Neural Networks Enhanced Optimal Admittance Control of Robot–Environment Interaction Using Reinforcement Learning

Guangzhu Peng, C. L. Philip Chen, *Fellow, IEEE*, and Chenguang Yang, *Senior Member, IEEE*

***Abstract*— In this paper, an adaptive admittance control scheme is developed for robots to interact with time-varying environments. Admittance control is adopted to achieve a compliant physical robot–environment interaction, and the uncertain environment with time-varying dynamics is defined as a linear system. A critic learning method is used to obtain the desired admittance parameters based on the cost function composed of interaction force and trajectory tracking without the knowledge of the environmental dynamics. To deal with dynamic uncertainties in the control system, a neural-network (NN)-based adaptive controller with a dynamic learning framework is developed to guarantee the trajectory tracking performance. Experiments are conducted and the results have verified the effectiveness of the proposed method.**

***Index Terms*— Adaptive control, admittance control, neural networks (NNs), reinforcement learning (RL), robot–environment interaction.**

I. Introduction

OVER the past few decades, with the great progress of new technology, the application fields of robots are becoming more extensive and extending to our social life. In most practical applications, robots inevitably interact with the environment. Due to the uncertainties of the environment [1], control of the robot in the environment becomes a challenging work. In order to achieve reliable performance, two major difficulties in robot interaction control need to be noted and solved, which is ensuring the safety of interaction control between the robot and the environment and enables the robot to adapt to the changing environmental dynamics.

In terms of the research works on interactive control, two main approaches have been widely used, i.e., hybrid position/force control and impedance control [2], [3]. Compared with hybrid position/force, which needs to decompose force and position into different directions, impedance control is more popular because of its good robustness and practicability. Through impedance control, the robot can interact with the environment in a compliant behavior to ensure safety. On the basis of cause-and-effect relationship, the admittance control is regarded as position-based impedance control [4]. Admittance control achieves an effect of compliant control performance by changing the reference trajectory [5]. A desired control performance is guaranteed through imposing a proper admittance model, which relates the external force to the manipulator position. Thus, selection of admittance model parameters appears particularly important. Some early works focus on dealing with the uncertainties of system dynamics with predefined impedance parameters [6]. However, because of the unpredictability and uncertainty of the environment [7], it is difficult to ensure that robots can achieve good interaction performance in unknown environments by traditional impedance control. To solve this, impedance learning has been proposed and studied [8], [9].

Optimization for impedance learning is essential in research works since the control objective consists of trajectory tracking and regulation of the interaction force. In [10] and [11], the linear quadratic regulator (LQR) was introduced to obtain the desired impedance parameters through a cost function. The disadvantage of the traditional LQR method is that the dynamics of the environment must be completely known. To achieve an optimal control performance under unknown environment dynamics, adaptive dynamic programming (ADP) has been extensively studied [12]–[15]. It simulates the idea of human learning through environmental feedback, and it is considered to be a close way to human intelligence. Executing a task using ADP method, the agent can judge whether the action is effective and update the control strategy according to feedback rewards [16]–[19]. Due to its special critic–actor structure, ADP has become a good method to solve the optimal control of nonlinear systems whose information is completely or partially unknown. Some existing works have been done

Manuscript received May 17, 2020; revised September 3, 2020 and January 8, 2021; accepted January 27, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1703600; in part by the National Natural Science Foundation of China under Grant 61702195, Grant 61751202, Grant U1813203, Grant U1801262, Grant 61751205, Grant U20A20200, and Grant 61861136009; in part by the Science and Technology Major Project of Guangzhou under Grant 202007030006; in part by the Science and Technology Development Fund, Macau, under Grant 0119/2018/A3; and in part by the Multiyear Research Grants of University of Macau. *(Corresponding author: C. L. Philip Chen.)*

Guangzhu Peng is with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau 999078, China (e-mail: gz.peng@qq.com).

C. L. Philip Chen is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510641, China, also with the Pazhou Laboratory, Guangzhou 510335, China, and also with the Faculty of Science and Technology, University of Macau, Macau 999078, China (e-mail: philip.chen@ieee.org).

Chenguang Yang is with the School of Automation Science and Engineering, South China University of Technology, Guangzhou 510641, China (e-mail: cyang@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3057958.

Digital Object Identifier 10.1109/TNNLS.2021.3057958

to apply ADP to the impedance control of robots [20], [21]. In these works, reinforcement learning (RL) was used to update the impedance variables according to the specified task. Unfortunately, most learning processes require robots to repeat tasks to obtain the desired impedance parameters [22], [23]. To copy with this problem, the impedance adaptation method has been proposed without repeating the operation and achieved some satisfactory results [24]. However, as pointed out in [25], most physical systems are time-varying dynamics [26]. Therefore, the impedance adaptation in [24] is not applicable in a time-varying parametric system because it assumes that the parameters of the environment are time-invariant.

It should be noted that trajectory tracking is important in admittance control schemes. In practice, uncertainties exist in system dynamics [27]–[29], which makes it difficult for the traditional model-based control methods to achieve a good control performance [30]. To solve this, adaptive control approaches with intelligent tools were adopted in control systems [31]–[35]. The NN-based adaptive controller is one of the most successful intelligent controllers, which uses the NNs to deal with system uncertainties. In [36], uncertainties of the object and the robot were compensated by the radial basis function neural networks (RBFNNs). In [37], NNs combined with the fuzzy logic were used to estimate the unknown dynamics of the control system. In [38], NNs were employed to copy with output nonlinearities. However, some drawbacks of traditional NN-based control methods should be paid attention. With more neural nodes, the function approximation performance will be better, and the desired control performance can be achieved, but too many nodes will in turn increase the computational load of the system and degrade the control performance. Moreover, if NN learning is insufficient, i.e., centers of neural nodes are far away from the input, the desired function approximation performance could not be achieved.

Recently, the broad learning system (BLS) has been widely studied [39]–[41]. Compared with traditional multilayer networks, the BLS with a novel structure can achieve a satisfactory recognition accuracy without suffering the time-consuming training process. In [42] and [43], it has been pointed out that NNs can accumulate the learning knowledge and store them in nodes under the persistent excitation (PE) condition. Based on the deterministic learning theory and the BLS structure, NN-based control with adaptive learning framework has been studied. In [44], a novel framework of BLS-based adaptive controller was proposed to approximate uncertainties and the controller can increment nodes automatically. However, increasing the number of nodes will not only bring a lot of computing burden to the system but also may lead to instability of the system. To solve this problem, in this paper, we have developed the RBFNN learning framework, which can not only automatically increment NN nodes but also discard nodes with little activation response. Then, the effectiveness and efficiency of the RBFNN learning control system can be improved.

The main contributions of this paper are highlighted as follows.

1) The environment with unknown time-varying parameters is considered for analysis and defined as a linear model to be interacted with the robotic arm.
2) The critic learning method is developed for optimal admittance adaptation of robots.
3) An RBFNN-based adaptive controller with a dynamic learning framework is employed to deal with the system uncertainties and guarantee the tracking performance.

The rest of this paper is organized as follows. We will start with some related and prepared works about the interactive control framework in Section II. Next, we give the optimal admittance adaptation method and the design of the tracking controller. The experiments are conducted in Section IV. Section V draws the conclusion of this paper.

II. Problem Formulation and Preliminaries

A. Problem Formulation

Considering that a robotic arm interacts with an unknown environment, the designed control scheme should meet the following requirements: 1) the robotic arm can achieve an optimal behavior in interaction with the environment and 2) desired tracking performance can be ensured by the adaptive controller and uncertainties of the robot system can be approximated.

B. System Dynamics

The forward kinematics of the robot is defined as

$$x = \kappa(q) \tag{1}$$

where $x \in \mathbb{R}^{N_0}$ represents the position of the robot end-effector with the degree of freedom N_0. The joint angles of an N-joint manipulator are denoted by $q \in \mathbb{R}^N$ in joint space. $\kappa(\cdot)$ represents the function of the forward kinematics. According to (1), we can obtain the following relation:

$$\dot{x} = J(q)\dot{q} \quad \ddot{x} = \dot{J}(q)\dot{q} + J(q)\ddot{q} \tag{2}$$

where $J(q)$ is the Jacobian matrix.

The dynamic model of the manipulator is

$$D_q(q)\ddot{q} + C_q(q, \dot{q})\dot{q} + G_q(q) = \tau_c + J^{\mathrm{T}} f_e \tag{3}$$

where $D_q(q) \in \mathbb{R}^{N\times N}$, $C_q(q, \dot{q}) \in \mathbb{R}^{N\times N}$, and $G_q(q) \in \mathbb{R}^{N\times N}$ are the inertial matrix, Coriolis matrix, and gravity load, respectively. τ_c is the joint torque and f_e is the external force from the environment.

The environmental dynamics in discrete time (DT) can be modeled as [45]

$$x_{k+1} = A_e x_k + B_e f_{ek} \tag{4}$$

where A_e and B_e are unknown matrices with time-varying parameters.

C. Admittance Control

Admittance control is an efficient way for robots to achieve a compliant interaction with the environment. To implement admittance control, a desired admittance model is required.

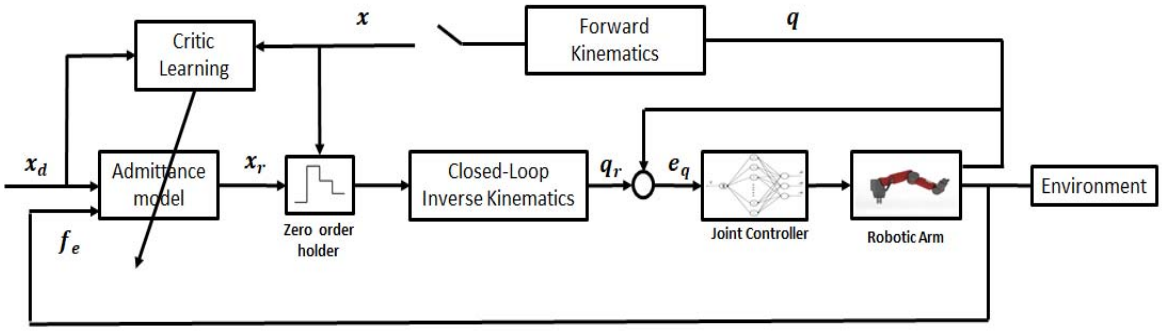

Fig. 1. Control diagram.

An admittance model in the Cartesian space can be expressed as

$$f_e(k) = \varphi(x_r(k), x_d(k)) \tag{5}$$

where $x_d(k)$ is the desired trajectory specified by designer and $x_r(k)$ is the modified desired trajectory. $\varphi(\cdot)$ denotes the admittance function of the relation between $x_d(k)$ and $x_r(k)$. By using the inverse kinematics as $q(k) = \kappa^{-1}(x(k))$, the corresponding trajectory in joint space is generated $q_r(k) = \kappa^{-1}(x_r(k))$ with the interaction force $f_e \in \mathbb{R}^{N_0}$ and the admittance model defined in (5).

As discussed above, a proper admittance model can guarantee a desired interaction performance, but it is hard to obtain subject to the time-varying environmental dynamics. To solve this, the critic learning method is used to realize admittance adaptation, and an optimal admittance model will be obtained, as shown in Fig. 1. After generating the trajectory q_r in joint space, the adaptive RBFNN learning controller is designed for trajectory tracking: $\lim_{k\to+\infty} q(k) \to q_r(k)$. If the ideal tracking performance is achieved: $q(k) = q_r(k)$ and $x(k) = x_r(k)$, then the admittance model becomes

$$f_e(k) = \varphi(x(k), x_d(k)). \tag{6}$$

D. Dynamic RBFNN Learning Framework

1) RBFNN: Consider a continuous vector function $F(Z_n) = [f_1(Z_n), f_2(Z_n), \ldots, f_n(Z_n)]^{\mathrm{T}} \in \mathbb{R}^n$, and

$$\hat{F}(Z_n) = \hat{W}^{\mathrm{T}} S(Z_n) \tag{7}$$

where $\hat{F}(Z_n)$ is the estimate of $F(Z_n)$, $Z_n \in \Omega_{Z_n} \subset \mathbb{R}^L$, and the L is the input dimension; $\hat{W}^{\mathrm{T}} = [\hat{W}_1, \hat{W}_2, \ldots, \hat{W}_n] \in \mathbb{R}^{n\times m}$ is the estimate of the ideal NN weight W^*, and m is the number of NN nodes; and $S_{Z_n} = [s_1(Z_n), s_2(Z_n), \ldots, s_m(Z_n)]^{\mathrm{T}} \in \mathbb{R}^m$ is the regressor vector. In RBFNN, $s(\cdot)$ is the activation function and commonly employed the Gaussian function as

$$S_i(Z_n) = \exp\left[\frac{-(Z_n - \theta_i^{\mathrm{T}})(Z_n - \theta_i)}{\sigma_i^2}\right], \quad i = 1, \ldots, m \tag{8}$$

where $\theta_i = [\theta_{i1}, \theta_{i2}, \ldots, \theta_{im}]^{\mathrm{T}}$ is the center of NN nodes, and σ_i is the variance of each NN node.

In [46], RBFNN has been proved to be able to approximate any continuous function as

$$F(Z_n) = W^{*\mathrm{T}} S(Z_n) + \varepsilon(Z_n) \quad \forall Z_n \in \Omega_{Z_n} \tag{9}$$

where W^* is the ideal weight vector and $\varepsilon(Z_n)$ is the approximation error. The weight estimation error is $\tilde{W} = \hat{W} - W^*$.

Definition 1 *[42]:* For approximating functions by the RBFNN in (7), we choose

$$\Theta \geq h := \frac{1}{2} \min_{i\neq j} ||\theta_i - \theta_j|| > 0. \tag{10}$$

Within this lattice, there exists a subvector $S_\xi(Z)$ of RBF

$$S_\xi(Z) = \left[s(Z_1), \ldots, s\left(Z_{m_\xi}\right)\right]. \tag{11}$$

Since the RBF is small with centers far away from trajectory, we can select

$$2h \leq \Theta \leq \Theta'. \tag{12}$$

For all the Z_i $(i = 1, 2, \ldots, m_\xi)$ satisfying $||Z_i - \theta_i|| < \Theta'$, we can obtain $|s(\cdot)| > \Theta''$, where Θ'' is a small constant.

Lemma 1 *[42]:* Consider the periodic-like trajectory $Z(t) \subset R^L$, the centers of RBFNN in (7) are placed on the lattice (large enough to cover Ω_{Z_n}) defined in (11) and (12), which can be said to be persistently exciting.

2) Dynamic RBFNN Learning Framework: In a deterministic learning theory, it has been pointed out that approximation of unknown functions can be achieved by local RBFNN ($\hat{W}^{\mathrm{T}} S$) with the convergence of tracking errors [42]. However, activation responses are small for the NN nodes whose centers are far from the input, as shown in Fig. 2. When this happens, the learning ability of the RBFNN could not guarantee the desired approximation performance since not enough NN nodes can be used to accumulate the learning the ability.

To solve this, the developed RBFNN learning framework can automatically place new NN nodes with centers close to input trajectory and discard the NN nodes that have little responses to the activation function during the learning period, as shown in Fig. 2. After that, the network will have a compact set Ω_{N1} to cover the input trajectory. According to *Lemma 1*, the dynamic RBFNN could achieve the desired approximation performance.

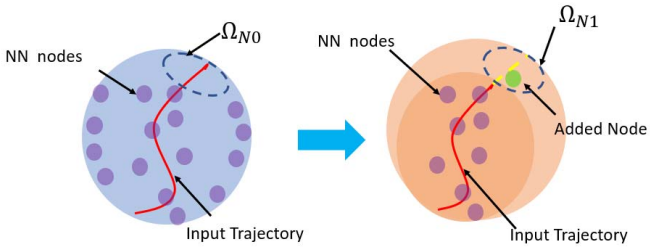

Fig. 2. Dynamic RBFNN learning framework (modified from [44]).

Here, we define the center position of existing NN nodes closest to the input trajectory as $[\Delta_1, \Delta_2, \ldots, \Delta_{m'}] \in \psi$, and the information of the added NN node

$$\begin{aligned} \theta_e &= \overline{\Delta} + \gamma \, ||Z_\xi - \overline{\Delta}|| \\ \sigma_e &= \sigma_i \\ W_e &= 0 \end{aligned} \tag{13}$$

where $0 < \gamma < 1$, Z_ξ is the position of input trajectory, $\overline{\Delta}$ is the average closest center position of existing NN nodes, that is

$$\overline{\Delta} = \frac{\sum_{k=1}^{m'} \Delta_i}{m'}, \quad i = 1, 2, \ldots m'. \tag{14}$$

Then, the new center vector can be represented as

$$\theta_{i+1} = \begin{cases} [\theta_i, \theta_e], & \text{if } \overline{\Delta} > \mu \\ \theta_i, & \text{otherwise} \end{cases} \tag{15}$$

where μ is the predefined threshold. After that, the weight of RBFNN can be represented as

$$W_1 = \begin{bmatrix} W \\ W_a \end{bmatrix} \tag{16}$$

where W_a is the weight of the new node.

III. Control Design

A. LQR Solution for Time-Varying System

In this section, the LQR method to solve the time-varying problem is introduced [18]. First, a linear system in DT domain is considered

$$\zeta_{k+1} = A_k \zeta_k + B_k u_k \tag{17}$$

where ζ_k and u_k denote the system state and input and A_k and B_k are the system matrix and input matrix with time-varying parameters, respectively.

The objective of interaction control is to minimize the following cost-to-go function:

$$V(\zeta_k) = \sum_{i=k}^{\infty} \beta^{i-k} r(\zeta_i, u_i) \tag{18}$$

where $0 < \beta \leq 1$, and the discount factor means that the future cost are less concerned about. A standard quadratic form is

$$r(\zeta_k, u_k) = \zeta_k^{\mathrm{T}} S \zeta_k + u_k^{\mathrm{T}} R \; u_k \tag{19}$$

where $S = S^{\mathrm{T}} \geq 0$ and $R = R^{\mathrm{T}} > 0$ are the weight matrices of system state and control input. By writing (18) as

$$V(\zeta_k) = r(\zeta_k, u_k) + \sum_{i=k+1}^{\infty} \beta^{i-(k+1)} r(\zeta_i, u_i) \tag{20}$$

the nonlinear function is known as the Bellman equation.

Design a control feedback

$$u_k = \iota \zeta_k \tag{21}$$

where ι is the feedback gain to be determined. Given the control input (21), the cost-to-go function can be minimized.

In [18], it has pointed out that optimal value of the current state is a quadratic form with an existing u_k^*, and the cost function can be

$$V^*(\zeta_k) = \zeta_k^{\mathrm{T}} P \zeta_k \tag{22}$$

where the matrix P is to be determined. Then, the Bellman equation becomes

$$\zeta_k^{\mathrm{T}} P \zeta_k = \zeta_k^{\mathrm{T}} S \zeta_k + u_k^{\mathrm{T}} R \; u_k + \zeta_{k+1}^{\mathrm{T}} P \zeta_{k+1}. \tag{23}$$

Consider the control law (21) and the system dynamics, and (23) becomes

$$\zeta_k^{\mathrm{T}} P \zeta_k = (A_k \zeta_k + B_k u_k)^{\mathrm{T}} P (A_k \zeta_k + B_k u_k) + \zeta_k^{\mathrm{T}} S \zeta_k + u_k^{\mathrm{T}} R \; u_k. \tag{24}$$

Taking the derivative of (24) with respect to u_k, the feedback gain is

$$\iota_k = -\left[R + B_k^{\mathrm{T}} P_k B_k\right]^{-1} \times B_k^{\mathrm{T}} P_k A_k. \tag{25}$$

Substituting (25) into the Bellman equation, we can have

$$A_k^{\mathrm{T}} P_k A_k - P_k + S - A_k^{\mathrm{T}} P_k B_k \left[R + B_k^{\mathrm{T}} P_k B_k\right]^{-1} B_k^{\mathrm{T}} P_k A_k = 0. \tag{26}$$

In [47], an HDP method was proposed to obtain the solution for algebraic Riccati equation

$$\begin{aligned} P_{k+1} &= A_k^{\mathrm{T}} P_k A_k + S - A_k^{\mathrm{T}} P_k B_k \left[R + B_k^{\mathrm{T}} P_k B_k\right]^{-1} B_k^{\mathrm{T}} P_k A_k \\ P(0) &= 0. \end{aligned} \tag{27}$$

B. RL With Q-Function

In this section, the value iteration method with Q-function approximation method is used to solve the problem for time-varying system [18], [48].

According to (23), we can define the Q function as

$$Q^*(\zeta_k, u_k) = \zeta_k^{\mathrm{T}} S \zeta_k + u_k^{\mathrm{T}} R \; u_k + \zeta_{k+1}^{\mathrm{T}} P \zeta_{k+1}. \tag{28}$$

Substituting system dynamics into (28)

$$\begin{aligned} Q^*(\zeta_k, u_k) &= \begin{bmatrix} \zeta_k \\ u_k \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} S & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} \zeta_k \\ u_k \end{bmatrix} + \begin{bmatrix} \zeta_k \\ u_k \end{bmatrix}^{\mathrm{T}} \\ &\quad \times \begin{bmatrix} A_k^{\mathrm{T}} \\ B_k^{\mathrm{T}} \end{bmatrix} P_{k+1} \begin{bmatrix} A_k^{\mathrm{T}} \\ B_k^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \zeta_k \\ u_k \end{bmatrix} \\ &= z_k^{\mathrm{T}} H_k z_k \end{aligned} \tag{29}$$

where $z_k = [\zeta_k \ u_k]^{\mathrm{T}}$. By using the Kronecker product, we define $\overline{H}_k = \mathrm{vec}(H_k)$ and

$$\overline{z}_k = z_k \otimes z_k = \begin{bmatrix} \zeta_k \\ u_k \end{bmatrix} \otimes \begin{bmatrix} \zeta_k \\ u_k \end{bmatrix}.$$

Then, (29) can be written as

$$\begin{aligned} Q^*(\zeta_k, u_k) &= \overline{H}_k^{\mathrm{T}} \overline{z}_k \\ &= \zeta_k^{\mathrm{T}} S \zeta_k + u_k^{\mathrm{T}} R \ u_k + \overline{H}_{k+1}^{\mathrm{T}} \overline{z}_{k+1} \end{aligned} \tag{30}$$

with $u_k = \iota \zeta_k$.

Algorithm 1 Value Iteration Algorithm

1: **Initialization** Select a control policy $\phi_0(\zeta_k)$

2: **Value Iteration** Determine the solution of Q_{j+1} to

$$\begin{aligned} Q_{j+1}(\zeta_k, u_k) &= r(\zeta_k, \ u_k) \\ &\quad + Q_j(\zeta_{k+1}, \phi_{k+1}). \end{aligned} \tag{31}$$

3: **Policy Improvement** Determine the control policy by

$$\phi_{j+1}(\zeta_k) = arg \min_{\phi(\cdot)} \big(Q_{j+1}(\zeta_k, u_k)\big). \tag{32}$$

In the LQR case, (29) can be rewritten as

$$\begin{aligned} Q^*(\zeta_k, u_k) &= z_k^{\mathrm{T}} H_k z_k \\ &= \begin{bmatrix} \zeta_k \\ u_k \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} H_{\zeta\zeta} & H_{\zeta u} \\ H_{u\zeta} & H_{uu} \end{bmatrix} \begin{bmatrix} \zeta_k \\ u_k \end{bmatrix}. \end{aligned} \tag{33}$$

The optimal control policy is based on

$$\frac{\partial}{\partial u_k} Q(\zeta_k, u_k) = 0. \tag{34}$$

Then, we have

$$\begin{aligned} 0 &= H_{u\zeta}\zeta_k + H_{uu} u_k \\ u_k &= -H_{uu}^{-1} H_{u\zeta} \zeta_k. \end{aligned} \tag{35}$$

In the LQR method, we can have the following relation:

$$\begin{aligned} H_{\zeta\zeta} &= A_k^{\mathrm{T}} P_k A_k + S \\ H_{u\zeta} &= H_{\zeta u}^{\mathrm{T}} = A_k^{\mathrm{T}} P_k B_k \\ H_{uu} &= B_k^{\mathrm{T}} P_k B_k + R. \end{aligned} \tag{36}$$

In the LQR case, we can obtain the optimal feedback gain under the condition that the system dynamics are known. However, in most situations, the environmental dynamics are unknown. To solve this, NN approximation method is used.

According to (31), an NN-based approximator is employed as

$$Q_{j+1}(\zeta_k, u_k) = \mathscr{W}_k^{\mathrm{T}} \vartheta_k \tag{37}$$

where $\vartheta_k = z_k \otimes z_k$ is the regression vector, and vector of $\mathscr{W}_k$ with time-varying parameters is to be determined. In this paper, the exponentially weighted recursive least-squares (EWRLS) method [48], [49] is employed to determine the time-varying parameter vector.

By applying the EWRLS method, the following mean squared error of loss function can be minimized:

$$E(\vartheta, k) = \frac{1}{2} \sum_{i=1}^{k} \beta^{k-i} \big(g(i) - \mathscr{W}_i^{\mathrm{T}} \vartheta_i\big)^2 \tag{38}$$

where $g(i) = r(\zeta(i), u(i)) + \overline{Q}_j(\zeta(i+1), \phi(i+1))$.

The recursive method to identify $\mathscr{W}_k$ is given as follows:

$$\hat{\mathscr{W}}_{k+1} = \hat{\mathscr{W}}_k + \mathcal{F}_{k+1}\big(g_{k+1} - \hat{\mathscr{W}}_k^{\mathrm{T}} \vartheta_{k+1}\big) \tag{39}$$

where the gain matrix F_{k+1} is determined by

$$\begin{aligned} \mathcal{F}_{k+1} &= \omega_{k+1} \vartheta_{k+1} \\ &= \omega_k \vartheta_{k+1} \big(\beta I + \vartheta_{k+1}^{\mathrm{T}} \omega_k \vartheta_{k+1}\big)^{-1} \\ \omega_{k+1} &= \frac{1}{\beta}\big(I - \omega_{k+1} \vartheta_{k+1}^{\mathrm{T}}\big)\omega_k \end{aligned} \tag{40}$$

where ω_k is the covariance matrix. To avoid the singularity of ω_k, we can define

$$\omega_k = K_\beta I \quad \text{if } \zeta_{min}(\omega_k) \le K'_\beta \tag{41}$$

where K_β and K'_β are positive constant, and $\zeta(\cdot)$ denotes the eigenvalue function.

Remark 1: The selection of the covariance matrix ω_k in (41) is to avoid its unlimited growth, and an unproper selection could result in a large estimation error. The similar trick has also been used and studied in [50].

Remark 2: To ensure the parameter convergence, exploration noise should be included in control input to satisfy the PE condition [51]

$$u_k = \iota_k \zeta_k + e_k \tag{42}$$

where e_k denotes the zero-mean white noise.

Given an optimal control policy, (29) is equal to (22), and we can have the following relationship:

$$\begin{bmatrix} \zeta_k \\ u_k \end{bmatrix}^{\mathrm{T}} H_k \begin{bmatrix} \zeta_k \\ u_k \end{bmatrix} = \zeta_k^{\mathrm{T}} P_k \zeta_k. \tag{43}$$

With the control law $u_k = \iota \zeta_k$, we have

$$P_k = \begin{bmatrix} I & \iota_k^{\mathrm{T}} \end{bmatrix} H_k \begin{bmatrix} I & \iota_k^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}. \tag{44}$$

Lemma 2 [48]: By using the iteration method (31) and (32), $Q(\zeta_k, u_k)$ can converge to its optimal value $Q^*(\zeta_k, u_k^*)$, and $\phi_j(\zeta_k) = \overline{\iota}_{j,k}\zeta_k$ will converge to its optimal value $u^*(\zeta_k)$.

Proof: From (30), we can obtain

$$\begin{aligned} Q_{j+1}(\zeta_k, u_k) &= z_k^{\mathrm{T}} \overline{H}_{j+1,k} z_k \\ Q_j\big(\zeta_{k+1}, \phi_{j,k+1}\big) &= z_{k+1}^{\mathrm{T}} \overline{H}_{j,k+1} z_{k+1} \end{aligned} \tag{45}$$

where $z_{k+1} = [\zeta_{k+1}^{\mathrm{T}} \ \ \overline{\iota}_{j,k+1}\zeta_{k+1}]^{\mathrm{T}}$, and $\overline{H}_{j+1}$ is the approximation of H_k at iteration the $j+1$. Substituting system dynamics (17) and (45) into (31), we have

$$\begin{aligned} \overline{H}_{j+1,k} &= \begin{bmatrix} S & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} A_k & B_k \\ \overline{\iota}_{j,k+1} A_k & \overline{\iota}_{j,k+1} B_k \end{bmatrix}^{\mathrm{T}} \\ &\quad \times \overline{H}_{j,k+1} \begin{bmatrix} A_k & B_k \\ \overline{\iota}_{j,k+1} A_k & \overline{\iota}_{j,k+1} B_k \end{bmatrix} \\ &= \begin{bmatrix} S & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} A_k & B_k \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} I & \overline{\iota}_{j,k+1}^{\mathrm{T}} \end{bmatrix} \\ &\quad \times \overline{H}_{j,k+1} \begin{bmatrix} I & \overline{\iota}_{j,k+1}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} A_k & B_k \end{bmatrix}. \end{aligned} \tag{46}$$

From (44), we have

$$\overline{P}_{j,k+1} = \begin{bmatrix} I & \bar{\iota}_{j,k+1}^{\mathrm{T}} \end{bmatrix} \overline{H}_{j,k+1} \begin{bmatrix} I & \bar{\iota}_{j,k+1}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}. \tag{47}$$

Then, substituting (47) into (46), we have

$$\begin{aligned} \overline{H}_{j+1,k} &= \begin{bmatrix} S & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} A_k & B_k \end{bmatrix}^{\mathrm{T}} \overline{P}_{j,k+1} \\ &\quad \times \begin{bmatrix} A_k & B_k \end{bmatrix} \\ &= \begin{bmatrix} \overline{H}_{\zeta\zeta,j+1} & \overline{H}_{\zeta u,j+1} \\ \overline{H}_{u\zeta,j+1} & \overline{H}_{uu,j+1} \end{bmatrix} \end{aligned} \tag{48}$$

where

$$\begin{aligned} \overline{H}_{\zeta\zeta,j+1} &= A_k^{\mathrm{T}} \overline{P}_{j,k+1} A_k + S \\ \overline{H}_{\zeta u,j+1} &= \overline{H}_{u\zeta,j+1}^{\mathrm{T}} = A_k^{\mathrm{T}} \overline{P}_{j,k+1} B_k \\ \overline{H}_{uu,j+1} &= B_k^{\mathrm{T}} \overline{P}_{j,k+1} B_k + R. \end{aligned} \tag{49}$$

According to (47), we also have

$$\overline{P}_{j+1,k} = \begin{bmatrix} I & \bar{\iota}_{j+1,k}{}^{\mathrm{T}} \end{bmatrix} \overline{H}_{j+1,k} \begin{bmatrix} I & \bar{\iota}_{j+1,k}{}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}. \tag{50}$$

Substituting (48) into (50) and using (35), we have

$$\bar{\iota}_{j+1,k} = -\left[R + B_k^{\mathrm{T}} \overline{P}_{j,k+1} B_k\right]^{-1} \times B_k^{\mathrm{T}} \overline{P}_{j,k+1} A_k \tag{51}$$

and

$$\begin{aligned} \overline{P}_{j+1,k} = A_k^{\mathrm{T}} \overline{P}_{j,k+1} A_k + S - A_k^{\mathrm{T}} \overline{P}_{j,k+1} B_k \left[R + B_k^{\mathrm{T}} \overline{P}_{j,k+1} B_k\right]^{-1} \\ \times B_k^{\mathrm{T}} \overline{P}_{j,k+1} A_k. \end{aligned} \tag{52}$$

From above analysis, we can conclude that $\overline{P}_{j+1,k} \to P_k$, and $\iota_{j+1,k} \to \iota_k^*$ as $j \to \infty$. Therefore, $Q(\zeta_k, u_k)$ will converge to the optimal value $Q^*(\zeta_k, u_k)$ as $j \to \infty$.

C. Optimal Admittance Adaptation

In this section, the admittance adaptation method will be presented. First, a cost function is defined

$$J(k) = \sum_{k=1}^{\infty} \left[(x_k - x_d)^{\mathrm{T}} S (x_k - x_d) + f_e^{\mathrm{T}} R f_e\right]. \tag{53}$$

In (53), the optimal problem is composed of trajectory tracking and force regulation. To make this identical in robot control system, some transformation should be considered. The desired trajectory is defined as

$$x_d(k+1) = \mathscr{G} x_d(k) \tag{54}$$

where $\mathscr{G}$ is the diagonal constant matrix. It is pointed out that various desired trajectories can be generated by (54) with $\mathscr{G}$ not Hurwitz [24].

Consider the environmental dynamics in (4) and (54), and the augmented system can be written as

$$\chi_{k+1} = \overline{A}_k \chi_k + \overline{B}_k f_e \tag{55}$$

where the augment matrices $\overline{A}_k$ and $\overline{B}_k$ are defined as

$$\begin{aligned} \overline{A}_k &= \begin{bmatrix} A_e & 0 \\ 0 & \mathscr{G} \end{bmatrix}, \quad \overline{B}_k = \begin{bmatrix} B_e \\ 0 \end{bmatrix} \\ \overline{S} &= \begin{bmatrix} S & -S \\ -S & S \end{bmatrix} \\ \overline{R} &= R. \end{aligned} \tag{56}$$

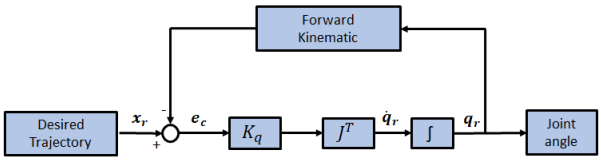

Fig. 3. Diagram of CLIK (modified from [53]).

Then, the cost function is rewritten as

$$J(k) = \sum_{k=1}^{\infty} \left[\chi_k^{\mathrm{T}} \overline{S} \chi_k + f_{ek}^{\mathrm{T}} \overline{R} f_{ek}\right]. \tag{57}$$

Now, the augment system (55) has the same form as (17), and we can use the RL method in Section III-B to obtain the optimal control input as

$$f_e(k) = \bar{\iota} \chi_k \tag{58}$$

where $\bar{\iota}$ is the optimal gain equivalent to LQR gain in (25). By comparing (58) with (5), we can regard (58) as the desired admittance model, from which the optimal adaptation x_r for the external force will be generated.

D. NN Control Design

In this section, an NN-based adaptive controller with dynamic learning framework will be introduced. As shown in Fig. 1, the designed controller is to ensure that the actual trajectory of the robot can track the modified desired trajectory q_r subject to the uncertainties of the system.

Before controller design, the closed-loop inverse kinematics (CLIK) will be introduced [52], [53]. As shown in Fig. 1, after generation of the modified desired trajectory x_r in the Cartesian space, we map it into joint space as the input signal of the tracking loop. In (2), $\dot{q}$ can be obtained by

$$\dot{q} = J^{+}(q) \dot{x} \tag{59}$$

where J^{+} denotes the pseudoinverse of J and can be given by $J^{+} = J^{\mathrm{T}}(J J^{\mathrm{T}})^{-1}$. However, this inverse method will bring a lot of numerical drifts and computational burden. To deal with the problem, the CLIK method is employed, as shown in Fig. 3. By using the CLIK, (59) can be written as

$$\dot{q} = K_q J^{\mathrm{T}}(q) e_c \tag{60}$$

where K_q is the control gain and e_c is the error. In Fig. 3, we can see that the CLIK method only uses the forward kinematics instead of computing the inverse of Jacobian matrix, which can effectively decrease the computational burden.

After the modified desired trajectory is mapped into joint space, we define the tracking errors as

$$\begin{aligned} e_p &= q_r - q \\ \alpha &= \dot{q}_r + K_p e_p \\ e_v &= \dot{e}_p + K_p e_p \end{aligned} \tag{61}$$

where K_p is the gain matrix and e_q and e_v denote the position and velocity tracking error, respectively.

We design the control torque as

$$\tau_c = \hat{D}_q\dot{\alpha} + \hat{C}_q\alpha + \hat{G}_q + K_v e_v - \tau_e \tag{62}$$

where $\hat{D}_q, \hat{C}_q$, and $\hat{G}_q$ are the estimates of D_q, C_q, and G_q, respectively; $\tau_e = J^{\mathrm{T}} f_e$; and K_v is the diagonal gain matrix with positive constant.

The updating law for NN weight is

$$\begin{aligned}\dot{\hat{W}}_D &= \Gamma_D^{-1}\big(S(Z_D)\dot{\alpha}e_v - \varrho_D\hat{W}_D\big)\\ \dot{\hat{W}}_C &= \Gamma_C^{-1}\big(S(Z_C)\alpha e_v - \varrho_C\hat{W}_C\big)\\ \dot{\hat{W}}_G &= \Gamma_G^{-1}\big(S(Z_G)e_v - \varrho_G\hat{W}_G\big)\end{aligned} \tag{63}$$

where $\Gamma_{(\cdot)}^{-1}$ is the learning rate with positive matrix and $\varrho_{(\cdot)}$ is the matrix with small positive constant for disturbance [42].

Substituting (62) into (3) yields

$$\begin{aligned}-D_q\dot{e}_v = K_v e_v + \big(\hat{D}_q - D_q\big)\dot{\alpha} + \big(\hat{C}_q - C_q\big)\alpha\\ + C_q e_v + \big(\hat{G}_q - G_q\big).\end{aligned} \tag{64}$$

E. Stability Analysis

In this section, the analysis of system stability will be presented. We construct the Lyapunov function as

$$\begin{aligned}V = \frac{1}{2}e_v^{\mathrm{T}}D_q e_v + \frac{1}{2}\mathrm{tr}\big(\tilde{W}_D^{\mathrm{T}}\Gamma_D\tilde{W}_D\big) + \frac{1}{2}\mathrm{tr}\big(\tilde{W}_C^{\mathrm{T}}\Gamma_C\tilde{W}_C\big)\\ + \frac{1}{2}\mathrm{tr}\big(\tilde{W}_G^{\mathrm{T}}\Gamma_G\tilde{W}_G\big).\end{aligned} \tag{65}$$

Taking derivative of (65) and yields

$$\begin{aligned}\dot{V} = e_v^{\mathrm{T}}D_q\dot{e}_v + \frac{1}{2}e_v^{\mathrm{T}}\dot{D}_q e_v + \mathrm{tr}\Big(\tilde{W}_D^{\mathrm{T}}\Gamma_D\dot{\hat{W}}_D\Big)\\ + \mathrm{tr}\Big(\tilde{W}_C^{\mathrm{T}}\Gamma_C\dot{\hat{W}}_C\Big) + \mathrm{tr}\Big(\tilde{W}_G^{\mathrm{T}}\Gamma_G\dot{\hat{W}}_G\Big).\end{aligned} \tag{66}$$

Substituting (64) into (66), we have

$$\begin{aligned}\dot{V} = e_v^{\mathrm{T}}\big(-\tilde{D}_q\dot{\alpha} - \tilde{C}_q\alpha - \tilde{G}_q\big)\\ - e_v^{\mathrm{T}}K_v e_v + \mathrm{tr}\Big(\tilde{W}_D^{\mathrm{T}}\Gamma_D\dot{\hat{W}}_D\Big)\\ + \mathrm{tr}\Big(\tilde{W}_C^{\mathrm{T}}\Gamma_C\dot{\hat{W}}_C\Big) + \mathrm{tr}\Big(\tilde{W}_G^{\mathrm{T}}\Gamma_G\dot{\hat{W}}_G\Big).\end{aligned} \tag{67}$$

Substituting the NN adaptive law into (67) yields

$$\begin{aligned}\dot{V} &= -e_v^{\mathrm{T}}K_v e_v - e_v^{\mathrm{T}}\tilde{D}_q\dot{\alpha} - e_v^{\mathrm{T}}\tilde{C}_q\alpha - e_v^{\mathrm{T}}\tilde{G}_q\\ &\quad + \mathrm{tr}\Big(\tilde{W}_D^{\mathrm{T}}\Gamma_D\dot{\hat{W}}_D\Big) + \mathrm{tr}\Big(\tilde{W}_C^{\mathrm{T}}\Gamma_C\dot{\hat{W}}_C\Big) + \mathrm{tr}\Big(\tilde{W}_G^{\mathrm{T}}\Gamma_G\dot{\hat{W}}_G\Big)\\ &= -e_v^{\mathrm{T}}K_v e_v - e_v^{\mathrm{T}}\tilde{D}_q\dot{\alpha} - e_v^{\mathrm{T}}\tilde{C}_q\alpha - e_v^{\mathrm{T}}\tilde{G}_q\\ &\quad + \mathrm{tr}\big[\tilde{W}_D^{\mathrm{T}}\big(S(Z_D)\dot{\alpha}e_v - \varrho_D\hat{W}_D\big)\big]\\ &\quad + \mathrm{tr}\big[\tilde{W}_C^{\mathrm{T}}\big(S(Z_C)\alpha e_v - \varrho_C\hat{W}_C\big)\big]\\ &\quad + \mathrm{tr}\big[\tilde{W}_G^{\mathrm{T}}\big(S(Z_G)e_v - \varrho_G\hat{W}_G\big)\big]\\ &= -e_v^{\mathrm{T}}K_v e_v - \varrho_D\mathrm{tr}\big(\tilde{W}_D^{\mathrm{T}}\tilde{W}_D\big) - \varrho_D\mathrm{tr}\big(\tilde{W}_D^{\mathrm{T}}W_D\big)\\ &\quad - \varrho_C\mathrm{tr}\big(\tilde{W}_C^{\mathrm{T}}\tilde{W}_C\big) - \varrho_C\mathrm{tr}\big(\tilde{W}_C^{\mathrm{T}}W_C\big) - \varrho_G\mathrm{tr}\big(\tilde{W}_G^{\mathrm{T}}\tilde{W}_G\big)\\ &\quad - \varrho_G\mathrm{tr}\big(\tilde{W}_G^{\mathrm{T}}W_G\big).\end{aligned} \tag{68}$$

By applying Young's inequality, the derivative of V becomes

$$\begin{aligned}\dot{V} \le -e_v^{\mathrm{T}}K_v e_v + \frac{1}{2}e_v^{\mathrm{T}}e_v - \frac{1}{2}\varrho_D\mathrm{tr}\big(\tilde{W}_D^{\mathrm{T}}\tilde{W}_D\big)\\ - \frac{1}{2}\varrho_C\mathrm{tr}\big(\tilde{W}_C^{\mathrm{T}}\tilde{W}_C\big) - \frac{1}{2}\varrho_G\mathrm{tr}\big(\tilde{W}_G^{\mathrm{T}}\tilde{W}_G\big) + \Delta\end{aligned} \tag{69}$$

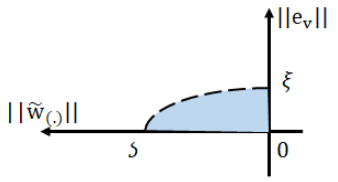

Fig. 4. Illustration of UUB defined in (72).

where $\Delta = (1/2)\varrho_D\mathrm{tr}(W_D^{*\mathrm{T}}W_D^*) + (1/2)\varrho_C\mathrm{tr}(W_C^{*\mathrm{T}}W_C^*) + (1/2)\varrho_G\mathrm{tr}(W_G^{*\mathrm{T}}W_G^*)$.

Then, if variables in (69) satisfy the following inequality:

$$\begin{aligned}\Delta \le e_v^{\mathrm{T}}K_v e_v - \frac{1}{2}e_v^{\mathrm{T}}e_v + \frac{1}{2}\varrho_D\mathrm{tr}\big(\tilde{W}_D^{\mathrm{T}}\tilde{W}_D\big)\\ + \frac{1}{2}\varrho_C\mathrm{tr}\big(\tilde{W}_C^{\mathrm{T}}\tilde{W}_C\big) + \frac{1}{2}\varrho_G\mathrm{tr}\big(\tilde{W}_G^{\mathrm{T}}\tilde{W}_G\big)\end{aligned} \tag{70}$$

we have $\dot{V} \le 0$.

Theorem 1: Consider the robotic system dynamics (3) and control input (62) together with adaptive NN law (63). The closed-loop signals can guarantee the semiglobal uniformly boundedness with a bounded desired trajectory x_d and $\dot{x}_d$. The closed-loop signals will converge to the invariant set as follows:

$$\begin{aligned}\Omega_s = \Bigg\{\big(||\tilde{W}_{(D)}||, ||\tilde{W}_C||, ||\tilde{W}_G||, ||e_v||\big), |\\ \frac{e_v^{\mathrm{T}}\big(K_v - \frac{1}{2}I\big)e_v}{\Delta} + \frac{\varrho_D||\tilde{W}_D||^2}{\Delta} + \frac{\varrho_\Delta||\tilde{W}_C||^2}{\Delta}\\ + \frac{\varrho_G||\tilde{W}_G||^2}{\Delta} \le 1\Bigg\}.\end{aligned} \tag{71}$$

As shown in Fig. 4, Ω_s will pass through the following points in the first quadrant:

$$\begin{aligned}\varrho_{(\cdot)}||\tilde{W}_{(\cdot)}||^2 = \Delta, \quad \tilde{W} = \varsigma\\ e_v^{\mathrm{T}}\Big(K_v - \frac{1}{2}I\Big)e_v = \Delta, \quad e_v = \xi.\end{aligned} \tag{72}$$

Remark 3: In [42], it has pointed out that the input orbit could not explore all neural nodes. Therefore, NN learning can only appear in the neural nodes near the input trajectory. According to the NN adaptive law defined in (63), responses to activation functions of those NN nodes with centers far from input are very small, and their weights will update slightly, i.e., little learning experience can be obtained. Considering the learning ability of NNs, we can discard those nodes that have little activation response to make the learning more effective, and we define a constraint for neural nodes and input trajectories as

$$d(q_{ri}, \theta_i) < \mathrm{dis} \Rightarrow |\hat{W}^{\mathrm{T}}S(Z_n) - F(Z_n)| < \epsilon \tag{73}$$

where $F(\cdot)$ is the unknown function and $\varepsilon(\cdot)$ is the approximation error and $d(\cdot)$ is the function of the distance between the input trajectory and centers of NN nodes. In this case, we can rewrite the approximation law as

$$\begin{aligned}F(Z_n) &= W^{*\mathrm{T}}S(Z_n) + \varepsilon(Z_n)\\ &= \hat{W}_c^{\mathrm{T}}S_c(Z_n)\end{aligned} \tag{74}$$

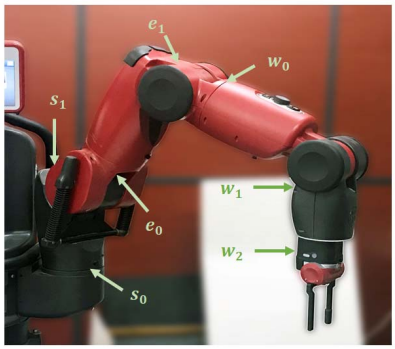

Fig. 5. Configuration of the Baxter robot arm.

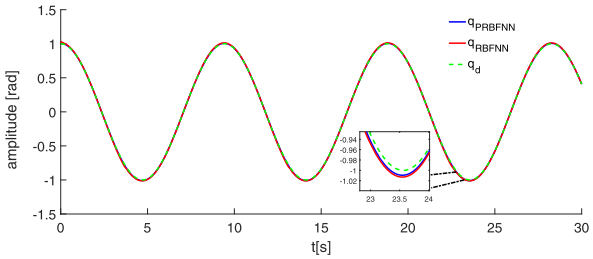

Fig. 6. Comparison of tracking performance between PRBFNN learning controller and conventional RBFNN learning controller [55]–[57].

where $\hat{W}_c$ denotes subvector of W and $S_c(Z_n)$ will satisfy the following equality:

$$S(Z_n) > \pi \tag{75}$$

where π denotes the constraint value, and it is a positive constant set by the designer.

IV. Experiment

In this section, experiments on the Baxter robot are conducted. The robotic arm with seven-DOF is employed to be interacting with the environment. The configuration of the robot is shown in Fig. 5, and model dynamics are given in [54]. In experiments, we verify the effectiveness of the proposed control method by testing the joint s_1, and other joints are fixed. The time-varying matrices in (4) are defined as

$$A_e(k) = 0.99 - \frac{0.01}{0.15[\sin(0.05k) + 1.1]}$$
$$B_e(k) = -\frac{0.01}{0.15[\sin(0.05k) + 1.1]}. \tag{76}$$

The interaction force is applied to the manipulator in one direction, and the manipulator will be in free notion in other directions. The initial position is set as $x(0) = [0; 0; 0; 0; 0; 0; 0]^{\mathrm{T}}$, and the gain matrix of the desired trajectory in (54) is $\mathcal{G} = 0.98$.

A. Test of RBFNN Adaptive Controller

In this group of experiments, the tracing performance of the proposed adaptive NN controller will be given, and results compared with conventional RBFNN learning controller [55]–[57] will be illustrated. The desired trajectory is set as $q_d = \sin(t/10)$. The RBFNN is employed to approximate nonlinear terms: $F(q, \dot{q}, \alpha, \dot{\alpha}) = D_q\dot{\alpha} + C_q\alpha + G_q$. The

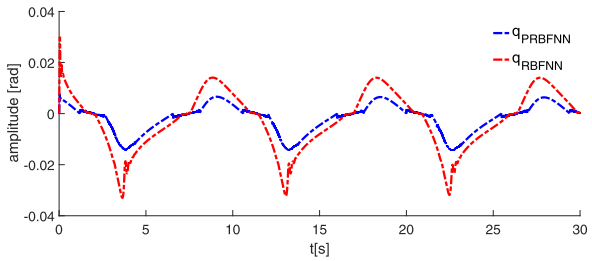

Fig. 7. Comparison of tracking errors.

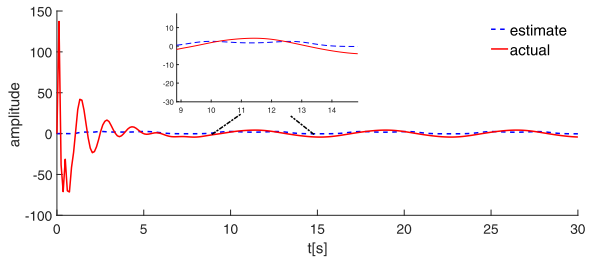

Fig. 8. Estimation performance under conventional RBFNN learning framework [55]–[57].

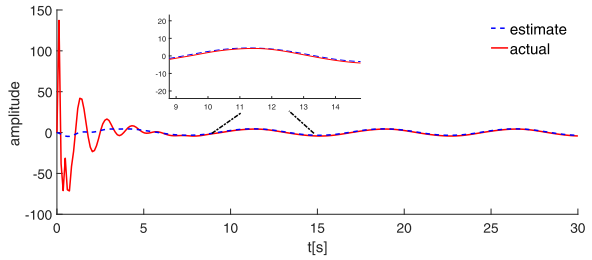

Fig. 9. Estimation performance under the PRBFNN learning framework.

control gain is set as: $K_p = \mathrm{diag}[40; 30; 25; 30; 20; 6; 7]^{\mathrm{T}}$, $K_v = \mathrm{diag}[2;3;2;2; 1.8; 1.5; 0.8]^{\mathrm{T}}$. The learning rate of RBFNN is $\Gamma_{(\cdot)} = 0.01$ and $\varrho_{(\cdot)} = 0.015$. The initial value of NN weight is $\hat{W}_0 = \mathbf{0}$. The initial number of NN nodes is selected as 256. To demonstrate the effectiveness of the proposed learning controller and make a fair comparison, we compare the proposed controller with other three related works [55]–[57], whose control gains are selected identically as the mentioned above. The results are shown in Figs. 6–12. In Fig. 6, the actual trajectory of the controller [55]–[57] is marked in red line and the trajectory of the proposed controller is marked in blue line, which are managed to follow the desired trajectory q_d marked in green line. From Figs. 6 and 7, the proposed RBFNN (PRBFNN) learning controller has a better tracking performance and the tracking error is within 0.016 rad, which is smaller than the error within 0.037 rad of conventional RBFNN learning controller in [55]–[57]. The reason is that the PRBFNN learning method has a more precise approximation of the dynamic uncertainties $F(\cdot)$ as a feedforward compensation signal in controller design. Figs. 11 and 12 show the convergence trend of the RBFNN weight without the proposed learning framework and the incremental weight with the proposed learning framework. From Fig. 11, we can see that those NN weights update slightly and close to zero with their centers far away from the input trajectory. Those NN nodes with their center close to the input trajectory are activated well and contribute more

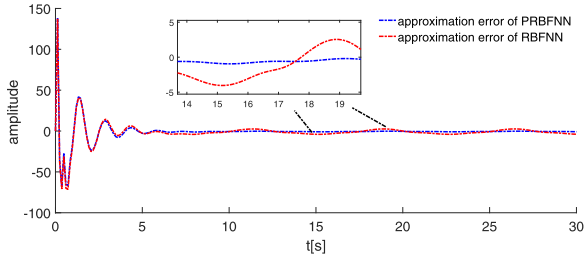

Fig. 10. Comparison of approximation errors.

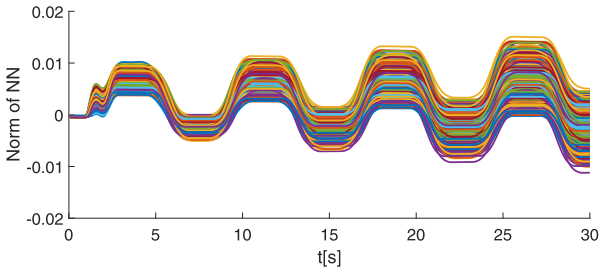

Fig. 11. Convergence of NN weight under conventional RBFNN learning framework [55]–[57].

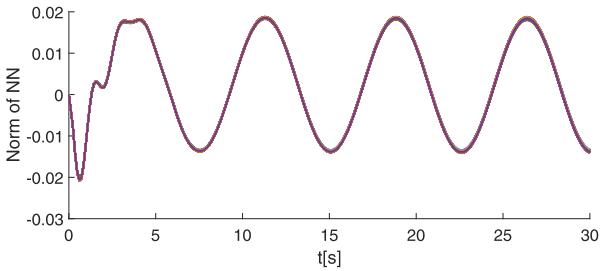

Fig. 12. Convergence of W_a under the PRBFNN learning framework.

to the approximation of nonlinear terms. According to the adaptive NN law in (63) and the constraint condition in (75), we discard the NN nodes that have little response to the activation function with $\pi = 0.1$. Then, 123 NN nodes are discarded, and 30 new nodes with the information defined in (13) are placed. Thus, the number of NN nodes under the proposed learning work is 163, which is smaller than the initial number 256. As shown in Fig. 12, the incremental nodes are activated well and their weights are bounded. The function approximation performance with and without the proposed learning framework is shown in Figs. 8 and 9, respectively. Compared with the approximation performance of conventional RBFNN controller in Fig. 8, in Fig. 9, RBFNN with the proposed learning framework shows a better approximation performance of system uncertainties after the convergence of tracking errors. According to the result of the approximation errors in Fig. 10, we can see that, with the conventional RBFNN learning, the average approximation error is more than 1.7 and the largest approximation error is about 3.8 around 14.8 s. Comparatively, under the PRBFNN learning framework, the average approximation error is about -0.5, and the largest approximation error is about -0.8 around 14.8 s, which outperforms the conventional RBFNN learning framework.

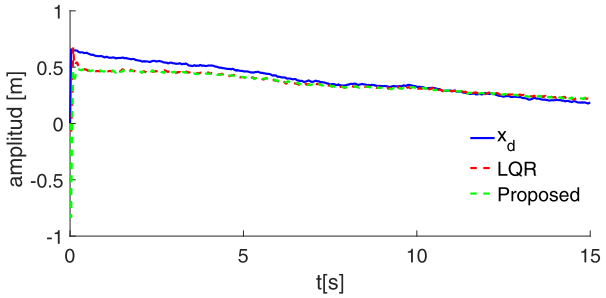

Fig. 13. Desired trajectory, modified desired trajectory with the proposed and LQR method in Cartesian space, $S = 1$ and $R = 1$.

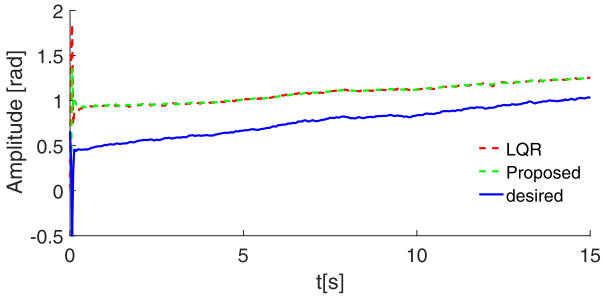

Fig. 14. Desired trajectory, modified desired trajectory with the proposed and LQR method in joint space, $S = 1$ and $R = 1$.

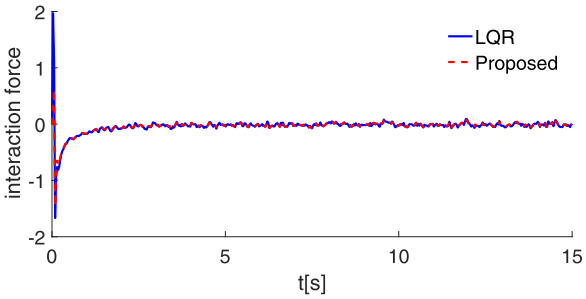

Fig. 15. Interaction force with the proposed and LQR method, $S = 1$ and $R = 1$.

B. Test of the Proposed Method Compared With LQR

In this group of experiments, the performance of the proposed admittance adaptation method is compared with the performance of LQR method. In the LQR method, the optimal control (25) (optimal admittance model) is obtained by solving the DARE equation (26). However, the drawback of LQR method is that information of environmental dynamics is required, which limits its application in many control systems. In contrast to LQR designs, the proposed method can realize the optimal control, ultimately without knowing the environment dynamics. The weight matrix in loss function is set as $S = 1$ and $R = 1$. The results are shown in Figs. 13–16. When the manipulator receives the external force from the environment, the desired trajectory x_d will be modified to adapt to the environmental force. In Figs. 13 and 14, the modified trajectory x_r (green dotted line) generated from the admittance model (58) by our proposed method can track the one (red dotted line) obtained by the LQR method, and the corresponding trajectory q_r in joint space is generated by using the CLIK method. In Fig. 15, the interaction force is decreasing to a neighborhood of zero with the admittance

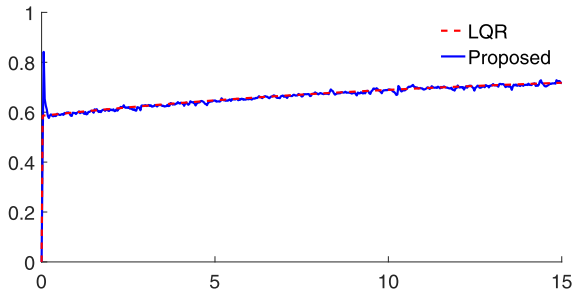

Fig. 16. Gain parameters with the proposed and LQR method, $S = 1$ and $R = 1$.

adaptation to the environment, and the force trajectory (red dotted line) of our proposed method follows the one (blue solid line) of the LQR method. The norm of the feedback gain obtained from the proposed (58) and the LQR method (25) is presented in Fig. 16, which shows that the value of our proposed method can converge to that of LQR method under the time-varying dynamics of the environment.

V. Conclusion

In this paper, we have proposed an NN enhanced admittance adaptation method for optimal robot–environment interaction control. The RL method is employed to solve the DARE equation without knowing the time-varying environmental dynamics and to make the robot have an optimal compliant motion to adapt to environmental forces. To guarantee the tracking performance, an adaptive RBFNN controller with dynamic learning framework is proposed for better estimating nonlinear terms of robot dynamics to ensure that the robotic arm can effectively follow the modified desired trajectory. The experiments are conducted and the results illustrate the feasibility of our proposed method.

References

[1] H. Qiao, M. Wang, J. Su, S. Jia, and R. Li, "The concept of 'attractive region in environment' and its application in high-precision tasks with low-precision systems," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 5, pp. 2311–2327, Oct. 2015.

[2] J. J. Craig and M. H. Raibert, "A systematic method of hybrid position/force control of a manipulator," in *Proc. Comput. Softw. IEEE Comput. Soc. 3rd Int. Appl. Conf. (COMPSAC)*, Nov. 1979, pp. 446–451.

[3] N. Hogan, "Impedance control—An approach to manipulation. I—Theory. II—Implementation. III—Applications," *Trans. ASME J. Dyn. Syst., Meas. Control*, vol. 107, no. 1, pp. 1–24, 1985.

[4] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, no. 6, pp. 418–432, Jun. 1981.

[5] I. Ranatunga, F. L. Lewis, D. O. Popa, and S. M. Tousif, "Adaptive admittance control for human–robot interaction using model reference design and adaptive inverse filtering," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 1, pp. 278–285, Jan. 2017.

[6] Y. Li, S. S. Ge, and C. Yang, "Learning impedance control for physical robot–environment interaction," *Int. J. Control*, vol. 85, no. 2, pp. 182–193, Feb. 2012.

[7] Y. Li, K. P. Tee, R. Yan, W. L. Chan, and Y. Wu, "A framework of human–robot coordination based on game theory and policy iteration," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1408–1418, Dec. 2016.

[8] Y. Li and S. S. Ge, "Impedance learning for robots interacting with unknown environments," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 4, pp. 1422–1432, Jul. 2014.

[9] C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schaeffer, and E. Burdet, "Human-like adaptation of force and impedance in stable and unstable interactions," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 918–930, Oct. 2011.

[10] R. Johansson and M. W. Spong, "Quadratic optimization of impedance control," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 1994, pp. 616–621.

[11] M. Matinfar and K. Hashtrudi-Zaad, "Optimization-based robot compliance control: Geometric and linear quadratic approaches," *Int. J. Robot. Res.*, vol. 24, no. 8, pp. 645–656, Aug. 2005.

[12] P. J. Werbos, "A menu of designs for reinforcement learning over time," in *Neural Networks for Control*. Cambridge, MA, USA: MIT Press, 1990, pp. 67–95.

[13] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 2, 3rd ed. Belmont, MA, USA: Athena Scientific, 2011.

[14] 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.

[15] B. Fan, Q. Yang, X. Tang, and Y. Sun, "Robust ADP design for continuous-time nonlinear systems with output constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2127–2138, Jun. 2018.

[16] G. G. Lendaris, "Adaptive dynamic programming approach to experience-based systems identification and control," *Neural Netw.*, vol. 22, nos. 5–6, pp. 822–832, Jul. 2009.

[17] P. J. Werbos, "Foreword–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.

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

[19] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Syst.*, vol. 32, no. 6, pp. 76–105, Dec. 2012.

[20] B. Kim, J. Park, S. Park, and S. Kang, "Impedance learning for robotic contact tasks using natural actor-critic algorithm," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 40, no. 2, pp. 433–443, Apr. 2010.

[21] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 820–833, Jun. 2011.

[22] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of dynamic systems by learning: A new control theory for servomechanism or mechatronics systems," in *Proc. 23rd IEEE Conf. Decis. Control*, Dec. 1984, pp. 1064–1069.

[23] T. Tsuji, K. Ito, and P. G. Morasso, "Neural network learning of robot arm impedance in operational space," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 26, no. 2, pp. 290–298, Apr. 1996.

[24] S. S. Ge, Y. Li, and C. Wang, "Impedance adaptation for optimal robot–environment interaction," *Int. J. Control*, vol. 87, no. 2, pp. 249–263, Feb. 2014.

[25] Y. Shmaliy, "Linear time-varying systems," in *Continuous-Time Systems*. Dordrecht, The Netherlands: Springer, 2007, pp. 349–423.

[26] Y.-J. Liu, Q. Zeng, S. Tong, C. P. Chen, and L. Liu, "Adaptive neural network control for active suspension systems with time-varying vertical displacement and speed constraints," *IEEE Trans. Ind. Electron.*, vol. 66, no. 12, pp. 9458–9466, 2019.

[27] W. He, T. Wang, X. He, L.-J. Yang, and O. Kaynak, "Dynamical modeling and boundary vibration control of a rigid-flexible wing system," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 6, pp. 2711–2721, Dec. 2020.

[28] B. Fan, Q. Yang, S. Jagannathan, and Y. Sun, "Asymptotic tracking controller design for nonlinear systems with guaranteed performance," *IEEE Trans. Cybern.*, vol. 48, no. 7, pp. 2001–2011, Jul. 2018.

[29] W. He, X. Mu, L. Zhang, and Y. Zou, "Modeling and trajectory tracking control for flapping-wing micro aerial vehicles," *IEEE/CAA J. Autom. Sinica*, vol. 24, no. 8, pp. 148–156, Sep. 2020.

[30] Z. Liu, G. Lai, Y. Zhang, and C. P. Chen, "Adaptive fuzzy tracking control of nonlinear time-delay systems with dead-zone output mechanism based on a novel smooth model," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 6, pp. 1998–2011, 2015.

[31] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, "Robot learning system based on adaptive neural control and dynamic movement primitives," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 777–787, Mar. 2019.

[32] S. Sui, C. P. Chen, and S. Tong, "Neural network filtering control design for nontriangular structure switched nonlinear systems in finite time," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 7, pp. 2153–2162, 2018.

[33] G. Peng, C. Yang, W. He, and C. L. P. Chen, "Force sensorless admittance control with neural learning for robots with actuator saturation," *IEEE Trans. Ind. Electron.*, vol. 67, no. 4, pp. 3138–3148, Apr. 2020.
[34] D. Li, C. P. Chen, Y.-J. Liu, and S. Tong, "Neural network controller design for a class of nonlinear delayed systems with time-varying full-state constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2625–2636, 2019.
[35] W. He, T. Meng, X. He, and C. Sun, "Iterative learning control for a flapping wing micro aerial vehicle under distributed disturbances," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1524–1535, Apr. 2019.
[36] C. Yang, Y. Jiang, Z. Li, W. He, and C.-Y. Su, "Neural control of bimanual robots with guaranteed global stability and motion precision," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1162–1171, Jun. 2017.
[37] W. He and Y. Dong, "Adaptive fuzzy neural network control for a constrained robot using impedance learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 1174–1186, Apr. 2018.
[38] Z. Liu, C. Chen, Y. Zhang, and C. L. P. Chen, "Adaptive neural control for dual-arm coordination of humanoid robot with unknown nonlinearities in output mechanism," *IEEE Trans. Cybern.*, vol. 45, no. 3, pp. 507–518, Mar. 2015.
[39] C. L. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.
[40] C. L. P. Chen, Z. Liu, and S. Feng, "Universal approximation capability of broad learning system and its structural variations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1191–1204, Apr. 2019.
[41] S. Sui, C. L. P. Chen, S. Tong, and S. Feng, "Finite-time adaptive quantized control of stochastic nonlinear systems with input quantization: A broad learning system based identification method," *IEEE Trans. Ind. Electron.*, vol. 67, no. 10, pp. 8555–8565, Oct. 2020.
[42] C. Wang and D. J. Hill, "Learning from neural control," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 130–146, Jan. 2006.
[43] C. Wang, T. Chen, G. Chen, and D. J. Hill, "Deterministic learning of nonlinear dynamical systems," *Int. J. Bifurcation Chaos*, vol. 19, no. 4, pp. 1307–1328, Apr. 2009.
[44] H. Huang, T. Zhang, C. Yang, and C. L. P. Chen, "Motor learning and generalization using broad learning adaptive neural control," *IEEE Trans. Ind. Electron.*, vol. 67, no. 10, pp. 8608–8617, Oct. 2020.
[45] G. Gilardi and I. Sharf, "Literature survey of contact dynamics modelling," *Mechanism Mach. Theory*, vol. 37, no. 10, pp. 1213–1239, Oct. 2002.
[46] T. H. Lee and C. J. Harris, *Adaptive Neural Network Control of Robotic Manipulators*, vol. 19. Singapore: World Scientific, 1998.
[47] T. Landelius, "Reinforcement learning and distributed local model synthesis," Ph.D. dissertation, Linköping Univ. Electronic Press, Linköping, Sweden, 1997.
[48] C. Wang, Y. Li, S. S. Ge, and T. H. Lee, "Optimal critic learning for robot control in time-varying environments," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2301–2310, Oct. 2015.
[49] K. J. Åström and B. Wittenmark, *Adaptive Control*. Chelmsford, MA, USA: Courier Corporation, 2013.
[50] K. Xiong, L. Liu, and H. Zhang, "Adaptive robust extended Kalman filter for nonlinear stochastic systems," *IET Control Theory Appl.*, vol. 2, no. 3, pp. 239–250, Mar. 2008.
[51] S. S. Ge, "Adaptive controller design for flexible joint manipulators," *Automatica*, vol. 32, no. 2, pp. 273–278, Feb. 1996.
[52] J.-J. Slotine and D. Yoerger, "A rule-based inverse kinematics algorithm for redundant manipulators," *Int. J. Robot. Autom.*, vol. 2, no. 2, pp. 86–89, 1987.
[53] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE J. Robot. Autom.*, vol. 4, no. 4, pp. 403–410, Aug. 1988.
[54] C. Yang, H. Ma, and M. Fu, *Advanced Technologies in Modern Robotic Applications*. Singapore: Springer, 2016.
[55] W. He, Z. Li, Y. Dong, and T. Zhao, "Design and adaptive control for an upper limb robotic exoskeleton in presence of input saturation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 1, pp. 97–108, Jan. 2019.
[56] C. Yang, X. Wang, L. Cheng, and H. Ma, "Neural-learning-based telerobot control with guaranteed performance," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3148–3159, Oct. 2017.
[57] W. He, Y. Dong, and C. Sun, "Adaptive neural impedance control of a robotic manipulator with input saturation," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 46, no. 3, pp. 334–344, Mar. 2016.

Guangzhu Peng received the B.Eng. degree in automation from Yangtze University, Jingzhou, China, in 2014, and the M.Eng. degree in pattern recognition and intelligent systems from the School of Automation Science and Engineering, South China University of Technology (SCUT), Guangzhou, China, in 2018. He is currently pursuing the Ph.D. degree in computer science with the Faculty of Science and Technology, University of Macau, Macau, China.

His current research interests include robotics, human–robot interaction, and intelligent control.

C. L. Philip Chen (Fellow, IEEE) received the M.S. degree in electrical and computer science from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA, in 1985, and the Ph.D. degree in electrical and computer science from Purdue University, West Lafayette, IN, USA, in 1988.

He is currently the Chair Professor and the Dean of the College of Computer Science and Engineering, South China University of Technology, Guangzhou, China. Being a Program Evaluator of the Accreditation Board of Engineering and Technology Education (ABET) in USA, for computer engineering, electrical engineering, and software engineering programs, he successfully architects the University of Macau's Engineering and Computer Science programs receiving accreditations from Washington/Seoul Accord through the Hong Kong Institute of Engineers (HKIE), of which is considered as his utmost contribution in engineering/computer science education for Macau as the former Dean of the Faculty of Science and Technology. His current research interests include cybernetics, systems, and computational intelligence.

Dr. Chen is a fellow of AAAS, IAPR, CAA, and HKIE; and a member of the Academia Europaea (AE), the European Academy of Sciences and Arts (EASA), and the International Academy of Systems and Cybernetics Science (IASCYS). He received the IEEE Norbert Wiener Award in 2018 for his contribution in systems and cybernetics, and machine learning. He is also a highly cited researcher by Clarivate Analytics in 2018, 2019, and 2020. He was a recipient of the 2016 Outstanding Electrical and Computer Engineers Award from his alma mater, Purdue University, in 1988, after he graduated from the University of Michigan at Ann Arbor in 1985. He was the Chair of the TC 9.1 Economic and Business Systems of the International Federation of Automatic Control from 2015 to 2017. He was the IEEE Systems, Man, and Cybernetics Society President from 2012 to 2013 and the Editor-in-Chief of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS from 2014 to 2019. He is also the Editor-in-Chief of the IEEE TRANSACTIONS ON CYBERNETICS and an Associate Editor of the IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE and IEEE TRANSACTIONS ON FUZZY SYSTEMS.

Chenguang Yang (Senior Member, IEEE) received the B.Eng. degree in measurement and control from Northwestern Polytechnical University, Xi'an, China, in 2005, and the Ph.D. degree in control engineering from the National University of Singapore, Singapore, in 2010. He completed his post-doctoral training in human robotics at Imperial College London, London, U.K.

His research interests include robotics and automation.

Dr. Yang was a recipient of the Best Paper Award of the IEEE TRANSACTIONS ON ROBOTICS as well as more than ten conference best paper awards.