the dynamics and the parameters of the system, even when there are nonlinearities in the system.

The letter is organized as follows. Section 2 introduces the nonlinear dynamic model of a robotic manipulator. In section 3, we describe the network model and the learning rules in detail. We apply the network model to control a simulated two-link robotic manipulator in section 4, and the simulation demonstrates the effectiveness of the model. The letter concludes in section 5.

2 Problem Formulation

The dynamics of a robotic manipulator with an n-dimension rigid body can be described by a nonlinear dynamic equation,

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) = \tau, \tag{2.1}$$

where $\theta \in R^n$, $\dot{\theta}$, and $\ddot{\theta}$ are the angular position, angular velocity, and angular acceleration of the robotic manipulator, respectively. $M(\theta) \in R^{n\times n}$ is the manipulator inertia matrix, $C(\theta,\dot{\theta}) \in R^n$ indicates the centrifugal and coriolis effects, $G(\theta) \in R^n$ is the gravitational force effect, and $\tau \in R^n$ is the control input signal (i.e., torques and forces). According to Alessandri (2004), the following property can be obtained: the matrix $M(\theta)$ is symmetric, bounded, and positive definite. Due to this property, the inverse matrix of $M(\theta)$ exists. The dynamics can be written as

$$\ddot{\theta} = M^{-1}(\theta)\{-C(\theta,\dot{\theta})\dot{\theta} - G(\theta)\} + M^{-1}(\theta)\tau. \tag{2.2}$$

Defining $x_1 = \theta$, $x_2 = \dot{\theta}$, and $y = \theta$, the state equation of the robotic manipulator becomes

$$\begin{aligned} \dot{x} &= \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} x_2 \\ M^{-1}(x_1)\{-C(x_1,x_2)x_2 - G(x_1)\} \end{pmatrix} + \begin{pmatrix} 0 \\ M^{-1}(x_1) \end{pmatrix}\tau, \\ y &= [1 \quad 0]x. \end{aligned} \tag{2.3}$$

$M^{-1}(x_1)\{-C(x_1,x_2)x_2 - G(x_1)\}$ is a nonlinear function of x_1 and x_2. We define $f(x) = \begin{pmatrix} x_2 \\ M^{-1}(x_1)\{-C(x_1,x_2)x_2 - G(x_1)\} \end{pmatrix}$ and $g(x) = \begin{pmatrix} 0 \\ M^{-1}(x_1) \end{pmatrix}$, $h = [1\ 0]$, $u = \tau$. Then the state equation can be rewritten as

$$\begin{aligned} \dot{x} &= f(x) + g(x)u, \\ y &= hx. \end{aligned} \tag{2.4}$$

For ease of computer simulation, we consider the discrete form of the state equation:

$$\begin{aligned} x(k+1) &= F(x(k)) + G(x(k))u(k), \\ y(k) &= H(x(k))x(k). \end{aligned} \tag{2.5}$$

According to modern control theory, the performance index of the state system is quantified by

$$J = \sum_{i=k}^{\infty} (x^T(i)Qx(i) + u^T(i)Ru(i)), \tag{2.6}$$

where $x(k)$ is the state information of state equation 2.5, $u(k)$ is the control input signal, and Q and R are positive-definite matrices. According to the Lyapunov stability theory, the performance index for a stable system is convergent.

We introduce a discount factor $\gamma \in (0, 1)$ into the performance index function:

$$J(x(k), u(k)) = \sum_{i=k}^{\infty} \gamma^{k-1} \left(x^T(i)Qx(i) + u^T(i)Ru(i)\right). \tag{2.7}$$

Since γ^k is convergent for $\gamma \in (0, 1)$, the performance index $J(x(k), u(k))$ is convergent. The performance index can be written in a recursive form:

$$J(x(k), u(k)) = \left(x^T(k)Qx(k) + u^T(k)Ru(k)\right) + \gamma J(x(k+1), u(k+1)). \tag{2.8}$$

Let $r = x^T(k)Qx(k) + u^T(k)Ru(k)$; the performance index then becomes

$$J(x(k), u(k)) = r + \gamma J\left(x(k+1), u(k+1)\right). \tag{2.9}$$

According to the Bellman optimality principle (Lewis, Vrabie, & Syrmos, 2012), equation 2.9 satisfies the discrete-time Hamilton-Jacobi-Bellman (HJB) equation:

$$J^*(x(k), u(k)) = \min_{u(k)} \left(r + \gamma J\left(x(k+1), u(k+1)\right)\right). \tag{2.10}$$

That is, the optimal control problem can be changed into solving the discrete-time HJB optimality equation. The optimal input signal can be obtained by minimizing equation 2.10:

$$u^* = \arg\min_{u(k)} \left(J\left(x(k), u(k)\right)\right) = \arg\min_{u(k)} \left(r + \gamma J\left(x(k+1), u(k+1)\right)\right). \tag{2.11}$$

For finite Markov decision processes (MDP), the HJB optimality equation has a unique solution (Sutton & Barto, 2017). Dynamic programming (DP) can be used to solve the HJB optimality equation. Consequently, equation 2.10 is solvable.

3 Control Design

3.1 Introduction to Reinforcement Learning. RL is a learning paradigm for sequential decision making, solving the delayed credit assignment problem in a wide range of fields in science and engineering (Sutton & Barto, 2017). The idea of RL comes from the phenomenon that humans and other animals learn from experience via reward and punishment for survival and growth (Si, Barto, Powell, & Wunsch, 2012; Werbos, 2009; Lewis & Liu, 2013). RL agents interact with the environment and learn by trial and error. At each time t, a state s_t is received by the agent, and an action a_t is selected from the action space A, following a policy $\pi(a|s)$, which represents the agent's behavior. Meanwhile, the environment transits to the next state s_{t+1} according to the state transition probability $P(s|s', a)$, and a reward r_t based on the reward function $R(s, a)$ is released to the agent (Szepesvari, 2010; Liu, Wei, Wang, Yang, & Li, 2013). In an iterative problem, this process always keeps going until the agent reaches the terminal state and then restarts. The agent aims to find the actions with maximal return from each state measured, for example, by the discounted cumulative reward $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, with the discount factor $\gamma \in (0, 1)$.

The state value function $V^\pi(s) = E[R_t]$ is the expected return when starting in s and following policy π. The optimal state value function $V^*(s)$ is the maximum expected return starting in state s. The value of a state-action pair is measured by the state-action value function $Q^\pi(s, a) = E[R_t|s_t = s, a_t]$, defined as the expected return when starting from a station-action pair following a policy. An optimal state-action value function $Q^*(s, a)$ is the maximal state-action value function achievable by any policy for state s and action a. When the maximum value is sought, the optimal actions are found correspondingly.

Although RL methods can be applied to solve some manipulation tasks for dynamic robotic systems, they suffer from poor scalability in high-dimensional nonlinear systems (Levine & Koltun, 2013). Neural networks offer the ability of universal function approximation in high-dimensional space. Therefore, in this letter, we use neural networks to represent the

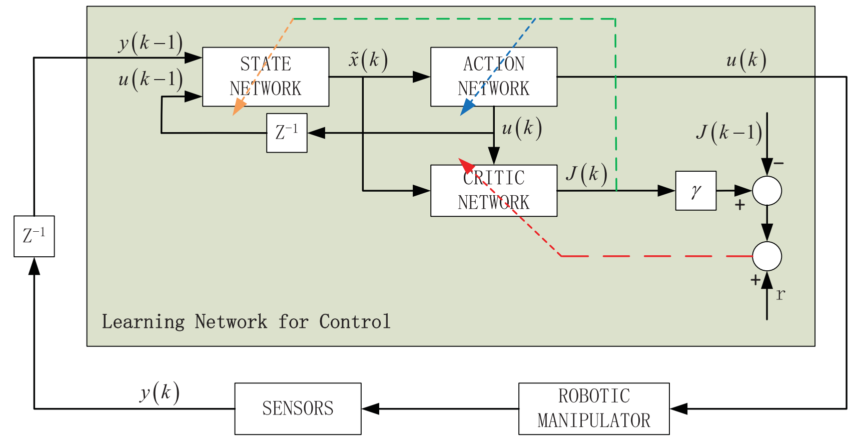

Figure 1: The neural network model for the control of a robotic manipulator by reinforcement learning.

states of the system, approximate the state-action value function, and learn the policy for optimal control.

3.2 Neural Network Model for Reinforcement Learning. To control a robotic manipulator, we introduce a neural network model (see Figure 1), which learns a control policy by RL. The structure of the network model is similar to the actor-critic algorithm and contains three subnetworks: the state network, the critic network, and the action network. The state network predicts the state of the system based on the previous state and the action, the critic network provides an internal estimation of the performance of the network, and the action network searches for a policy to minimize the estimated performance index. Each subnetwork is implemented by a three-layer perceptron network with the same activation function $S(x) = (1 - e^{-x})/(1 + e^{-x})$.

3.2.1 State Network. The state network is composed of N_s^I input units, N_s^H hidden units, and N_s^O output units. The inputs to the network are the system state $y(k-1)$ and the output sequences $u(k-1)$. The output of state network is the predicted state $\tilde{x}(k)$, which has the same dimension as the real internal state. The state network can be described as follows.

$$\begin{aligned}\text{Input layer:}\quad & f(k) = \omega_{s1}^T(k)X(k-1),\\ \text{Hidden layer:}\quad & \phi_s(k) = S(f(k)),\\ \text{Output layer:}\quad & \tilde{x}(k) = \omega_{s2}^T(k)\phi_s(k),\end{aligned}$$

where $X(k-1) = [y^T(k-1), u^T(k-1)]^T$ is the input vector of the state network, T is the transpose operator, and $\omega_{s1}(k)$ and $\omega_{s2}(k)$ are the weight matrices from the input layer to the hidden layer and from the hidden layer to the output layer, respectively.

The prediction error of the state network is qualified by

$$E_s(k) = \frac{1}{2}e_s^2(k), \tag{3.1}$$

where $e_s(k) = J(\tilde{x}(k), u(k))$. Weight matrices $\omega_{s1}(k)$ and $\omega_{s2}(k)$ are initially set as random matrices and are updated by gradient descent,

$$\omega_s(k+1) = \omega_s(k) - \eta_s \frac{\partial E_s(k)}{\partial \omega_s(k)}, \tag{3.2}$$

where $\eta_s > 0$ is the learning rate of the state network. For convenience, let $J = J(\tilde{x}(k), u(k))$. According to backpropagation rules, the change of the weight matrix is given by

$$\Delta\omega_s = \frac{\partial E_s(k)}{\partial \omega_s(k)} = \frac{\partial E_s(k)}{\partial J}\left(\frac{\partial J}{\partial \tilde{x}(k)} + \frac{\partial J}{\partial u(k)}\frac{\partial u(k)}{\partial \tilde{x}(k)}\right)\frac{\partial \tilde{x}(k)}{\partial \omega_s(k)}, \tag{3.3}$$

leading to

$$\begin{aligned}\Delta\omega_{s1} &= \omega_{c2}^T(k)\phi_c(k)\omega_{c2}^T(k)\phi_c(k)\omega_{c1}^T(k)\left(1 + \omega_{a2}^T(k)\phi_a(k)\omega_{a1}^T(k)\right)\\ &\quad \omega_{s2}^T(k)\phi_s(k)X(k-1), \qquad (3.4)\\ \Delta\omega_{s2} &= \omega_{c2}^T(k)\phi_c(k)\omega_{c2}^T(k)\phi_c(k)\omega_{c2}^T(k)\left(1 + \omega_{a2}^T(k)\phi_a(k)\omega_{a2}^T(k)\right)\phi_s(k). \qquad (3.5)\end{aligned}$$

3.2.2 Critic Network. The performance index of the robotic manipulator system is approximated by the critic network. Representing how "good" or "bad" an action and the estimated state are, the approximated performance index serves as the reward signal for the action network. The critic network is realized by a three-layered perceptron neural network, with N_c^I input units, N_c^H hidden units, and N_c^O output units:

$$\begin{aligned}\text{Input layer: } & g(k) = \omega_{c2}^T(k)[\tilde{x}^T(k), u^T(k)]^T,\\ \text{Hidden layer: } & \phi_c(k) = S(g(k)),\\ \text{Output layer: } & J(\tilde{x}(k), u(k)) = \omega_{c2}^T(k)\phi_c(k),\end{aligned}$$

where $J(\tilde{x}(k), u(k))$ is the approximated performance index, $\omega_{c1}(k)$ denotes the weight matrix from the input layer to the hidden layer, and $\omega_{c1}(k)$ is the weight matrix from the hidden layer to the output layer. $\tilde{x}(k)$, but not $x(k)$, is used as the input to the critic network in order to reduce the effects of the unknown parameters and the dead zone.

The loss of the performance approximation is defined by

$$E_c(k) = \frac{1}{2} e_c^2(k), \tag{3.6}$$

where $e_c(k) = r + \gamma J(\tilde{x}(k+1), u(k+1)) - J(\tilde{x}(k), u(k))$, and $r = \tilde{x}^T(k) Q \tilde{x}(k) + u^T(k) R u(k)$.

The weight matrices are initialized randomly and are updated by gradient descent,

$$\omega_c(k+1) = \omega_c(k) - \eta_c \frac{\partial E_c(k)}{\partial \omega_c(k)}, \tag{3.7}$$

where $\eta_c > 0$ is the learning rate of the critic network. According to the chain rule in backpropagation,

$$\Delta\omega_c = \frac{\partial E_c(k)}{\partial \omega_c(k)} = \frac{\partial E_c(k)}{\partial J(\tilde{x}(k))} \frac{\partial J(\tilde{x}(k))}{\partial \omega_c(k)}. \tag{3.8}$$

The learning rules for the weight matrices become

$$\begin{aligned} \Delta\omega_{c1} &= \gamma \left(\gamma \omega_{c2}^T(k) \phi_c(k) + r - \omega_{c2}^T(k-1) \phi_c(k-1) \right) \\ &\quad \omega_{c2}^T \phi_c(k) \left[\tilde{x}^T(k), u^T(k) \right]^T, \end{aligned} \tag{3.9}$$

$$\Delta\omega_{c2} = \gamma \left(\gamma \omega_{c2}^T(k) \phi_c(k) + r - \omega_{c2}^T(k-1) \phi_c(k-1) \right) \phi_c(k). \tag{3.10}$$

3.2.3 Action Network. In the model, the action network is used to provide the control input signal $u(k)$ for the robotic manipulator. The action network is constructed as a three-layer perception network, with N_a^I input units, N_a^H hidden units, and N_a^O output units, respectively. The action network transforms the estimated state $\tilde{x}(k)$ through the connection matrices ω_{a1} and ω_{a2} into the control signal:

$$\begin{aligned} \text{Input layer: } & h(k) = \omega_{a1}^T(k) \tilde{x}(k), \\ \text{Hidden layer: } & \phi_a(k) = S(h(k)), \\ \text{Output layer: } & u(k) = \omega_{a2}^T(k) \phi_a(k). \end{aligned}$$

The control law $u(k)$ is learned by minimizing an objective function $E_a(k)$,

$$E_a(k) = \frac{1}{2} e_a^2(k), \tag{3.11}$$

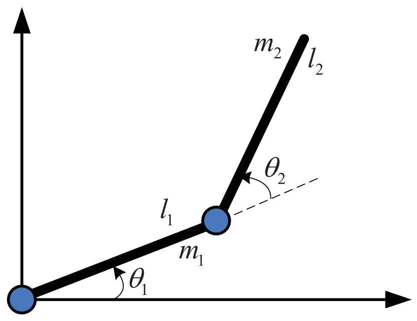

Figure 2: A two-link robotic manipulator.

where $e_a(k) = J(\tilde{x}(k), u(k))$. This is done by performing gradient descent on the space of connection matrices,

$$\omega_a(k+1) = \omega_a(k) - \eta_a \frac{E_a(k)}{\omega_a(k)}, \tag{3.12}$$

where $\eta_a > 0$ is the learning rate of the action network. For convenience, let $J = J(\tilde{x}(k), u(k))$. According to the chain rule, the update of the weight matrices has the form

$$\Delta\omega_a = \frac{\partial E_a(k)}{\partial \omega_a(k)} = \frac{\partial E_a(k)}{\partial J} \frac{\partial J}{\partial u(k)} \frac{\partial u(k)}{\partial \omega_a(k)}. \tag{3.13}$$

As a consequence, the learning rules for the weight matrices of the action network are

$$\Delta\omega_{a1} = \omega_{c2}^T(k)\phi_c(k)\omega_{a2}^T(k)\phi_a(k)\tilde{x}(k), \tag{3.14}$$

$$\Delta\omega_{a2} = \omega_{c2}^T(k)\phi_c(k)\phi_a(k). \tag{3.15}$$

3.2.4 Weights Convergence Analysis. It is important that the three subnetworks work together and the whole network is kept stable. According to Lyapunov stability theory, we define a Lyapunov function of the model and show that the function is always decreasing. Thus, the learning algorithm proposed in this letter is convergent. The details of the proof are shown in the appendix.

4 Simulation

We apply the proposed model on a simulated two-link robotic manipulator (see Figure 2). According to the discussion in section 2, the dynamics of the

Table 1: Meanings and Values of the Physical Parameters of the Two-Link Robotic Manipulator.

Symbol	Parameter	Value
l_1	Length of the first link	0.2 m
l_2	Length of the second link	0.2 m
g	Gravity acceleration	9.8 $\mathrm{m/s^2}$
m_1	Mass of the first link	2 kg
m_2	Mass of the second link	1 kg
$\dot{\theta}_{1,\max}$	Maximum angular speed of the first link	2π rad/sec
$\dot{\theta}_{2,\max}$	Maximum angular speed of the second link	2π rad/sec
$\tau_{1,\max}$	Maximum torque of the first link	0.2 N·m
$\tau_{2,\max}$	Maximum torque of the second link	0.2 N·m

robotic manipulator can be described as

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) = \tau,$$

where $\theta = [\theta_1\ \theta_2]^T$ are the angles of the two links and $\tau = [\tau_1\ \tau_2]^T$ the torques applied to the two joints. The state of the manipulator is composed of the link angles θ_1, θ_2, and the angular velocities $\dot{\theta}_1$, $\dot{\theta}_2$. $\ddot{\theta}_1$ and $\ddot{\theta}_2$ are the angular acceleration. The mass matrix $M(\theta)$, the coriolis and centrifugal forces matrix $C(\theta,\dot{\theta})$, and the gravity vector $G(\theta)$ are given by

$$M(\theta) = \begin{bmatrix} (m_1+m_2)\,l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos\theta_2 & m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2 \\ m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2 & m_2 l_2^2 \end{bmatrix},$$

$$C(\theta,\dot{\theta}) = \begin{bmatrix} -2m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_2 & -m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_2 \\ m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1 & 0 \end{bmatrix},$$

$$G(\theta) = \begin{bmatrix} (m_1+m_1)\,g l_1 \sin\theta_1 + m_2 g l_2 \sin(\theta_1+\theta_2) \\ m_2 g l_2 \sin(\theta_1+\theta_2) \end{bmatrix}.$$

According to the engineering needs, the physical parameters of the two-link robotic manipulator and the corresponding values used in the simulation are listed in Table 1.

Given these parameters, $M(\theta)$, $C(\theta,\dot{\theta})$, and $G(\theta)$ can be calculated:

$$M(\theta) = \begin{bmatrix} 0.16 + 0.08\cos\theta_2 & 0.04 + 0.04\cos\theta_2 \\ 0.04 + 0.04\cos\theta_2 & 0.04 \end{bmatrix},$$

$$C(\theta,\dot{\theta}) = \begin{bmatrix} -0.08\sin\theta_2\dot{\theta}_2 & -0.04\sin\theta_2\dot{\theta}_2 \\ 0.04\sin\theta_2\dot{\theta}_1 & 0 \end{bmatrix},$$

Table 2: Meanings and Values of the Parameters of the Network Model.

Symbol	Parameter	Value
γ	Discount factor	0.85
η_s, η_c, η_a	Learning rate	0.1
Q	Coefficient matrix in the performance index	4×4 identity matrix
R	Coefficient matrix in the performance index	2×2 identity matrix
N_s^I	Number of input units in the state network	4
N_s^H	Number of hidden units in the state network	8
N_s^O	Number of output units in the critic network	4
N_c^I	Number of input units in the critic network	6
N_c^H	Number of hidden units in the critic network	10
N_c^O	Number of output units in the action network	1
N_a^I	Number of input units in the action network	4
N_a^H	Number of hidden units in the action network	10
N_a^O	Number of output units in the action network	2

$$G(\theta) = \begin{bmatrix} 5.88 \sin \theta_1 + 1.96 \sin(\theta_1 + \theta_2) \\ 1.96 \sin(\theta_1 + \theta_2) \end{bmatrix}.$$

Through numerical calculation, the existence of $M^{-1}(\theta)$ can be confirmed for $\theta_1, \theta_2 \in [-\pi, \pi]$, a result guaranteed by the property shown in section 2. Therefore, the algorithm proposed in this letter based on the state equation of robotic manipulators is valid. The complete process state, given by $x = \left[\theta^T \ \dot{\theta}^T\right]^T$, $\theta = [\theta_1 \ \theta_2]^T$. $u \equiv [u_1 \ u_2]^T = \tau \equiv [\tau_1 \ \tau_2]^T$, is the command of the input to the manipulator.

In the simulation, the dynamics and the parameters of the two-link robotic manipulator are not known to the network model. The dead zone (the threshold value is chosen as 1 s) is added in the two-link manipulator to test the control performance of the proposed network model. The network treats the manipulator as a black box and searches for a policy to optimize the performance index by RL.

On the basis of the discussion above and the feature of this two-link robotic manipulator, the parameters for the network model are given in Table 2. The initial weights for these three networks are set randomly within (0, 1). We trained the network for 500 trials, and each trial lasted 100 seconds. In order to show the generalization ability of the proposed network model, during the training process, the initial positions and the final positions are set randomly for this two-link manipulator. During the full simulation, the angular position, $\theta \in [-\pi, \pi]$ and its speed $\dot{\theta}$ and acceleration $\ddot{\theta}$ are kept within certain bounds to match the physical limits of the manipulator. During the learning trials, the loss functions ($E_s(k)$, $E_a(k)$, $E_c(k)$) are reduced quickly and are already stabilized after about 10 seconds (see Figure 3). This demonstrates that the learning methods converge.

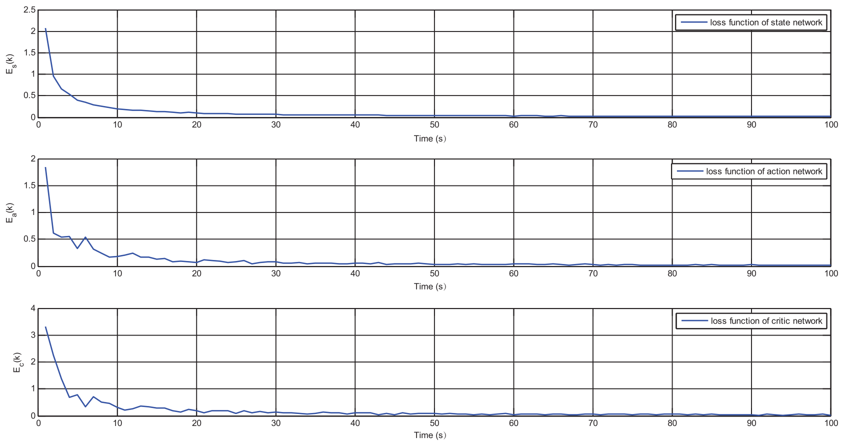

Figure 3: The loss functions of the network model converge quickly during training. (Top) The loss function $E_s(k)$ of the state network. (Middle) The loss function $E_a(k)$ of the action network. (Bottom) The loss function $E_c(k)$ of the critic network.

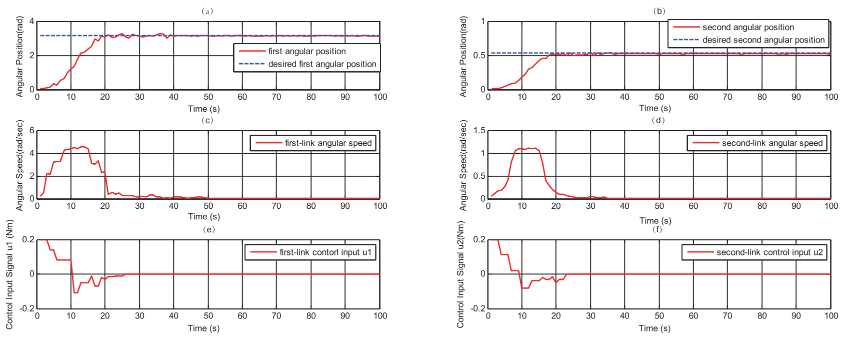

Figure 4: After learning, the network is able to quickly control the robotic manipulator to the desired positions. (a, b) The angular position of the robotic manipulator. The dashed lines are the desired position of the two joints. (c, d) The angular speeds of the robotic manipulator. (e, f) The control inputs to the robotic manipulator produced by the network.

After training, the learned network model is used to control the robotic manipulator. The initial values of θ_1 and θ_2 are set as 0 for the test experiment, and the final values of θ_1 and θ_2 are set as π and $\pi/6$, respectively. Within 20 seconds, the angular positions of the robotic manipulator are steered to the desired final positions (see Figures 4a and 4b). The trajectories of the angles of the manipulator are quite smooth. The angular speeds of the links are increased first and then decreased to zero (see Figures 4c and 4d). The control signals produced by the network reach the maximal admissible torque of the manipulator in the beginning (see Figures 4e and 4f and Table 1). This allows the network to quickly steer the manipulator toward the desired positions. At almost halfway to the desired positions, the control signals change the sign to reduce the speeds of the links. When the manipulator is close to the desired positions, the control signals become very small, stabilizing the manipulator in the end. This kind of control strategy learned by the network shows that the proposed network model is able to solve the control task without a priori knowledge of the robotic manipulator.

5 Conclusion

The problem of controlling a class of robotic manipulator systems with unknown parameters and a dead zone is discussed in this letter. A neural network for reinforcement learning is presented to search for the optimal control policy. The network model comprises three distinctive subnetworks: the state network, used to predict the state of the robotic manipulator; the performance index, approximated by the critic network; and the action network, used to learn the optimal control policy by optimizing the performance index. Due to its learning capability, the network model is able to cope with the unknown dynamics and unknown physical parameters of the robotic manipulator. The convergence of the learning rules of the proposed model is proved based on the Lyapunov stability theory. The effectiveness of the approach is illustrated by simulation results of a two-link manipulator. The application of the proposed network model to control a robotic manipulator with more than two links will be investigated in the future.

Appendix: Proof of Weights Convergence

The Lyapunov function candidate of this model is defined as

$$V = V_1 + V_2 + V_3 + V_4$$

where $V_1 = \frac{1}{\eta_s} tr\{\omega_s^T \omega_s\}$, $V_2 = \frac{1}{\beta_1 \eta_a} tr\{\omega_a^T \omega_a\}$, $V_3 = \frac{1}{\beta_2 \eta_c} tr\{\omega_c^T \omega_c\}$, $V_4 = \frac{1}{2}\|\mu_c(k-1)\|^2$. $\beta_1, \beta_2 > 0$, and $\mu_c(k-1) = \omega_c^T \phi_c(k-1)$.

The first-order difference of the Lyapunov function candidate is written as

$$\Delta V = \Delta V_1 + \Delta V_2 + \Delta V_3 + \Delta V_4.$$

It remains to determine ΔV_1, ΔV_2, ΔV_3, and ΔV_4.

According to equation 3.5, ΔV_1 is given by

$$\begin{aligned}\Delta V_1 &= \frac{1}{\eta_s} tr\{\omega_s^T(k+1)\omega_s(k+1) - \omega_s^T(k)\omega_s(k)\}\\
&= \frac{1}{\eta_s} tr\{-2\eta_s\omega_s^T(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k)\\
&\quad + \|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k)\|^2\|\omega_s^T(k)\|^2\\
&= -(1-\eta_s)\|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k)\|^2\\
&\quad + \|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k) - \omega_s^T(k)\|^2\\
&\quad - \|\omega_s^T(k)\|^2.\end{aligned}$$

Due to the Cauchy-Schwarz inequality, ΔV_1 is bounded by

$$\begin{aligned}\Delta V_1 &\le -(1-\eta_s)\|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k)\|^2\\
&\quad + \|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k) - \omega_s^T(k)\|^2.\end{aligned}$$

According to equation 3.15, ΔV_2 is computed as

$$\begin{aligned}\Delta V_2 &= \frac{1}{\beta_1\eta_a} tr\{\omega_a^T(k+1)\omega_a(k+1) - \omega_a^T(k)\omega_a(k)\}\\
&= \frac{1}{\beta_1\eta_a} tr\{-2\eta_s\omega_a^T(k)\omega_c^T(k)\phi_c(k) + \eta_a^2\|\omega_c^T(k)\phi_c(k)\omega_a^T(k)\|^2\}\\
&= \frac{1}{\beta_1} tr\{-2\omega_a^T(k)\omega_c^T(k)\phi_c(k) + \eta_a\|\omega_c^T(k)\phi_c(k)\omega_a^T(k)\|^2\}\\
&= \frac{1}{\beta_1}\{-(1-\eta_a)\|\omega_c^T(k)\phi_c(k)\omega_a^T(k)\|^2 + \|\omega_c^T(k)\phi_c(k)\omega_a^T(k) - \omega_a^T(k)\|^2\\
&\quad - \|\omega_a^T(k)\|^2\}.\end{aligned}$$

Based on the Cauchy-Schwarz inequality, ΔV_2 is further bounded by

$$\Delta V_2 \le \frac{1}{\beta_1}\{-(1-\eta_a)\|\omega_c^T(k)\phi_c(k)\omega_a^T(k)\|^2 + \|\omega_c^T(k)\phi_c(k)\omega_a^T(k) - \omega_a^T(k)\|^2\}.$$

From equation 3.10, ΔV_3 can be written as

$$\begin{aligned}\Delta V_3 &= \frac{1}{\beta_2\eta_c} tr\{\omega_c^T(k+1)\omega_c(k+1) - \omega_c^T(k)\omega_c(k)\}\\
&= \frac{1}{\beta_2\eta_c} tr\{-2\gamma\eta_c\omega_c^T(k)\left(\gamma\omega_c^T(k)\phi_c(k) + r - \omega_c^T(k-1)\phi_c(k-1)\right)\phi_c(k)\end{aligned}$$

$$
\begin{aligned}
&+\gamma^2\eta_c^2\|\omega_c^T(k)\phi_c(k)+r-\omega_c^T(k-1)\phi_c(k-1)\|^2\|\phi_c(k)\|^2\}\\
=\;&\frac{1}{\beta_2\eta_c}tr\{-\gamma^2\|\mu_c(k)\|^2-\gamma^2\|\mu_c(k)\|^2(I-\gamma^2\eta_c\|\phi_c(k)\|^2)\\
&-2\gamma\left(\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\omega_c^T(k-1)\phi_c(k-1)\right)\phi_c^T(k)(I-\gamma^2\eta_c\|\phi_c(k)\|^2)\\
&\cdot\omega_c(k)+\gamma^2\eta_c\left(\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\omega_c^T(k-1)\phi_c(k-1)\right)\phi_c^T(k)\phi_c(k)\\
&\cdot\left(\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\omega_c^T(k-1)\phi_c(k-1)\right)^T\}\\
=\;&\frac{1}{\beta_2}tr\{-\gamma^2\|\mu_c(k)\|^2-\gamma^2(I-\gamma^2\eta_c\|\phi_c(k)\|^2)\|\mu_c(k)\\
&+\gamma^{-1}\left(\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\omega_c^T(k-1)\phi_c(k-1)\right)\|^2\\
&+\gamma^2(I-\gamma^2\eta_c\|\phi_c(k)\|^2)\|\gamma^{-1}\left(\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\omega_c^T(k-1)\phi_c(k-1)\right)\|^2\\
&+\gamma^2\eta_c\|\phi_c(k)\|^2\|\left(\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\omega_c^T(k-1)\phi_c(k-1)\right)\|^2\}\\
=\;&\frac{1}{\beta_2}\{-\gamma^2\|\mu_c(k)\|^2-\gamma^2(I-\gamma^2\eta_c\|\phi_c(k)\|^2)\|\mu_c(k)\\
&+\gamma^{-1}\left(\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\omega_c^T(k-1)\phi_c(k-1)\right)\|^2\\
&+\|\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\omega_c^T(k-1)\phi_c(k-1)\|^2\}.
\end{aligned}
$$

Applying the Cauchy-Schwarz inequality, an upper bound of ΔV_3 is

$$
\begin{aligned}
\Delta V_3\le\frac{1}{\beta_2}\Bigg\{&\frac{1}{2}\|\mu_c(k-1)\|^2-\gamma^2\|\mu_c(k)\|^2-\gamma^2(I-\gamma^2\eta_c\|\phi_c(k)\|^2)\\
&\cdot\|\mu_c(k)+\Delta\omega_c^T(k)\phi_c(k)+\gamma^{-1}r-\gamma^{-1}\omega_c^T(k-1)\phi_c(k-1)\|^2\\
&+2\|\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\frac{1}{2}\omega_c^T(k-1)\phi_c(k-1)\\
&-\frac{1}{2}\Delta\omega_c^T(k)\phi_c(k-1)\|^2\Bigg\}.
\end{aligned}
$$

ΔV_4 is simply given by

$$
\Delta V_4=\frac{1}{2}\left(\|\mu_c(k)\|^2-\|\mu_c(k-1)\|^2\right).
$$

Therefore, ΔV is bounded by

$$
\begin{aligned}
\Delta V&=\Delta V_1+\Delta V_2+\Delta V_3+\Delta V_4\\
&\le-(1-\eta_s)\|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k)\|^2
\end{aligned}
$$

$$
\begin{aligned}
&+ \|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k)-\omega_s^T(k)\|^2 \\
&+ \frac{1}{\beta_1}\{-(1-\eta_a)\|\omega_c^T(k)\phi_c(k)\omega_a^T(k)\|^2 + \|\omega_c^T(k)\phi_c(k)\omega_a^T(k)-\omega_a^T(k)\|^2\} \\
&+ \frac{1}{\beta_2}\Bigg\{-\gamma^2\|\mu_c(k)\|^2-\gamma^2(I-\gamma^2\eta_c\|\phi_c(k)\|^2) \\
&\cdot\|\mu_c(k)+\Delta\omega_c^T(k)\phi_c(k)+\gamma^{-1}r-\gamma^{-1}\omega_c^T(k-1)\phi_c(k-1)\|^2 \\
&+2\|\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\frac{1}{2}\omega_c^T(k-1)\phi_c(k-1)-\frac{1}{2}\Delta\omega_c^T(k)\phi_c(k-1)\|^2\Bigg\} \\
&+\frac{1}{2}\|\mu_c(k-1)\|^2+\frac{1}{2}\left(\|\mu_c(k)\|^2-\|\mu_c(k-1)\|^2\right) \\
=\;& -(1-\eta_s)\|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k)\|^2 \\
&-\frac{1}{\beta_1}(1-\eta_a)\|\omega_c^T(k)\phi_c(k)\omega_a^T(k)\|^2-\frac{1}{\beta_2}\gamma^2(I-\gamma^2\eta_c\|\phi_c(k)\|^2) \\
&\cdot\|\mu_c(k)+\Delta\omega_c^T(k)\phi_c(k)+\gamma^{-1}r-\gamma^{-1}\omega_c^T(k-1)\phi_c(k-1)\|^2 \\
&-\frac{1}{\beta_2}\gamma^2\|\mu_c(k)\|^2+\frac{1}{2}\left(\|\mu_c(k)\|^2-\|\mu_c(k-1)\right) \\
&+\|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k)-\omega_s^T(k)\|^2 \\
&+\frac{2}{\beta_2}\|\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\frac{1}{2}\omega_c^T(k-1)\phi_c(k-1)-\frac{1}{2}\Delta\omega_c^T(k)\phi_c(k-1)\|^2 \\
&+\frac{1}{\beta_1}\|\omega_c^T(k)\phi_c(k)\omega_a^T(k)-\omega_a^T(k)\|^2+\frac{1}{2\beta_2}\|\mu_c(k-1)\|^2 \\
=\;& -(1-\eta_s)\|(\omega_s^T(k)\phi_s(k)-x(k))\phi_s(k)\|^2 \\
&-\frac{1}{\beta_1}(1-\eta_a)\|\omega_c^T(k)\phi_c(k)\omega_a^T(k)\|^2-\frac{\beta_2-2\gamma^2}{2\beta_2}\|\mu_c(k)\|^2 \\
&-\frac{1}{\beta_2}\gamma^2(I-\gamma^2\eta_c\|\phi_c(k)\|^2) \\
&\cdot\|\mu_c(k)+\Delta\omega_c^T(k)\phi_c(k)+\gamma^{-1}r-\gamma^{-1}\omega_c^T(k-1)\phi_c(k-1)\|^2+\Delta F.
\end{aligned}
$$

Here, ΔF is defined as

$$
\begin{aligned}
\Delta F =\;& \|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k)-\omega_s^T(k)\|^2 \\
&+\frac{2}{\beta_2}\|\gamma\Delta\omega_c^T(k)\phi_c(k)+r-\frac{1}{2}\omega_c^T(k-1)\phi_c(k-1)-\frac{1}{2}\Delta\omega_c^T(k)\phi_c(k-1)\|^2
\end{aligned}
$$

$$+\frac{1-\beta_2}{2\beta_2}\|\mu_c(k-1)\|^2+\frac{1}{\beta_1}\|\omega_c^T(k)\phi_c(k)\omega_a^T(k)-\omega_a^T(k)\|^2.$$

Due to the Cauchy-Schwarz inequality, ΔF can be further bounded by

$$\begin{aligned}\Delta F \leq\ & \|\omega_c^T(k)\phi_c(k)\omega_c^T(k)\phi_c(k)\omega_c^T(k)\left(1+\omega_a^T(k)\phi_a(k)\omega_a^T(k)\right)\phi_s(k)\|^2\\ &+\|\omega_s^T(k)\|^2+\frac{1-\beta_2}{2\beta_2}\|\mu_c(k-1)\|^2+\frac{2}{\beta_2}\|\gamma\Delta\omega_c^T(k)\phi_c(k)\|^2\\ &+\frac{2}{\beta_2}\|r\|^2+\frac{1}{2\beta_2}\|\omega_c^T(k-1)\phi_c(k-1)\|^2+\frac{1}{2\beta_2}\|\Delta\omega_c^T(k)\phi_c(k-1)\|^2\\ &+\frac{1}{\beta_1}\|\omega_c^T(k)\phi_c(k)\omega_a^T(k)\|^2+\frac{1}{\beta_1}\|\omega_a^T(k)\|^2\\ \equiv\ & \Delta F_{\max}.\end{aligned}$$

Therefore, $\Delta F_{\max}$ is an upper bound of ΔF. The conditions for $\Delta F_{\max}$ to be negative are $0<\eta_s<1$, $\beta_1>0$, $0<\eta_a<1$, $\beta_2>2\gamma^2>0$, $\gamma>0$, $\gamma^2\eta_c\|\phi_c(k)\|^2<1$, $\|\mu_c\|^2>\frac{\beta_2-1}{2\beta_2}\Delta F_{\max}$.

Under these conditions, $\Delta V<0$ is always negative. According to the standard Lyapunov extension theorem (Khalil, 2002), the auxiliary error and the error in the weight updating are uniformly ultimately bounded, and the weights used in this letter are bounded, correspondingly. As a consequence, the algorithm proposed in this letter is convergent.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2016YFC0801808). We thank our reviewers for suggestions in reporting this work.

References

Alessandri, A. (2004). Adaptive neural network control of robotic manipulators. *Automatica*, *40*(11), 2011–2012.

Azeem, M. M., Iqbal, J., Toivanen, P., & Samad, A. (2012). Emotions in robots. *Communications in Computer and Information Science*, *281*, 144–153.

Baizid, K., Chellali, R., Yousnadj, A., Meddahi, Iqbal, J., & Bentaleb, T. (2010). Modelling of robotized site and simulation of robots optimum placement and orientation zone. In *Proceedings of the 21st IASTED International Conference on Modelling and Simulation*.

Chen, C. L., Liu, Y., & Wen, G. (2014). Fuzzy neural network-based adaptive control for a class of uncertain nonlinear stochastic systems. *IEEE Transactions on Systems, Man, and Cybernetics*, *44*(5), 583–593.

Chen, C. L., Wen, G., Liu, Y., & Wang, F. (2014). Adaptive consensus control for a class of nonlinear multiagent time-delay systems using neural network. *IEEE Transactions on Neural Networks*, *25*(6), 1217–1226.

Chen, W., & Jiao, L. (2010). Adaptive tracking for periodically time-varying and nonlinearly parameterized systems using multilayer neural networks. *IEEE Transactions on Neural Networks*, *21*(2), 345–351.

Chen, W., Jiao, L., Li, J., & Li, R. (2010). Adaptive NN backstepping output-feedback control for stochastic nonlinear strict-feedback systems with time-varying delays. *Systems Man and Cybernetics*, *40*(3), 939–950.

Chinta, L. V., & Tweed, D. B. (2012). Adaptive optimal control without weight transport. *Neural Computation*, *24*(6), 1487–1518.

Gao, W., & Selmic, R. R. (2006). Neural network control of a class of nonlinear systems with actuator saturation. *IEEE Transaction on Neural Networks*, *17*(1), 147–156.

Gosavi, A. (2014). Variance-penalized Markov decision processes: Dynamic programming and reinforcement learning techniques. *International Journal of General Systems*, *43*(6), 649–669.

Gu, S., Holly, E., Lillicrap, T., & Levine, S. (2016). *Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates*. arXiv:1610.006.33.

He, W., Dong, Y., & Sun, C. (2015) Adaptive neural impedance control of a robotic manipulator with input saturation. *IEEE Transactions on Systems, Man, and Cybernetics*, *46*(3), 334–344.

Huang, S., & Tan, K. K. (2012). Intelligent friction modeling and compensation using neural network approximations. *IEEE Transactions on Industrial Electronic*, *59*(8), 3342–3349.

Huang, X., Naghdy, F., Du, H., & Naghdy, G. (2015). Reinforcement learning neural network (RLNN) based adaptive control of fine hand motion rehabilitation robot. In *Proceedings of the IEEE Conference on Control Applications*. Piscataway, NJ: IEEE.

Iqbal, J., Pasha, S. M., Baizid, K., Khan, A., & Iqbal, J. (2013). Computer vision inspired realtime autonomous moving target detection, tracking and locking. *Life Science Journal*, *10*(4), 3338–3345.

Iqbal, J., Pasha, S. M., Nabi, U. R., Khan, N., & Tqbal, J. (2013). Real-time target detection and tracking: A comparative in-depth review of strategies. *Life Science Journal*, *10*(3), 804–813.

Iqbal, J., Rehmansaad, M., Malik, A., & Mahmood Tahir, A. (2014). State estimation technique for a planetary robotic rover. *Revista Facultad de Ingeniera*, *73*, 58–68.

Ka, H. W., Ding, D., & Cooper, R. A. (2016). Three dimensional computer vision-based alternative control method for assistive robotic manipulator. In *Proceedings of the International Journal of Advanced Robotics and Automation*, *1*(1), 1–6.

Khalil, H. K. (2002). *Nonlinear systems* (3rd ed.). Upper Saddle River. NJ: Prentice Hall.

Kim, C., Seo, Y., Lee, S., Choi, B., & Moon, J. (2015). Design of a heavy-duty manipulator for dismantling of a nuclear power plant. In *Proceedings of the International Conference on Control and Systems Automation* (pp. 1154–1158). Piscataway, NJ: IEEE.

Kim, Y. H., & Lewis, F. L. (2000). Optimal design of CMAC neural-network controller for robot manipulator. *IEEE Transactions on Systems, Man, and Cybernetics*, *30*(1), 22–31.

Leitner, J., Harding, S., Frank, M., Förster, A., & Schmidhuber, J. (2013). Artificial neural networks for spatial perception: Towards visual object localisation in humanoid robots. In *Proceedings of the International Joint Conference on Neural Networks*. Piscataway, NJ: IEEE.

Levine, S., & Koltun, V. (2013). Guided policy search. In *Proceedings of the International Conference on Machine Learning* (pp. 1–9).

Lewis, F. L., Vrabie, D., & Syrmos, V. L. (2012). *Optimal control* (3rd ed.). New York: Wiley.

Lewis, F. L., & Liu, D. R. (2013). *Reinforcement learning and approximate dynamic programming for feedbeck control*. Piscataway, NJ: IEEE Press.

Li, Z., Cao, X., & Ding, N. (2011). Adaptive fuzzy control for synchronization of nonlinear teleoperators with stochastic time-varying communication delays. *IEEE Transactions on Fuzzy Systems*, *19*(4), 745–757.

Li, Z., Ge, S. S., Adams, M., & Wijesoma, W. S. (2008). Robust adaptive control of uncertain force/motion constrained nonholonomic mobile manipulators. *Automatica*, *44*(3), 776–784.

Lin, C. (2009). H_∞ reinforcement learning control of robot manipulators using fuzzy wavelet networks. *Fuzzy Sets and Systems*, *160*(12), 1765–1786.

Liu, D. R., Wei, Q. L., Wang, D., Yang, X., & Li, H. L. (2013). *Adaptive dynamic programming with applications in optimal control*. Berlin: Springer.

Liu, G., Kadirkamanathan, V., & Billings, S. A. (1999). Variable neural networks for adaptive control of nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics*, *29*(1), 34–43.

Liu, Y., Chen, C. L., Wen, G., & Tong, S. (2011). Adaptive neural output feedback tracking control for a class of uncertain discrete time nonlinear systems. *IEEE Transactions on Neural Networks*, *22*(7), 1162–1167.

Liu, Y., Tang, L., Tong, S., Chen, C. L., & Li, D. (2015). Reinforcement learning design-based adaptive tracking control with less learning parameters for nonlinear discrete-time MIMO systems. *IEEE Transactions on Neural Network and Learning Systems*, *26*(1), 165–176.

Liu, Y., Tang, L., Tong, S., & Chen, C. L. (2015). Adaptive NN controller design for a class of nonlinear MIMO discrete-time systems. *IEEE Transactions on Neural Networks*, *26*(5), 1007–1018.

Liu, Y., Tong, S., Wang, D., Li, T., & Chen, C. L. (2011). Adaptive neural output feedback controller design with reduced-order observer for a class of uncertain nonlinear SISO systems. *IEEE Transactions on Neural Networks*, *22*(8), 1328–1334.

Liu, Z., Chen, C., Zhang, Y., & Chen, C. L. (2015). Adaptive neural control for dual-arm coordination of humanoid robot with unknown nonlinearities in output mechanism. *IEEE Transactions on Systems, Man, and Cybernetics*, *45*(3), 507–518.

Min, H., Lu, J., Xu, S., Duan, N., & Chen, W. (2017). Neural network-based output-feedback control for stochastic high-order non-linear time-delay systems with application to robot system. *IET Control Theory and Applications*, *11*(10), 1578–1588.

Naitchabane, K., Hoppenot, P., & Colle, E. (2016). Design of a heavy-duty manipulator for dismantling of a nuclear power plant. In *Proceedings of the International Conference on Informatics in Control, Automation and Robotics*. Berlin: Springer.

Naveed, K., Iqbal, J., & Ur Rehman, H. (2012). Brain controlled human robot interface. In *Proceedings of the International Conference on Robotics and Artificial Intelligence*. Piscataway, NJ: IEEE.

Prokhorov, D., & Wunsch, D. C. (1997). Adaptive critic designs. *IEEE Transactions on Neural Network*, *8*(5), 997–1007.

Purwar, S., Kar, I. N., & Jha, A. N. (2005). Adaptive control of robot manipulators using fuzzy logic systems under actuator constraints. *Fuzzy Sets and Systems*, *152*(3), 651–664.

Rossomando, F. G., Soria, C., & Carelli, R. (2013). Neural network-based compensation control of mobile robots with partially known structure. *IET Control Theory and Application*, *6*(12), 1851–1860.

Runa, J., & Sharma, R. (2015). A Lyapunov theory based adaptive fuzzy learning control for robotic manipulator. In *Proceedings of the International Conference on Recent Developments in Control, Automation and Power Engineering*. Piscataway, NJ: IEEE.

Sanner, R. M., & Slotine, J. E. (1995). Stable adaptive control of robot manipulator using neural networks. *Neural Computation*, *7*(4), 753–790.

Schindele, D., & Aschemann, H. (2009). Adaptive friction compensation based on the LuGre model for a pneumatic rodless cylinder. In *Proceedings of the IEEE Conference of Industrial Electronics*. Piscataway, NJ: IEEE.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

Seidl, D. R., Lam, S., Putnam, J., & Lorenz, R. D. (1993). Neural network compensation of gear backlash hysteresis in position-controlled mechanisms. *IEEE Transactions on Industrial Application*, *31*(6), 2027–2034.

Sharma, R., & Gopal, M. (2010). Synergizing reinforcement learning and game theory: A new direction for control. *Applied Soft Computing*, *10*(3), 675–688.

Si, J., Barto, A., Powell, W., & Wunsch, D. (2004). *Handbook of learning and approximate dynamic programming: An introduction*. New York: Wiley–IEEE Press.

Si, J., Barto, A., Powell, W., & Wunsch, D. (2012). *Supervised actor critic reinforcement learning: Handbook of learning and approximate dynamic programming*. New York: Wiley.

Sun, T., Pei, H., Pan, Y., Zhou, H., & Zhang, C. (2011). Neural network-based sliding mode adaptive control for robot manipulators. *Neurocomputing*, *74*(14), 2377–2384.

Sutton, R. S., & Barto, A. G. (2017). *Reinforcement learning: An introduction* (2nd ed.). Cambridge, MA: MIT Press.

Szepesvari, C. (2010). *Algorithm for reinforcement learning*. London: Morgan & Claypool.

Tao, G., & Kokotovic, P. V. (1994). Adaptive control of plants with unknown deadzones. *IEEE Transactions on Automatic Control*, *39*(1), 59–68.

Tang, L., Liu, Y., & Tong, S. (2014). Adaptive neural control using reinforcement learning for a class of robot manipulator. *Neural Computing and Applications*, *25*(1), 135–141.

Tong, S., Wang, T., Li, Y., & Chen, B. (2013). A combined backstepping and stochastic small-gain approach to robust adaptive fuzzy output feedback control. *IEEE Transactions on Fuzzy Systems*, *21*(2), 314–327.

Wang, G., Sun, T., Pan, Y., & Yu, H. (2017). Adaptive neural network control for constrained robot manipulators. In F. Cong, A. Leung, & Q. Wei (Eds.), *Lecture Notes in Computer Science: Vol. 10262. Advances in Neural Networks*. Berlin: Springer.

Wang, H. (2010). On the recursive implementation of adaptive control for robot manipulators. In *Proceedings of the IEEE Control Conference*. Piscataway, NJ: IEEE.

Wang, F. Y., Zhang, H. G., & Liu, D. R. (2009). Adaptive dynamic programming: An introduction. *IEEE Computational Intelligence Magazine*, *4*(2), 39–47.

Wang, Z. H., Yan, B., Cai, L., & Zhang, S. S. (2006). Robust adaptive deadzone compensation of DC servo system. *IEEE Proceedings—Control Theory and Applications*, *153*(6), 709–713.

Werbos, P. J. (2009). Intelligence in the brain: A theory of how it works and how to build it. *Neural Networks*, *22*(3), 200–212.

Yahya, A., Li, A., Kalakrishnan, M., Chebotar, Y., & Levine, S. (2016). *Collective robot reinforcement learning with distributed asynchronous guided policy search*. arXiv:1610.00673v1.

Yang, Q., & Jagannathan, S. (2012). Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators. *IEEE Transactions on Systems, Man, and Cybernetics*, *42*(2), 377–390.

Zareinia, K., & Sepehri, N. (2015). A hybrid haptic sensation for teleoperation of hydraulic manipulators. *Journal of Dynamic Systems Measurement and Control Transactions of the ASME*, *137*(9), 316–317.

Received October 4, 2017; accepted January 26, 2018.