Adaptive Q-Learning for Data-Based Optimal Output Regulation With Experience Replay

Biao Luo, *Member, IEEE*, Yin Yang, and Derong Liu, *Fellow, IEEE*

***Abstract*—In this paper, the data-based optimal output regulation problem of discrete-time systems is investigated. An off-policy adaptive Q-learning (QL) method is developed by using real system data without requiring the knowledge of system dynamics and the mathematical model of utility function. By introducing the Q-function, an off-policy adaptive QL algorithm is developed to learn the optimal Q-function. An adaptive parameter α_i in the policy evaluation is used to achieve tradeoff between the current and future Q-functions. The convergence of adaptive QL algorithm is proved and the influence of the adaptive parameter is analyzed. To realize the adaptive QL algorithm with real system data, the actor-critic neural network (NN) structure is developed. The least-squares scheme and the batch gradient descent method are developed to update the critic and actor NN weights, respectively. The experience replay technique is employed in the learning process, which leads to simple and convenient implementation of the adaptive QL method. Finally, the effectiveness of the developed adaptive QL method is verified through numerical simulations.**

***Index Terms*—Data-based, experience replay, neural networks (NNs), off-policy, optimal control, Q-learning (QL).**

I. Introduction

REINFORCEMENT learning (RL) has achieved great success in artificial intelligence. Most recently, Google deepmind group developed the ambitious AlphaGo [1], [2], which defeated the best professional Go players Lee Sedol and Ke Jie. One of the key technique in AlphaGo is the use of RL to improve its play. RL concerns the interaction between agents and environment, where the agent interacts with the environment to receive feedback in the form of rewards with the goal of learning to act so as to maximize the expected rewards. Through the iterative operations between policy evaluation and policy improvement, the long-term reward/cost can be maximized/minimized and the optimal policy can be found. In RL, the environment is typically formulated as a Markov decision process and many RL algorithms [3], [4] have been studied, to list a few, value iteration [4], policy iteration [4], temporal-difference learning [4], Q-learning (QL) [3], [5], advantage updating [6], etc. On important feature of RL is that the exact model of the environment may not be needed, which implies that RL can achieve data-based learning for the optimal policy. Most of RL algorithms utilize dynamic programming techniques, which are also the key points in developing optimal control theories. That is to say, there exist close relationships between RL and optimal control problems. It makes RL a penitential method for the design of control problems.

With the development of control theories, many intelligent control methods have been introduced, such as, neural network (NN) control [7]–[12], fuzzy control [13]–[15], advanced adaptive control [16]–[22], etc. For example, a new adaptive neural output feedback control scheme was proposed by Chen and Ge [7] for uncertain nonlinear systems. In [12], an online data-based composite neural control method was investigated for uncertain strict-feedback systems. In [14], by considering actuator saturation and uncertainty, an important adaptive sliding mode controller was designed. Yang *et al.* [16] designed a novel adaptive fault tolerant controller for a class of nonlinear unknown systems with multiple actuators. Liu and Tong [17] proposed a new adaptive controller for nonlinear systems with parameter uncertainties and full state constraints. In [21], an interesting observer-based adaptive neural fault-tolerant tracking control was studied for nonlinear systems in nonstrict-feedback form. In recent years, many adaptive dynamic programming methods [23]–[51] have been proposed to solve optimal control problems. For the optimal control problems, it is required to solve the Bellman equation for discrete-time systems and the Hamilton–Jacobi–Bellman equation for continuous-time systems. These equations are usually difficult to solve analytically even if the system model is available. For most results on adaptive dynamic programming methods, they try to solve the Bellman equation or the Hamilton–Jacobi–Bellman equation approximately. To discuss a few, for the optimal control of discrete-time linear systems, the policy iteration and value iteration methods were suggested in [27] by using input and output data. The nonlinear pure-feedback systems was considered in [29] and an novel adaptive-critic-based NN controller was proposed. The adaptive dynamic programming [31] was employed to the problems of optimal switching and control design of nonlinear switching

Manuscript received October 9, 2017; revised December 28, 2017; accepted March 20, 2018. Date of publication April 27, 2018; date of current version November 15, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61503377, Grant 61533017, and Grant U1501251, and in part by the Qatar National Research Fund under National Priority Research Project under Grant NPRP9-466-1-103. This paper was recommended by Associate Editor H. Li. *(Corresponding author: Biao Luo.)*

B. Luo is with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: biao.luo@hotmail.com).

Y. Yang is with the College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar (e-mail: yyang@qf.org.qa).

D. Liu is with the School of Automation, Guangdong University of Technology, Guangzhou 510006, China (e-mail: derongliu@foxmail.com).

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

Digital Object Identifier 10.1109/TCYB.2018.2821369

systems. For the tracking control problem of linear continuous-time systems with partially unknown dynamics, an online integral RL method [33] was used to learn the solution of the algebraic Riccati equation. The concepts of exploration, integral temporal difference and invariant admissibility were introduced in [40], and integral RL methods were investigated. With the consideration of limits of computational and communication sources, RL methods [42], [45] have been employed to solve event-triggered control problems in recent years.

Although RL techniques have received extensive attention in solving control problems, only few results [52]–[59] have been reported on using QL for optimal control design. QL is a powerful method in RL, which was proposed by Watkins and Dayan [3]. In QL, the action-state value function, i.e., Q-function, is used. Due to the action in the Q-function can be given randomly, it can be implemented in an off-policy scheme and overcomes the "inadequate exploration" problem. For the optimal control design problems, it can be converted to find the optimal Q-function and then the thoughts of QL can be used. For linear systems, the QL methods were proposed for discrete-time systems [52], [53], [56] and continuous-time systems [54], [57]. In [59], the critic-only QL method was developed for tracking control problems which starts from an initial admissible control policy. In addition, the QL method was also applied to path-planning of mobile robots [55] and battery management in smart residential environments [58]. Generally speaking, the use of QL techniques for optimal control design and its theoretical problems are far from solved and still remain open issues that require further investigation.

In this paper, the data-based optimal output regulation problem is considered and an adaptive QL method is developed. Contributions of this paper can be briefly summarized as follows.

1) A novel adaptive QL algorithm is developed and its convergence theories are established.
2) In the adaptive QL, an adaptive parameter α_i in the policy evaluation is employed to achieve tradeoff between the current and future Q-functions, which avoids the requirement of an initial admissible control policy.
3) The developed adaptive QL is a data-based off-policy learning method, where the system data used for policy evaluation can be generated with exploratory control actions. Thus, the experience replay technique is employed, which makes the implementation of the adaptive QL simple and efficient.

The rest of this paper is arranged as follows. The problem description is presented in Section II. The off-policy adaptive QL algorithm is proposed in Section III and its implementation procedure is developed in Section IV. The adaptive QL method is simplified for linear systems in Section V. Finally, the simulation results are demonstrated in Section VI and the conclusions are given in Section VII.

Notation: $\mathbb{R}^n$ is the set of n-dimensional Euclidean space. The superscript T is used for the transpose of a matrix or vector. $\mathcal{X}$ and $\mathcal{U}$ are two compact sets, and denote $\mathcal{D} \triangleq \{(x,u)|x \in \mathcal{X}, u \in \mathcal{U}\}$. $\|f(x,u)\|_\infty$ is the maximum norm of function $f(x,u)$ defined on $\mathcal{X} \times \mathcal{U}$. $\otimes$ denotes Kronecker product and $\mathrm{vec}(\cdot)$ is matrix vectorization operation.

II. Problem Description

Let us consider the following general nonaffine nonlinear discrete-time systems:

$$\begin{cases} x(k+1) = f(x(k), u(k)) \\ y(k) = h(x(k)) \end{cases} \tag{1}$$

where $x(k) = [x_1(k), \ldots, x_n(k)]^\mathsf{T} \in \mathbb{R}^n$ is the state, $y(k) = [y_1(k), \ldots, y_m(k)]^\mathsf{T} \in \mathbb{R}^m$ is the output and $u(k) = [u_1(k), \ldots, u_p(k)]^\mathsf{T} \in \mathbb{R}^p$ is the control input. Assume that $f(x,u)$ and $h(x)$ are continuous nonlinear vector functions with $f(0,0) = 0, h(0) = 0$, and $x = 0$ is the unique equilibrium on $\mathcal{X}$. The system (1) is stabilizable on $\mathcal{X}$, i.e., there exists a continuous control function $u(x)$ such that the system is asymptotically stable on $\mathcal{X}$.

In this paper, the data-based optimal output regulation problem is considered, where the analytical expressions of $f(x,u)$ is *unknown*. The objective is to learn the control $u(k)$ from data of system (1), such that the output $y(k)$ approaches zero and minimize the following performance index:

$$J(y(0), u) \triangleq \sum_{l=0}^{\infty} U(y(l), u(l)) \tag{2}$$

where $U(y,u)$ denotes the utility function that is a positive definite function, i.e., $U(y,u) > 0$ for all $y, u \neq 0$, and $U(y,u) = 0$ only when $y = 0, u = 0$. $U(y,u)$ is measurable with *unknown* mathematical expression.

III. Adaptive Q-Learning

It is noticed that the mathematical models of $f(x,u)$ and $U(y,u)$ are unknown. In this section, an off-policy adaptive QL is developed to solve the data-based optimal output regulation problem and its convergence theory is established.

A. Off-Policy Adaptive Q-Learning

Before deriving the off-policy adaptive QL algorithm, some preliminaries are required. For description convenience, denote $\mathcal{R}(x,u) \triangleq U(h(x),u) = U(y,u)$. Let $u(x)$ be a stablizing control policy for system (1), and define its state value function as follows:

$$V_u(x(k)) \triangleq \sum_{l=k}^{\infty} \mathcal{R}(x(l), u(x(l))) \tag{3}$$

where $V_u(0) = 0$. According (3), the state-value function $V_u(x)$ satisfies the following equation:

$$V_u(x(k)) = \mathcal{R}(x(k), u(x(k))) + V_u(x(k+1)). \tag{4}$$

Then, the optimal state-value function is represented as

$$V^*(x) \triangleq V_{u^*}(x) = \min_u V_u(x) \tag{5}$$

which satisfies the following Bellman equation:

$$V^*(x(k)) = \min_a \big\{\mathcal{R}(x(k), a) + V^*(x(k+1))\big\}. \tag{6}$$

Then, the optimal control policy is given by

$$u^*(x(k)) = \arg\min_a \big\{\mathcal{R}(x(k), a) + V^*(x(k+1))\big\}. \tag{7}$$

Algorithm 1 Adaptive QL

- *Step 1:* Give $Q^{(0)}(x, a)$ and let $i = 0$.
- *Step 2:* (**Policy improvement**) Update control policy with

$$u^{(i)}(x) = \arg\min_a Q^{(i)}(x, a). \tag{11}$$

- *Step 3:* (**Policy evaluation**) Solve the equation

$$\begin{aligned} Q^{(i+1)}(x(k), a) &= \mathcal{R}(x(k), a) \\ &\quad + \alpha_i Q^{(i+1)}(x(k+1), u^{(i)}(x(k+1))) \\ &\quad + (1-\alpha_i) Q^{(i)}(x(k+1), u^{(i)}(x(k+1))) \end{aligned} \tag{12}$$

  for $Q^{(i+1)}(x, a)$, where $0 < \alpha_i < 1$ is an adaptive parameter.
- *Step 4:* If $\|Q^{(i+1)}(x, a) - Q^{(i)}(x, a)\|_\infty \le \varepsilon$ for a small constant $\varepsilon > 0$, terminate iteration, else, let $i = i + 1$, go back to Step 2 and continue. □

The proposed adaptive QL algorithm is based on the Q-function, which is an action-state function defined as

$$\mathcal{Q}_u(x(k), a) \triangleq \mathcal{R}(x(k), a) + V_u(x(k+1)) \tag{8}$$

where $\mathcal{Q}_u(0, 0) = 0$. For the optimal control policy u^*, denote its optimal Q-function as $\mathcal{Q}^*(x, a) \triangleq \mathcal{Q}_{u^*}(x, a)$ that is given by

$$\begin{aligned} \mathcal{Q}^*(x(k), a) &= \mathcal{R}(x(k), a) + \mathcal{Q}^*\big(x(k+1), u^*(x(k+1))\big) \\ &= \mathcal{R}(x(k), a) + V^*(x(k+1)). \end{aligned} \tag{9}$$

Based on (7) and (9), we have

$$u^*(x) = \arg\min_a \mathcal{Q}^*(x, a). \tag{10}$$

From (10), the design of the optimal control policy u^* is converted to finding the optimal Q-function $\mathcal{Q}^*(x, a)$.

To learn $\mathcal{Q}^*(x, a)$ from real system data, Algorithm 1 is proposed.

Remark 1: It is worth mentioning that the adaptive QL algorithm, i.e., Algorithm 1, has three improtant features. First, it is an off-policy learning method [4], [36], [60], [61], which overcomes the inadequate exploration problem by using data generated with arbitrary control actions a. Second, the adaptive QL algorithm uses an adaptive parameter α_i to achieve a tradeoff between the current and future Q-functions $Q^{(i)}$ and $Q^{(i+1)}$. The larger α_i means that $Q^{(i)}$ plays less and $Q^{(i+1)}$ plays more. Furthermore, the adaptive QL algorithm is a data-based approach, where the analytic mathematical expressions of system model and utility function are not required.

Remark 2: It is worthwhile to give some further discussions about off-policy learning methods. Off-policy learning and on-policy learning are two important frameworks of RL [4]. The main difference between two frameworks is that their policy evaluation operations are different. Before analyzing their differences, it requires to have a brief introduction on target policy and behavior policy. Generally speaking, the target policy represents a policy required to be evaluated for its value function. The behavior policies are the policies used for generating data for learning. For on-policy learning methods, the policy evaluation for a target control policy u requires generating system data by using the same policy u. It is usually difficult to implement and will often result in the inadequate exploration problem [4], [36]. For the policy evaluation in off-policy learning methods, the target policy can be evaluated with system data generated by arbitrary behavior policies, which means that any daily operational data of the real system is useful for learning. Thus, the implementation of off-policy learning methods is much simpler and can overcome the inadequate exploration problem [4], [36]. From step 3 of Algorithm 1, it is noted that behavior policies a can be arbitrary, which means that the adaptive QL is an off-policy learning method.

B. Convergence Analysis for Adaptive QL Algorithm

Note that the adaptive QL algorithm will generate a sequence $\{Q^{(i)}(x, a)\}$. In this section, the convergence of the adaptive QL algorithm will be proved by showing the sequence $\{Q^{(i)}(x, a)\}$ converges to the optimal Q-function $\mathcal{Q}^*(x, a)$. Before starting, the following conclusions in Theorem 1 are required.

Theorem 1: Let $Q(x, a) \ge 0$ satisfies

$$Q(x(k), a) \ge \mathcal{R}(x(k), a) + Q(x(k+1), \mu(x(k+1))) \tag{13}$$

where

$$\mu(x) = \arg\min_a Q(x, a). \tag{14}$$

Letting $Z^{(0)}(x, a) = Q(x, a)$, the sequence $\{Z^{(j)}(x, a)\}$ is generated with

$$\begin{aligned} Z^{(j+1)}(x(k), a) &= \mathcal{R}(x(k), a) + \alpha Z^{(j)}(x(k+1), \mu(x(k+1))) \\ &\quad + (1-\alpha) Q(x(k+1), \mu(x(k+1))) \end{aligned} \tag{15}$$

where $0 < \alpha < 1$. Let $\underline{Q}(x, a)$ be the solution of the following equation:

$$\begin{aligned} \underline{Q}(x(k), a) &= \mathcal{R}(x(k), a) + \alpha \underline{Q}(x(k+1), \mu(x(k+1))) \\ &\quad + (1-\alpha) Q(x(k+1), \mu(x(k+1))). \end{aligned} \tag{16}$$

Then, for all j:

1) $Z^{(j+1)}(x, a) \le Z^{(j)}(x, a)$;
2) $\|Z^{(j+1)}(x, a) - \underline{Q}(x, a)\|_\infty \le [\alpha/(1-\alpha)]\|Z^{(j+1)}(x, a) - Z^{(j)}(x, a)\|_\infty$;
3) $\|Z^{(j)}(x, a) - \underline{Q}(x, a)\|_\infty \le [\alpha^j/(1-\alpha)]\|Z^{(1)}(x, a) - Z^{(0)}(x, a)\|_\infty$;
4) $\lim_{j\to\infty} Z^{(j)}(x, a) = \underline{Q}(x, a)$.

Proof: 1) The mathematical induction is employed to prove part 1) of Theorem 1. For index $j = 0$, it follows from (13)–(15) that:

$$\begin{aligned} Z^{(1)}(x(k), a) &= \mathcal{R}(x(k), a) + \alpha Z^{(0)}(x(k+1), \mu(x(k+1))) \\ &\quad + (1-\alpha) Q(x(k+1), \mu(x(k+1))) \\ &= \mathcal{R}(x(k), a) + \alpha Q(x(k+1), \mu(x(k+1))) \\ &\quad + (1-\alpha) Q(x(k+1), \mu(x(k+1))) \\ &= \mathcal{R}(x(k), a) + Q(x(k+1), \mu(x(k+1))) \\ &\le Q(x(k), a) \\ &= Z^{(0)}(x(k), a). \end{aligned} \tag{17}$$

Assume that $Z^{(j)}(x,a) \le Z^{(j-1)}(x,a)$. Based on (15), we have

$$\begin{aligned} Z^{(j+1)}(x(k),a) &= \mathcal{R}(x(k),a) + \alpha Z^{(j)}(x(k+1),\mu(x(k+1))) \\ &\quad + (1-\alpha)Q(x(k+1),\mu(x(k+1))) \\ &\le \mathcal{R}(x(k),a) + \alpha Z^{(j-1)}(x(k+1),\mu(x(k+1))) \\ &\quad + (1-\alpha)Q(x(k+1),\mu(x(k+1))) \\ &= Z^{(j)}(x(k),a). \end{aligned}$$

It means that part 1) of Theorem 1 holds for all j.

2) According to (15) and (16)

$$\begin{aligned} &Z^{(j+1)}(x(k),a) - \underline{Q}(x(k),a) \\ &= \alpha\Big[Z^{(j)}(x(k+1),\mu(x(k+1))) - \underline{Q}(x(k),\mu(x(k+1)))\Big] \\ &= \alpha\Big[Z^{(j+1)}(x(k+1),\mu(x(k+1))) \\ &\qquad - \underline{Q}(x(k),\mu(x(k+1)))\Big] \\ &\quad - \alpha\Big[Z^{(j+1)}(x(k+1),\mu(x(k+1))) \\ &\qquad - Z^{(j)}(x(k+1),\mu(x(k+1)))\Big]. \end{aligned}$$

Then, for all $x(k), x(k+1) \in \mathcal{X}$ and $a, \mu(x(k+1)) \in \mathcal{U}$

$$\begin{aligned} &\|Z^{(j+1)}(x,a) - \underline{Q}(x,a)\|_\infty \\ &\le \alpha\|Z^{(j+1)}(x,a) - \underline{Q}(x,a)\|_\infty \\ &\quad + \alpha\|Z^{(j+1)}(x,a) - Z^{(j)}(x,a)\|_\infty. \end{aligned} \tag{18}$$

Thus, part 2) of Theorem 1 holds.

3) Based on part 2) of Theorem 1

$$\begin{aligned} &\|Z^{(j)}(x,a) - \underline{Q}(x,a)\|_\infty \\ &\le \frac{\alpha}{1-\alpha}\|Z^{(j)}(x,a) - Z^{(j-1)}(x,a)\|_\infty \\ &= \frac{\alpha^2}{1-\alpha}\|Z^{(j-1)}(x,a) - Z^{(j-2)}(x,a)\|_\infty \\ &\quad \vdots \\ &= \frac{\alpha^j}{1-\alpha}\|Z^{(1)}(x,a) - Z^{(0)}(x,a)\|_\infty. \end{aligned}$$

4) From part 3) of Theorem 1, $\lim_{j\to\infty}\|Z^{(j)}(x,a) - \underline{Q}(x,a)\|_\infty = 0$, i.e., $\lim_{j\to\infty} Z^{(j)}(x,a) = \underline{Q}(x,a)$. ■

The convergence of Algorithm 1 is proved in the following Theorem 2 with the use of the conclusions in Theorem 1.

Theorem 2: Let the sequence $\{Q^{(i)}(x,a)\}$ be generated by Algorithm 1. Let the condition $Q^{(0)}(x(k),a) \ge \mathcal{R}(x(k),a) + Q^{(0)}(x(k+1),u^{(0)}(x(k+1)))$ hold. Then

1) for all i

$$\begin{aligned} Q^{(i+1)}(x(k),a) &\le \mathcal{R}(x(k),a) + Q^{(i)}\Big(x(k+1),u^{(i)}(x(k+1))\Big) \\ &\le Q^{(i)}(x(k),a) \end{aligned} \tag{19}$$

2) $\lim_{i\to\infty} Q^{(i)}(x,a) = \mathcal{Q}^*(x,a)$.

Proof: 1) The mathematical induction is employed to prove the conclusion (19) based on Theorem 1.

For index $i = 0$, let $Z^{(0)}(x,a) = Q(x,a) = Q^{(0)}(x,a)$ and $\mu(x) = u^{(0)}(x)$. It follows from Theorem 1 that $Z^{(j+1)}(x,a) \le Z^{(j)}(x,a)$ and $\lim_{j\to\infty} Z^{(j)}(x,a) = Q^{(1)}(x,a)$. Then, we have

$$\begin{aligned} Q^{(1)}(x(k),a) &= \lim_{j\to\infty} Z^{(j)}(x(k),a) \\ &\le Z^{(1)}(x(k),a) \\ &= \mathcal{R}(x(k),a) + \alpha_0 Z^{(0)}(x(k+1),\mu(x(k+1))) \\ &\quad + (1-\alpha_0)Q^{(0)}(x(k+1),\mu(x(k+1))) \\ &= \mathcal{R}(x(k),a) + Q^{(0)}(x(k+1),\mu(x(k+1))). \end{aligned}$$

This means that the conclusion (19) holds for $i = 0$.

Assume that the conclusion (19) holds for index $i-1$, i.e.,

$$\begin{aligned} &Q^{(i)}(x(k),a) \\ &\le \mathcal{R}(x(k),a) + Q^{(i-1)}\Big(x(k+1),u^{(i-1)}(x(k+1))\Big) \\ &\le Q^{(i-1)}(x(k),a). \end{aligned} \tag{20}$$

From (11) and (12)

$$\begin{aligned} &Q^{(i)}(x(k),a) = \mathcal{R}(x(k),a) \\ &\quad + \alpha_{i-1}Q^{(i)}\Big(x(k+1),u^{(i-1)}(x(k+1))\Big) \\ &\quad + (1-\alpha_{i-1})Q^{(i-1)}\Big(x(k+1),u^{(i-1)}(x(k+1))\Big) \\ &= \mathcal{R}(x(k),a) + Q^{(i)}\Big(x(k+1),u^{(i-1)}(x(k+1))\Big) \\ &\quad + (1-\alpha_{i-1})\Big[Q^{(i-1)}\Big(x(k+1),u^{(i-1)}(x(k+1))\Big) \\ &\qquad - Q^{(i)}\Big(x(k+1),u^{(i-1)}(x(k+1))\Big)\Big] \\ &\ge \mathcal{R}(x(k),a) + Q^{(i)}\Big(x(k+1),u^{(i-1)}(x(k+1))\Big) \\ &\ge \mathcal{R}(x(k),a) + Q^{(i)}\Big(x(k+1),u^{(i)}(x(k+1))\Big). \end{aligned} \tag{21}$$

Next, the conclusions of Theorem 1 is used to prove the left side of the conclusion (19). Let $Z^{(0)}(x,a) = Q(x,a) = Q^{(i)}(x,a)$ and $\mu(x) = u^{(i)}(x)$. It follows from Theorem 1 that $Z^{(j+1)}(x,a) \le Z^{(j)}(x,a)$ and $\lim_{j\to\infty} Z^{(j)}(x,a) = Q^{(i+1)}(x,a)$. Then, we have

$$\begin{aligned} Q^{(i+1)}(x(k),a) &= \lim_{j\to\infty} Z^{(j)}(x(k),a) \\ &\le Z^{(1)}(x(k),a) \\ &= \mathcal{R}(x(k),a) + \alpha_i Z^{(0)}\Big(x(k+1),u^{(i)}(x(k+1))\Big) \\ &\quad + (1-\alpha_i)Q^{(i)}\Big(x(k+1),u^{(i)}(x(k+1))\Big) \\ &= \mathcal{R}(x(k),a) + Q^{(i)}\Big(x(k+1),u^{(i)}(x(k+1))\Big). \end{aligned} \tag{22}$$

The combination of (21) and (22) demonstrates that the conclusion (19) holds for all i.

2) From (19), $\{Q^{(i)}(x,a)\}$ is a nonincreasing sequence and lower bounded by $Q^{(i)}(x,a) \ge 0$. Since a bounded monotone sequence always has a limit, we denote it by $Q^{(\infty)}(x,a) \triangleq \lim_{i\to\infty} Q^{(i)}(x,a)$ and $u^{(\infty)}(x,a) \triangleq \arg\min_a Q^{(\infty)}(x,a)$. Taking limit on (19) yields

$$\begin{aligned} &Q^{(\infty)}(x(k),a) \\ &\le \mathcal{R}(x(k),a) + Q^{(\infty)}\Big(x(k+1),u^{(\infty)}(x(k+1))\Big) \\ &\le Q^{(\infty)}(x(k),a) \end{aligned}$$

i.e.,

$$Q^{(\infty)}(x(k),a) = \mathcal{R}(x(k),a) + Q^{(\infty)}\Big(x(k+1), u^{(\infty)}(x(k+1))\Big)$$

which is the same as (9), i.e., $Q^{(\infty)}(x,a) = \mathcal{Q}^*(x,a)$. ■

Corollary 1: Let $0 < \beta_1 \leq \beta_2 < 1$, and $Q(x,a) \geq 0$ satisfies

$$Q(x(k),a) \geq \mathcal{R}(x(k),a) + Q(x(k+1), \mu(x(k+1))) \quad (23)$$

where

$$\mu(x) = \min_a Q(x,a). \quad (24)$$

For β_l, $l = 1, 2$, $Q_{\beta_l}(x,a)$ satisfies

$$Q_{\beta_l}(x(k),a) = \mathcal{R}(x(k),a) + \beta_l Q_{\beta_l}(x(k+1), \mu(x(k+1))) + (1-\beta_l)Q(x(k+1), \mu(x(k+1))). \quad (25)$$

Then, $Q_{\beta_1}(x,a) \geq Q_{\beta_2}(x,a)$.

Proof: For $l = 1, 2$, let $Z^{(0)}_{\beta_l}(x,a) = Q(x,a)$ and the sequence $\{Q^{(j)}_{\beta_l}(x,a)\}$ be generated with

$$Z^{(j+1)}_{\beta_l}(x(k),a) = \mathcal{R}(x(k),a) + \beta_l Z^{(j)}_{\beta_l}(x(k+1), \mu(x(k+1))) + (1-\beta_l)Q(x(k+1), \mu(x(k+1))). \quad (26)$$

According to the part 1) of Theorem 1, we have $Z^{(j+1)}_{\beta_l}(x,a) \leq Z^{(j)}_{\beta_l}(x,a)$ for all j and $l = 1, 2$.

Next, the mathematical induction is employed to prove $Z^{(j)}_{\beta_1}(x,a) \geq Z^{(j)}_{\beta_2}(x,a)$ for all j. Note that $Z^{(0)}_{\beta_1}(x,a) = Z^{(0)}_{\beta_2}(x,a) = Q(x,a)$. Assume that $Z^{(m)}_{\beta_1}(x) \geq Z^{(m)}_{\beta_2}(x)$ for $m = j-1$. We will show it for $m = j$. From (26), we have

$$\begin{aligned} Z^{(j)}_{\beta_1}(x(k),a) &= \mathcal{R}(x(k),a) + \beta_1 Z^{(j-1)}_{\beta_1}(x(k+1), \mu(x(k+1))) \\ &\quad + (1-\beta_1)Q(x(k+1), \mu(x(k+1))) \\ &\geq \mathcal{R}(x(k),a) + \beta_1 Z^{(j-1)}_{\beta_2}(x(k+1), \mu(x(k+1))) \\ &\quad + (1-\beta_1)Q(x(k+1), \mu(x(k+1))). \end{aligned} \quad (27)$$

It follows from (26) and (27) that:

$$\begin{aligned} &Z^{(j)}_{\beta_1}(x(k),a) - Z^{(j)}_{\beta_2}(x(k),a) \\ &\geq (\beta_1 - \beta_2) Z^{(j-1)}_{\beta_2}(x(k+1), \mu(x(k+1))) \\ &\quad - (\beta_1 - \beta_2) Q(x(k+1), \mu(x(k+1))) \\ &= (\beta_2 - \beta_1)[Q(x(k+1), \mu(x(k+1))) \\ &\qquad - Z^{(j-1)}_{\beta_2}(x(k+1), \mu(x(k+1)))] \\ &= (\beta_2 - \beta_1)\Big[Z^{(0)}_{\beta_2}(x(k+1), \mu(x(k+1))) \\ &\qquad - Z^{(j-1)}_{\beta_2}(x(k+1), \mu(x(k+1)))\Big] \\ &\geq 0 \end{aligned} \quad (28)$$

which means that $Z^{(j)}_{\beta_1}(x,a) \geq Z^{(j)}_{\beta_2}(x,a)$ holds for all j. From the part 4) of Theorem 1, we have $Q_{\beta_l}(x,a) = \lim_{j\to\infty} Z^{(j)}_{\beta_l}(x,a)$ for $j = 1, 2$. Then, it follows from (28) that $Q_{\beta_1}(x,a) \geq V_{\beta_2}(x,a)$. This completes the proof. ■

In fact, the conclusions of Theorem 1 represent one policy evaluation operation (12) in Algorithm 1. That is to say, for all i, if letting $Z^{(0)}(x,a) = Q^{(i)}(x,a)$, the sequence $\{Z^{(j)}(x,a)\}$ generated by (15) is nonincreasing and converges to $Q^{(i+1)}(x,a)$, i.e., $\lim_{j\to\infty} Z^{(j)}(x,a) = Q^{(i+1)}(x,a)$. Based on the conclusions of Theorem 1, the convergence of adaptive QL algorithm (i.e., Algorithm 1) is proved in Theorem 2 by demonstrating that $\{Q^{(i)}(x,a)\}$ is a nonincreasing sequence and converges to the optimal Q-function $\mathcal{Q}^*(x,a)$. Furthermore, it is proved in Corollary 1 that a larger adaptive parameter α_i in Algorithm 1 may result in a better Q-function at each iteration under the condition (23).

IV. Actor-Critic Structure With Experience Replay

In this section, the actor-critic structure is developed to implement the adaptive QL algorithm. The actor and critic NNs are employed to approximate the Q-function and the control policy, respectively. By using the experience replay technique, the actor NN weights are updated with least-squares scheme and the critic NN weights are updated with batch gradient descent method. Moreover, an adaptive rule is proposed to tune the parameter α_i in the adaptive QL algorithm.

A. Actor and Critic Neural Networks With Experience Replay

It is known that NN is an universal approximator [62], which can be employed to approximate any continuous function on a compact set. By using NNs, the output of critic and actor NNs are given by

$$Q^{(i)}(x,a) \triangleq \sum_{j=1}^{L_1} \theta^{(i)}_j \psi_j(x,a) = \Psi^{\mathsf{T}}_L(x,a)\theta^{(i)} \quad (29)$$

$$u^{(i)}(x) \triangleq \sum_{j=1}^{L_2} \beta^{(i)}_j \phi_j(x) = \Phi^{\mathsf{T}}_L(x)\beta^{(i)} \quad (30)$$

where $\theta^{(i)} \triangleq [\theta^{(i)}_1 \ \cdots \ \theta^{(i)}_{L_1}]^{\mathsf{T}}$ and $\beta^{(i)} \triangleq [\beta^{(i)}_1 \ \cdots \ \beta^{(i)}_{L_2}]^{\mathsf{T}}$ denote the critic and actor NN weights, $\Psi_L(x,a) \triangleq [\psi_1(x,a) \ \cdots \ \psi_{L_1}(x,a)]^{\mathsf{T}}$ and $\Phi_L(x) \triangleq [\phi_1(x) \ \cdots \ \phi_{L_2}(x)]^{\mathsf{T}}$ are NN activation function vectors.

Remark 3: Like all function approximation techniques, large size of activation functions for critic and actor NNs in (29) and (30) can improve the estimation performance at the price of higher computational effort. Thus, an appropriate selection of $\Psi_L(x,a)$, $\Phi_L(x)$ and their size are useful to balance the performance and computation. However, it is still difficult to develop a general optimal selection rule for all systems. This is simply because the optimal selection is usually different for different systems. In a word, for a specific system, prior experiences would be helpful for the selection of NN activation functions and their sizes.

To learn $\theta^{(i)}$ and $\beta^{(i)}$, real system data is required and experience replay technique is employed. In the experience replay technique, the training data set $\mathcal{S}_M$ is collected and stored before starting the learning procedure. That is to say, at each iteration of the adaptive QL algorithm, it learns $\theta^{(i)}$

and $\beta^{(i)}$ directly based on $\mathcal{S}_M$ which does not require to collect any additional data from system. Denote the training data set as $\mathcal{S}_M \triangleq \{(x_{[l]}, a_{[l]}, x'_{[l]}, \mathcal{R}_{[l]}) | (x_{[l]}, a_{[l]}) \in \mathcal{D}, x'_{[l]} \in \mathcal{X}, l = 1, 2, \ldots, M\}$, where M is size of the data set, $\mathcal{R}_{[l]} = \mathcal{R}(x_{[l]}, a_{[l]})$ and $x'_{[l]}$ represents the next system state under control action $a_{[l]}$ at state $x_{[l]}$.

Remark 4: The experience replay technique was introduced in [63]. By using a memory buffer to store historical experiences (data) collected from real environment or system, these data will be sampled repeatedly for updating NN weights during the learning process. Generally speaking, experience replay can reduce the amount of experience required during learning [64]. The experience replay technique became extremely popular after it was employed in the deep Q-network algorithm [1]. Recently, the experience replay technique has also been employed to solve control problems [65], [66]. In this paper, the developed adaptive QL is an off-policy method, where the data for learning can be generated with arbitrary behavior policies. Therefore, the experiences are easy to collect and the experience replay technique can be implemented for the adaptive QL algorithm.

For the update of the actor NN weights $\beta^{(i)}$, the batch gradient descent method is developed. Based on (11), (29), and (30), the computation of the actor NN weights $\beta^{(i)}$ aims to solve the following optimization problem:

$$\beta^{(i)} = \arg\min_{\beta} \Psi_L^{\mathsf{T}}\Big(x, \Phi_L^{\mathsf{T}}(x)\beta\Big)\theta^{(i)}. \tag{31}$$

The actor NN weights $\beta^{(i)}$ is computed to minimize the following sum of square over the data set $\mathcal{S}_M$:

$$L(\beta) = \frac{1}{M}\sum_{l}^{M} \Psi_L^{\mathsf{T}}\Big(x_{[l]}, \Phi_L^{\mathsf{T}}(x_{[l]})\beta\Big)\theta^{(i)}. \tag{32}$$

Based on the batch gradient descent method, the following iterative scheme is employed:

$$\beta^{(i,j+1)} = \beta^{(i,j)} - \lambda \frac{\partial L(\beta)}{\partial \beta}\Big|_{\beta=\beta^{(i,j)}} \tag{33}$$

where $\lambda > 0$ is the step-size. According to (32) and (33), we have

$$\beta^{(i,j+1)} = \beta^{(i,j)} - \frac{\lambda}{M}\sum_{l}^{M}\left[\Big(\theta^{(i)}\Big)^{\mathsf{T}} \frac{\partial \Psi_L(x_{[l]}, a)}{\partial a}\Big|_{a=\Phi_L^{\mathsf{T}}(x_{[l]})\beta^{(i,j)}} \Phi_L(x_{[l]})\right]. \tag{34}$$

To learn the unknown critic NN weights $\theta^{(i+1)}$, the following linear equation is constructed based on (12) for each data $(x_{[l]}, a_{[l]}, x'_{[l]}, \mathcal{R}_{[l]}) \in \mathcal{S}_M$:

$$\Psi_L^{\mathsf{T}}(x_{[l]}, a_{[l]})\theta^{(i+1)} = \mathcal{R}_{[l]} + \alpha_i \Psi_L^{\mathsf{T}}\Big(x'_{[l]}, u^{(i)}(x'_{[l]})\Big)\theta^{(i+1)} + (1-\alpha_i)\Psi_L^{\mathsf{T}}\Big(x'_{[l]}, u^{(i)}(x'_{[l]})\Big)\theta^{(i)}. \tag{35}$$

Note that (35) are linear equations ($l = 1, 2, \ldots, M$) with respect to the unknown critic NN weights $\theta^{(i+1)}$. Thus, the following least-squares scheme is employed to compute $\theta^{(i+1)}$ in (35):

$$\theta^{(i+1)} = \left[\Big(Z^{(i)}\Big)^{\mathsf{T}} Z^{(i)}\right]^{-1}\Big(Z^{(i)}\Big)^{\mathsf{T}}\eta^{(i)} \tag{36}$$

where $Z^{(i)} \triangleq [z_{[1]}^{(i)}, \ldots, z_{[M]}^{(i)}]^{\mathsf{T}}$ and $\eta^{(i)} \triangleq [\eta_{[1]}^{(i)}, \ldots, \eta_{[M]}^{(i)}]^{\mathsf{T}}$ with $z_{[l]}^{(i)} = \Psi_L^{\mathsf{T}}(x_{[l]}, a_{[l]}) - \alpha_i \Psi_L^{\mathsf{T}}(x'_{[l]}, u^{(i)}(x'_{[l]}))$ and $\eta_{[l]}^{(i)} = \mathcal{R}_{[l]} + (1-\alpha_i)\Psi_L^{\mathsf{T}}(x'_{[l]}, u^{(i)}(x'_{[l]}))\theta^{(i)}$.

Algorithm 2 Implementation Procedure of Adaptive QL

- *Step 1:* Collect the data set $\mathcal{S}_M$. Give $Q^{(0)}(x, a)$ and let $i = 0$;
- *Step 2:* Compute the actor NN weight vector $\beta^{(i)}$ using (34) as j increases till convergence.
- *Step 3:* Compute the critic NN weight vector $\theta^{(i+1)}$ using (36).
- *Step 4:* If $\|\theta^{(i+1)} - \theta^{(i)}\| \leqslant \epsilon$ for small $\epsilon > 0$, terminate iteration. Else, let $i = i + 1$, go back to Step 2 and continue.

B. Implementation of Adaptive Q-Learning

By using (34) and (36) for updating the actor and critic NN weights, Algorithm 2 is developed.

It is noted from Algorithm 2 that the developed adaptive QL method is completely data-based, which learns the weights of the actor and critic NNs using the data set $\mathcal{S}_M$. After convergence is achieved in Algorithm 2, the converged actor NN weights are employed in real control applications.

Remark 5: It is worthwhile to give a brief analysis on factors that affect the computational complexity of Algorithm 2. Note that the computation of the actor and critic NN weight vectors $\beta^{(i)}$ and $\theta^{(i+1)}$ with (34) and (36) is the most time-consuming part at each iteration. In (34) and (36), the computation is increasing with the parameters M, L_1, and L_2, where M is the size of the data set, L_1 and L_2 are the size of critic and actor NNs, respectively. The parameters M, L_1, and L_2 are usually affected by the dimension and the complexity of the system. That is to say, for complex or higher dimensional systems, more hidden-layer nodes (i.e., larger L_1, L_2) are required in critic and actor NNs, and more system data (i.e., larger M) is needed for learning their weights. Therefore, the dimension and the complexity of the system are the two main factors that affect the execution time of Algorithm 2.

C. Adaptive Tuning Rule for α_i

For the adaptive QL (i.e., Algorithm 1), it is noted that the parameter α_i can be different at each iteration. The following adaptive rule is proposed to tune α_i:

$$\alpha_i = \tanh(b\ln(i+1)) \tag{37}$$

where $b > 0$ is a constant. Note that (37) is a monotone increasing function with respect to the iteration i. If $\alpha_i \equiv 1$ for all i, the adaptive QL becomes a policy iteration method [59], which achieves fast convergence while requires an initial stabilizing control policy. If $\alpha_i \equiv 0$ for all i, the adaptive QL

becomes a value iteration method [41], which avoids the requirement of an initial stabilizing control policy while converges slowly. Thus, the rationale for the design of (37) is that a smaller α_i is employed at the beginning of the adaptive QL to avoid the requirement of an initial stabilizing control policy, and gradually increase α_i to speed up the adaptive QL.

V. Special Case: Linear Systems

Although the method is proposed for nonlinear systems, it is difficult to verify its effectiveness on nonlinear systems for their optimal control policy is usually unavailable for comparison in simulation studies. For simulation convenience, it is necessary to investigate the adaptive QL method on linear systems.

Considering the following linear discrete-time systems:

$$\begin{cases} x(k) = Ax(k) + Bu(k) \\ y(k) = Cx(k) \end{cases} \tag{38}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, and $C \in \mathbb{R}^{m\times n}$. For the performance index (2), the quadratic form is employed with $U(y,u) = y^{\mathsf{T}}Sy + u^{\mathsf{T}}Ru$, i.e., $\mathcal{R}(x,u) = x^{\mathsf{T}}C^{\mathsf{T}}SCx + u^{\mathsf{T}}Ru$. For this optimal output regulation problem, its optimal value function is $V^*(x) = x^{\mathsf{T}}Px$, where $P \geq 0$. From (7), we have $(\partial/\partial u^*)\{\mathcal{R}(x(k),u(x(k))) + V^*(x(k+1))\} = 0$, i.e.,

$$\begin{aligned} u^*(x(k)) &= -R^{-1}B^{\mathsf{T}}Px(k+1) \\ &= -R^{-1}B^{\mathsf{T}}P\big[Ax(k) + Bu^*(x(k))\big]. \end{aligned} \tag{39}$$

That is,

$$u^*(x(k)) = Kx(k) \tag{40}$$

where $K = -(R + B^{\mathsf{T}}PB)^{-1}B^{\mathsf{T}}PA$ is the gain of the optimal controller. For all $x(k) \in \mathcal{X}$, we have

$$V^*(x(k)) = \mathcal{R}\big(x(k), u^*(x(k))\big) + V^*(x(k+1)). \tag{41}$$

By using $V^*(x) = x^{\mathsf{T}}Px$, we obtain

$$\begin{aligned} x^{\mathsf{T}}(k)Px(k) = {} & x^{\mathsf{T}}(k)C^{\mathsf{T}}SCx(k) + \big[u^*(x(k))\big]^{\mathsf{T}}Ru^*(x(k)) \\ & + x^{\mathsf{T}}(k+1)Px(k+1). \end{aligned} \tag{42}$$

Based on (38) and (40)

$$\begin{aligned} x^{\mathsf{T}}(k+1)Px(k+1) &= x^{\mathsf{T}}(k)[A+BK]^{\mathsf{T}}P[A+BK]x(k) \\ &= x^{\mathsf{T}}(k)\Big[A^{\mathsf{T}}PA + A^{\mathsf{T}}PBK\Big]x(k) \\ &\quad + x^{\mathsf{T}}(k)[A+BK]^{\mathsf{T}}PBKx(k) \\ &= x^{\mathsf{T}}(k)\Big[A^{\mathsf{T}}PA + A^{\mathsf{T}}PBK\Big]x(k) \\ &\quad + x^{\mathsf{T}}(k+1)PBR^{-1}RKx(k). \end{aligned} \tag{43}$$

According to (39), we have $x^{\mathsf{T}}(k+1)PBR^{-1} = [u^*(x(k))]^{\mathsf{T}}(k)$. Thus, it follows from (43) that:

$$\begin{aligned} x^{\mathsf{T}}(k+1)Px(k+1) = {} & x^{\mathsf{T}}(k)\Big[A^{\mathsf{T}}PA + A^{\mathsf{T}}PBK\Big]x(k) \\ & -\big[u^*(x(k))\big]^{\mathsf{T}}(k)Ru^*(x(k)). \end{aligned} \tag{44}$$

The substitution of (44) into (42) yields

$$\begin{aligned} x^{\mathsf{T}}(k)Px(k) = {} & x^{\mathsf{T}}(k)C^{\mathsf{T}}SCx(k) \\ & + x^{\mathsf{T}}(k)\Big[A^{\mathsf{T}}PA + A^{\mathsf{T}}PBK\Big]x(k) \end{aligned}$$

holds for all $x(k) \in \mathcal{X}$. Then

$$P = C^{\mathsf{T}}SC + A^{\mathsf{T}}PA + A^{\mathsf{T}}PBK$$

i.e.,

$$C^{\mathsf{T}}SC - P + A^{\mathsf{T}}PA - A^{\mathsf{T}}PB\Big(R + B^{\mathsf{T}}PB\Big)^{-1}B^{\mathsf{T}}PA = 0. \tag{45}$$

Equation (45) is called algebraic Riccati equation. It is observed that the optimal control policy (40) depends on the solution P of the algebraic Riccati equation (45), which requires the system matrices A and B.

By using $V^*(x) = x^{\mathsf{T}}Px$, it follows from (9) that:

$$\begin{aligned} \mathcal{Q}^*(x(k),a) &= x^{\mathsf{T}}(k)C^{\mathsf{T}}SCx(k) + a^{\mathsf{T}}Ra \\ &\quad + [Ax(k)+Ba]^{\mathsf{T}}P[Ax(k)+Ba] \\ &= x^{\mathsf{T}}(k)\Big[C^{\mathsf{T}}SC + A^{\mathsf{T}}PA\Big]x(k) + 2x^{\mathsf{T}}(k)A^{\mathsf{T}}PBa \\ &\quad + a^{\mathsf{T}}\Big[R + B^{\mathsf{T}}PB\Big]a \\ &= \begin{bmatrix} x(k) \\ a \end{bmatrix}^{\mathsf{T}} G \begin{bmatrix} x(k) \\ a \end{bmatrix} \end{aligned} \tag{46}$$

where G is a block matrix given by

$$G \triangleq \begin{bmatrix} G_{11} & G_{12} \\ G_{12}^{\mathsf{T}} & G_{22} \end{bmatrix} \tag{47}$$

with $G_{11} \triangleq C^{\mathsf{T}}SC + A^{\mathsf{T}}PA$, $G_{12} \triangleq A^{\mathsf{T}}PB$ and $G_{22} \triangleq R + B^{\mathsf{T}}PB$. Based on (10), we have $(\partial\mathcal{Q}^*(x,a)/\partial a)|_{a=u^*(x)} = 0$. Then, it follows from (46) and (47) that:

$$u^*(x) = -G_{22}^{-1}G_{12}^{\mathsf{T}}x. \tag{48}$$

That means that the optimal control gain can also be represented as $K = -G_{22}^{-1}G_{12}^{\mathsf{T}}$.

For the adaptive QL algorithm, i.e., Algorithm 1, the iterative Q-function $Q^{(i)}(x,a)$ is denoted as

$$Q^{(i)}(x,a) = \begin{bmatrix} x \\ a \end{bmatrix}^{\mathsf{T}} G^{(i)} \begin{bmatrix} x \\ a \end{bmatrix} \tag{49}$$

where

$$G^{(i)} \triangleq \begin{bmatrix} G_{11}^{(i)} & G_{12}^{(i)} \\ \Big(G_{12}^{(i)}\Big)^{\mathsf{T}} & G_{22}^{(i)} \end{bmatrix}. \tag{50}$$

With (50), it follows from (11) and (49) that:

$$u^{(i)}(x) = K^{(i)}x \tag{51}$$

where $K^{(i)} \triangleq -[G_{22}^{(i)}]^{-1}[G_{12}^{(i)}]^{\mathsf{T}}$. To learn the iterative matrix $G^{(i)}$ from real system data, it follows from (12) and (49) that:

$$\begin{aligned}&\begin{bmatrix} x(k) \\ a \end{bmatrix}^{\mathsf{T}} G^{(i+1)} \begin{bmatrix} x(k) \\ a \end{bmatrix} \\ &\quad -\alpha_i \begin{bmatrix} x(k+1) \\ K^{(i)}x(k+1) \end{bmatrix}^{\mathsf{T}} G^{(i+1)} \begin{bmatrix} x(k+1) \\ K^{(i)}x(k+1) \end{bmatrix} \\ &= x^{\mathsf{T}}(k)C^{\mathsf{T}}SCx(k) + x^{\mathsf{T}}(k)\left[K^{(i)}\right]^{\mathsf{T}} RK^{(i)}x(k) \\ &\quad +(1-\alpha_i)\begin{bmatrix} x(k+1) \\ K^{(i)}x(k+1) \end{bmatrix}^{\mathsf{T}} G^{(i)} \begin{bmatrix} x(k+1) \\ K^{(i)}x(k+1) \end{bmatrix}. \end{aligned} \tag{52}$$

As described in Section IV-B, for each data $(x_{[l]}, a_{[l]}, x'_{[l]}, \mathcal{R}_{[l]}) \in \mathcal{S}_M$, (52) is rewritten as follows by using Kronecker product $\otimes$ and matrix vectorization operation vec$(\cdot)$:

$$z_{[l]}^{(i)}\theta^{(i+1)} = \eta_{[l]}^{(i)} \tag{53}$$

where $\theta^{(i+1)} = \text{vec}(G^{(i+1)})$, $Z^{(i)} \triangleq [z_{[1]}^{(i)}, \ldots, z_{[M]}^{(i)}]^{\mathsf{T}}$ and $\eta^{(i)} = [\eta_{[1]}^{(i)}, \ldots, \eta_{[M]}^{(i)}]^{\mathsf{T}}$ with

$$z_{[l]}^{(i)} = \left\{ \begin{bmatrix} x_{[l]} \\ a_{[l]} \end{bmatrix} \otimes \begin{bmatrix} x_{[l]} \\ a_{[l]} \end{bmatrix} - \alpha_i \begin{bmatrix} x'_{[l]} \\ K^{(i)}x'_{[l]} \end{bmatrix} \otimes \begin{bmatrix} x'_{[l]} \\ K^{(i)}x'_{[l]} \end{bmatrix} \right\}^{\mathsf{T}}$$

and

$$\begin{aligned}\eta_{[l]}^{(i)} &= \left[x'_{[l]} \otimes x'_{[l]}\right]^{\mathsf{T}} \text{vec}\left(C^{\mathsf{T}}SC + \left[K^{(i)}\right]^{\mathsf{T}} RK^{(i)}\right) \\ &\quad + (1-\alpha_i)\left\{ \begin{bmatrix} x'_{[l]} \\ K^{(i)}x'_{[l]} \end{bmatrix} \otimes \begin{bmatrix} x'_{[l]} \\ K^{(i)}x'_{[l]} \end{bmatrix} \right\}^{\mathsf{T}} \theta^{(i)}. \end{aligned}$$

Then, the least-squares method (36) can be used to compute $\theta^{(i+1)}$.

VI. Simulation Studies

The adaptive QL method is developed for nonlinear systems. However, the optimal control of nonlinear systems cannot be given analytically for comparison purpose. Thus, the effectiveness of adaptive QL method cannot be verified through simulation studies on nonlinear systems. Therefore, a linear numerical example is employed, which is treated as a nonlinear system during simulation. This is mainly because the optimal control policy and the optimal Q-function of the linear system can be given analytically, which is convenient to verify the effectiveness of the adaptive QL method by showing the iterative control policies and Q-functions converge to their optimums.

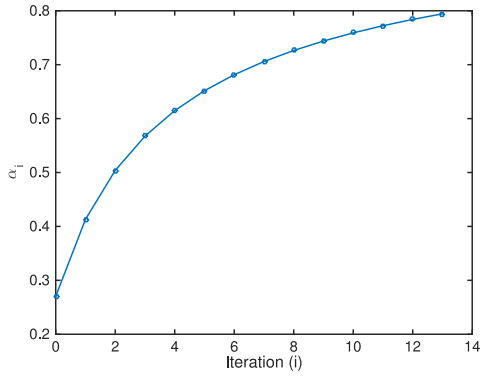

Fig. 1. Relationship between the iteration i and the parameter b.

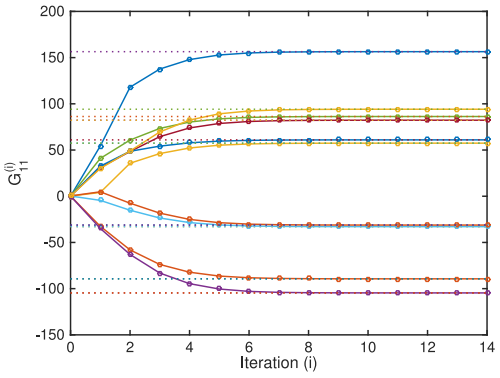

Fig. 2. Iterative $G_{11}^{(i)}$ at each iteration i.

Consider the linear system with the system matrices given as follows:

$$A = \begin{bmatrix} 0.8758 & -0.8242 & -0.0729 & -0.4742 \\ -0.5306 & -0.7615 & -0.0930 & -0.5801 \\ -0.4295 & 0.0434 & -0.1747 & -0.3058 \\ -0.0305 & 0.3551 & -0.7820 & 0.7308 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.9507 & 0.8429 \\ -0.5688 & -0.9364 \\ 0.7536 & -1.8178 \\ 0.0274 & 0.4811 \end{bmatrix}$$

$$C = \begin{bmatrix} -0.0390 & 0.8977 & 0.7337 & 0.6241 \\ 0.4288 & -0.8056 & 0.6309 & -0.4018 \\ -0.8298 & 0.4312 & 0.4571 & 0.6468 \end{bmatrix}$$

and $x(0) = [0.5, -0.5, 2, -1]^{\mathsf{T}}$. For the performance index, let $\mathcal{R}(x,u) = x^{\mathsf{T}}C^{\mathsf{T}}SCx + u^{\mathsf{T}}Ru$, where S and R are identity matrices. With MATLAB command DARE, the solution of the

$$G = \begin{bmatrix} 156.4314 & -89.3857 & 57.4382 & -104.6136 & -52.6086 & 84.6568 \\ -89.3857 & 86.2679 & -32.8981 & 82.2742 & 38.4252 & -27.9424 \\ 57.4382 & -32.8981 & 60.9841 & -31.1980 & -26.2625 & 45.6221 \\ -104.6136 & 82.2742 & -31.1980 & 94.1520 & 31.6434 & -17.4743 \\ -52.6086 & 38.4252 & -26.2625 & 31.6434 & 48.9440 & -63.6683 \\ 84.6568 & -27.9424 & 45.6221 & -17.4743 & -63.6683 & 150.6507 \end{bmatrix} \tag{56}$$

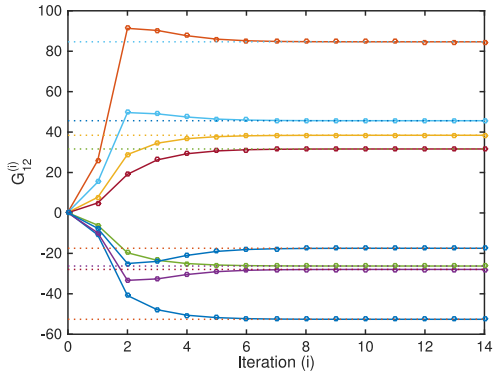

Fig. 3. Iterative $G_{12}^{(i)}$ at each iteration i.

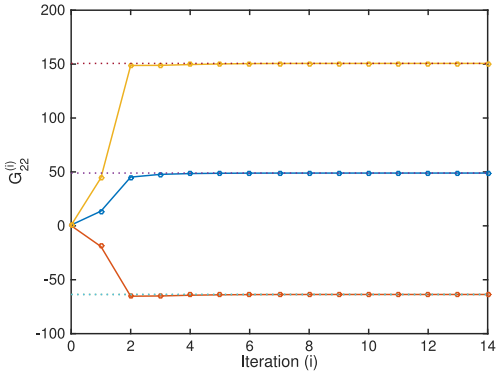

Fig. 4. Iterative $G_{22}^{(i)}$ at each iteration i.

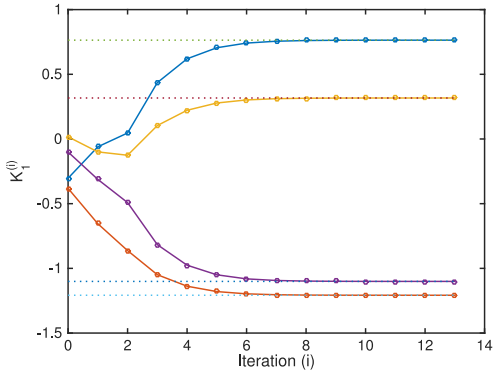

Fig. 5. Iterative control gain $K_1^{(i)}$ at each iteration i.

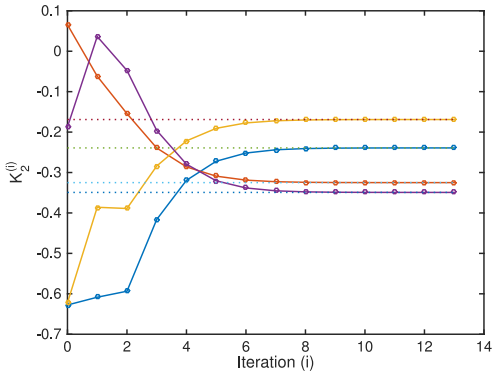

Fig. 6. Iterative control gain $K_2^{(i)}$ at each iteration i.

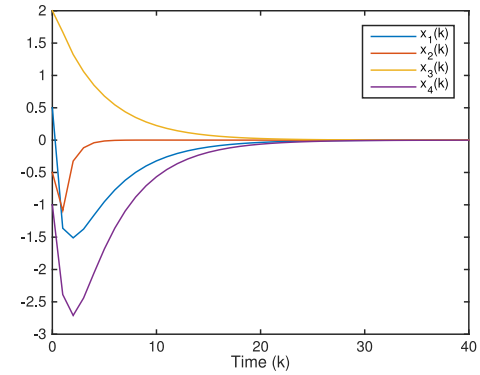

Fig. 7. Trajectories of state $x(k)$.

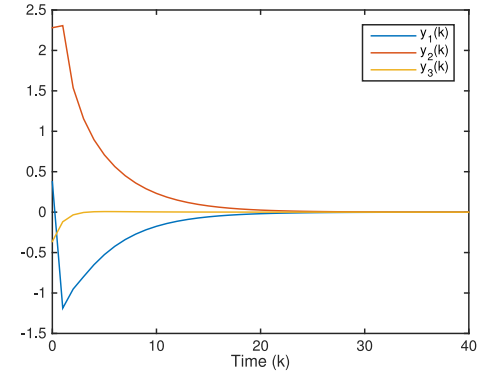

Fig. 8. Trajectories of output $y(k)$.

algebraic Riccati equation (45) is given by

$$P = \begin{bmatrix} 96.0044 & -53.3551 & 26.4689 & -76.2663 \\ -53.3551 & 48.9376 & -16.0037 & 49.7332 \\ 26.4689 & -16.0037 & 44.9563 & -18.2206 \\ -76.2663 & 49.7332 & -18.2206 & 65.4206 \end{bmatrix} \tag{54}$$

and the optimal control gain K is given by

$$K = \begin{bmatrix} 0.7638 & -1.2078 & 0.3168 & -1.1008 \\ -0.2392 & -0.3250 & -0.1689 & -0.3492 \end{bmatrix}. \tag{55}$$

By using (54), the matrix G of (47) is given in (56), as shown at the bottom of the previous page.

Letting $b = 0.4$ for (37), the adaptive QL method is applied to solve this data-based output regulation problem without using system model. The simulation results are given in Figs. 1–10. Fig. 1 gives the relationship between the iteration i and the parameter α_i, where α_i increases monotonically with respect to i. Figs. 2–4 give the iterative matrix $G^{(i)}$ at each iteration and Figs. 5 and 6 show the iterative control gain $K^{(i)}$ at each iteration, where the dotted lines represent the ideal values of the optimal matrix G in (56) and the optimal control gain K in (55), respectively. It is shown from the figures that $G^{(i)}$ and $K^{(i)}$ approach to their optimums G and K at $i = 13$th iteration. By using the converged control gain to the closed-loop system, the state trajectories $x(k)$, output trajectories $y(k)$ and control trajectories $u(k)$ are shown in Figs. 7–9, respectively. From Fig. 8, it is observed that regulation is achieved for output signal $y(k)$. Furthermore, we also investigate the influence of the parameter b in (37) on the convergence of the

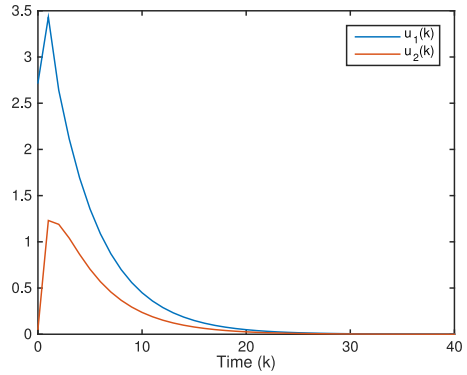

Fig. 9. Trajectories of control $u(k)$.

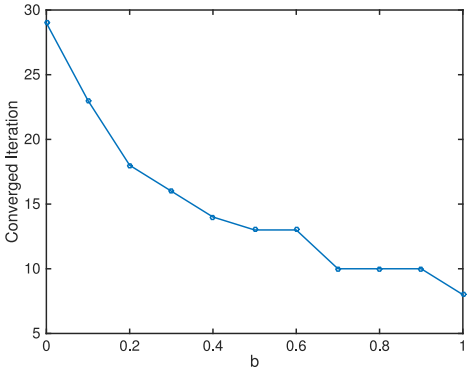

Fig. 10. Relationship between the parameter b and the converged iteration of the adaptive QL method.

adaptive QL method. By using different parameters b, Fig. 10 demonstrates the relationship between the parameter b and the converged iteration of the adaptive QL method, where the converged iteration decreases with respect to b.

VII. Conclusion

The off-policy adaptive QL method has been developed for data-based optimal output regulation problem of discrete-time nonlinear systems. First, the Q-function is introduced, which is an action-state value function. Then, an adaptive QL algorithm is proposed and its convergence theories is established. For the implementation purpose, the actor-critic NN structure is developed. To learn the weights of actor and critic NNs, the experience replay technique has been employed, where the training data is collected before learning and used during the whole learning process. Furthermore, a tuning rule is proposed to tune the adaptive parameter α_i in the adaptive QL method. Through computer simulations, it is observed that the developed adaptive QL method converges to the optimal Q-function and the optimal control policy.

References

[1] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[2] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[3] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[5] E. Even-Dar and Y. Mansour, "Learning rates for Q-learning," *J. Mach. Learn. Res.*, vol. 5, pp. 1–25, Jan. 2004.
[6] L. C. Baird, III, "Advantage updating," Wright Lab., Dayton, OH, USA, Rep. WL-TR-93-1146, 1993.
[7] M. Chen and S. S. Ge, "Adaptive neural output feedback control of uncertain nonlinear systems with unknown hysteresis using disturbance observer," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7706–7716, Dec. 2015.
[8] Q. Yang, S. Jagannathan, and Y. Sun, "Robust integral of neural network and error sign control of MIMO nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3278–3286, Dec. 2015.
[9] M. Chen, G. Tao, and B. Jiang, "Dynamic surface control using neural networks for a class of uncertain nonlinear systems with input saturation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 9, pp. 2086–2097, Sep. 2015.
[10] B. Xu and Y. Yuan, "Two performance enhanced control of flexible-link manipulator with system uncertainty and disturbances," *Sci. China Inf. Sci.*, vol. 60, no. 5, pp. 1–11, Feb. 2017.
[11] Y.-J. Liu and S. Tong, "Optimal control-based adaptive NN design for a class of nonlinear discrete-time block-triangular systems," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2670–2680, Nov. 2016.
[12] B. Xu *et al.*, "Online recorded data-based composite neural control of strict-feedback systems with application to hypersonic flight dynamics," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/document/8049319/, doi: 10.1109/TNNLS.2017.2743784.
[13] C. Wu, J. Liu, X. Jing, H. Li, and L. Wu, "Adaptive fuzzy control for nonlinear networked control systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 8, pp. 2420–2430, Aug. 2017.
[14] H. Li, J. Wang, and P. Shi, "Output-feedback based sliding mode control for fuzzy systems with actuator saturation," *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 6, pp. 1282–1293, Dec. 2016.
[15] Y.-J. Liu, M. Gong, S. Tong, C. L. P. Chen, and D.-J. Li, "Adaptive fuzzy output feedback control for a class of nonlinear systems with full state constraints," *IEEE Trans. Fuzzy Syst.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/document/8270373/, doi: 10.1109/TFUZZ.2018.2798577.
[16] Q. Yang, S. S. Ge, and Y. Sun, "Adaptive actuator fault tolerant control for uncertain nonlinear systems with multiple actuators," *Automatica*, vol. 60, pp. 92–99, Oct. 2015.
[17] Y.-J. Liu and S. Tong, "Barrier Lyapunov functions for Nussbaum gain adaptive control of full state constrained nonlinear systems," *Automatica*, vol. 76, pp. 143–152, Feb. 2017.
[18] M. Chen, S.-Y. Shao, and B. Jiang, "Adaptive neural control of uncertain nonlinear systems using disturbance observer," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3110–3123, Oct. 2017.
[19] Y.-J. Liu, S. Lu, D. Li, and S. Tong, "Adaptive controller design-based ABLF for a class of nonlinear time-varying state constraint systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1546–1553, Jul. 2017.
[20] B. Fan, Q. Yang, S. Jagannathan, and Y. Sun, "Asymptotic tracking controller design for nonlinear systems with guaranteed performance," *IEEE Trans. Cybern.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/document/7987778/, doi: 10.1109/TCYB.2017.2726039.
[21] C. Wu, J. Liu, Y. Xiong, and L. Wu, "Observer-based adaptive fault-tolerant tracking control of nonlinear nonstrict-feedback systems," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/document/7961225/, doi: 10.1109/TNNLS.2017.2712619.
[22] Y.-J. Liu *et al.*, "Adaptive control-based barrier Lyapunov functions for a class of stochastic nonlinear systems with full state constraints," *Automatica*, vol. 87, pp. 83–93, Jan. 2018.
[23] 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.
[24] B. Luo and H.-N. Wu, "Online policy iteration algorithm for optimal control of linear hyperbolic PDE systems," *J. Process Control*, vol. 22, no. 7, pp. 1161–1170, 2012.
[25] Q. Yang and S. Jagannathan, "Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 377–390, Apr. 2012.
[26] B. Luo and H.-N. Wu, "Approximate optimal control design for nonlinear one-dimensional parabolic PDE systems using empirical eigenfunctions and neural network," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 6, pp. 1538–1549, Nov. 2012.

[27] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 1, pp. 14–25, Feb. 2011.
[28] B. Luo, H.-N. Wu, and H.-X. Li, "Data-based suboptimal neuro-control design with reinforcement learning for dissipative spatially distributed processes," *Ind. Eng. Chem. Res.*, vol. 53, no. 29, pp. 8106–8119, 2014.
[29] B. Xu, C. Yang, and Z. Shi, "Reinforcement learning output feedback NN control using deterministic learning technique," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 3, pp. 635–641, Mar. 2014.
[30] B. Luo, H.-N. Wu, T. Huang, and D. Liu, "Data-based approximate policy iteration for affine nonlinear continuous-time optimal control design," *Automatica*, vol. 50, no. 12, pp. 3281–3290, 2014.
[31] A. Heydari and S. N. Balakrishnan, "Optimal switching and control of nonlinear switching systems using approximate dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1106–1117, Jun. 2014.
[32] B. Luo and H.-N. Wu, "Computationally efficient simultaneous policy update algorithm for nonlinear H_∞ state feedback control with Galerkin's method," *Int. J. Robust Nonlin. Control*, vol. 23, no. 9, pp. 991–1012, 2013.
[33] H. Modares and F. L. Lewis, "Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning," *IEEE Trans. Autom. Control*, vol. 59, no. 11, pp. 3051–3056, Nov. 2014.
[34] B. Luo, H.-N. Wu, T. Huang, and D. Liu, "Reinforcement learning solution for HJB equation arising in constrained optimal control problem," *Neural Netw.*, vol. 71, pp. 150–158, Nov. 2015.
[35] J. Y. Lee, J. B. Park, and Y. H. Choi, "On integral generalized policy iteration for continuous-time linear quadratic regulations," *Automatica*, vol. 50, no. 2, pp. 475–489, 2014.
[36] B. Luo, H.-N. Wu, and T. Huang, "Off-policy reinforcement learning for H_∞ control design," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 65–76, Jan. 2015.
[37] B. Luo, T. Huang, H.-N. Wu, and X. Yang, "Data-driven H_∞ control for nonlinear distributed parameter systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2949–2961, Nov. 2015.
[38] J. Škach, B. Kiumarsi, F. L. Lewis, and O. Straka, "Actor-critic off-policy learning for optimal control of multiple-model discrete-time systems," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 29–40, Jan. 2018.
[39] B. Luo, H.-N. Wu, and H.-X. Li, "Adaptive optimal control of highly dissipative nonlinear spatially distributed processes with neuro-dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 4, pp. 684–696, Apr. 2015.
[40] J. Y. Lee, J. B. Park, and Y. H. Choi, "Integral reinforcement learning for continuous-time input-affine nonlinear systems with simultaneous invariant explorations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 916–932, May 2015.
[41] B. Luo, D. Liu, and H.-N. Wu, "Adaptive constrained optimal control design for data-based nonlinear discrete-time systems with critic-only structure," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/document/8057603/, doi: 10.1109/TNNLS.2017.2751018.
[42] A. Sahoo and S. Jagannathan, "Stochastic optimal regulation of nonlinear networked control systems by using event-driven adaptive dynamic programming," *IEEE Trans. Cybern.*, vol. 47, no. 2, pp. 425–438, Feb. 2017.
[43] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, *Adaptive Dynamic Programming With Applications in Optimal Control*. Cham, Switzerland: Springer, 2017.
[44] B. Luo, D. Liu, H.-N. Wu, D. Wang, and F. L. Lewis, "Policy gradient adaptive dynamic programming for data-based optimal control," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3341–3354, Oct. 2017.
[45] X. Zhong and H. He, "An event-triggered ADP control approach for continuous-time system with unknown internal states," *IEEE Trans. Cybern.*, vol. 47, no. 3, pp. 683–694, Mar. 2017.
[46] B. Luo, D. Liu, T. Huang, and J. Liu, "Output tracking control based on adaptive dynamic programming with multistep policy evaluation," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/document/8122054/, doi: 10.1109/TSMC.2017.2771516.
[47] H. Zhang, H. Liang, Z. Wang, and T. Feng, "Optimal output regulation for heterogeneous multiagent systems via adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 1, pp. 18–29, Jan. 2017.
[48] D. Liu, Y. Xu, Q. Wei, and X. Liu, "Residential energy scheduling for variable weather solar energy based on adaptive dynamic programming," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 36–46, Jan. 2018.
[49] M. Mazouchi, M. B. Naghibi-Sistani, and S. K. H. Sani, "A novel distributed optimal adaptive control algorithm for nonlinear multi-agent differential graphical games," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 331–341, Jan. 2018.
[50] B. Luo, H.-N. Wu, and T. Huang, "Optimal output regulation for model-free Quanser helicopter with multistep Q-learning," *IEEE Trans. Ind. Electron.*, vol. 65, no. 6, pp. 4953–4961, Jun. 2018.
[51] 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.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/document/8310930/, doi: 10.1109/TNNLS.2018.2806347.
[52] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control," *Automatica*, vol. 43, no. 3, pp. 473–481, 2007.
[53] J.-H. Kim and F. L. Lewis, "Model-free H_∞ control design for unknown linear discrete-time systems via Q-learning with LMI," *Automatica*, vol. 46, no. 8, pp. 1320–1326, 2010.
[54] J. Y. Lee, J. B. Park, and Y. H. Choi, "Integral Q-learning and explorized policy iteration for adaptive optimal control of continuous-time linear systems," *Automatica*, vol. 48, no. 11, pp. 2850–2859, 2012.
[55] A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar, "A deterministic improved Q-learning for path planning of a mobile robot," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1141–1153, Sep. 2013.
[56] B. Kiumarsi, F. L. Lewis, H. Modares, A. Karimpour, and M.-B. Naghibi-Sistani, "Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics," *Automatica*, vol. 50, no. 4, pp. 1167–1175, 2014.
[57] M. Palanisamy, H. Modares, F. L. Lewis, and M. Aurangzeb, "Continuous-time Q-learning for infinite-horizon discounted cost linear quadratic regulator problems," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 165–176, Feb. 2015.
[58] Q. Wei, D. Liu, and G. Shi, "A novel dual iterative Q-learning method for optimal battery management in smart residential environments," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2509–2518, Apr. 2015.
[59] B. Luo, D. Liu, T. Huang, and D. Wang, "Model-free optimal tracking control via critic-only Q-learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 10, pp. 2134–2144, Oct. 2016.
[60] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," in *Proc. 29th Int. Conf. Mach. Learn.*, Edinburgh, U.K., Jun. 2012, pp. 457–464.
[61] M. Geist and B. Scherrer, "Off-policy learning with eligibility traces: A survey," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 289–333, 2014.
[62] J. T. Spooner, M. Maggiore, R. Ordonez, and K. M. Passino, *Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques*, vol. 43. New York, NY, USA: Wiley, 2004.
[63] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 293–321, May 1992.
[64] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. (Nov. 2017). *Prioritized Experience Replay*. [Online]. Available: https://arxiv.org/abs/1511.05952
[65] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems," *Automatica*, vol. 50, no. 1, pp. 193–202, 2014.
[66] D. Zhao, Q. Zhang, D. Wang, and Y. Zhu, "Experience replay for optimal control of nonzero-sum game systems with unknown dynamics," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 854–865, Mar. 2016.

Biao Luo (M'15) received the Ph.D. degree from Beihang University, Beijing, China, in 2014.

He is an Associate Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing. His current research interests include distributed parameter systems, intelligent control, reinforcement learning, deep learning, and computational intelligence.

Dr. Luo was a recipient of the Chinese Association of Automation Outstanding Ph.D. Dissertation Award in 2015. He serves as an Associate Editor of the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, the *Artificial Intelligence Review*, the *Journal of Industrial and Management Optimization*, and the *Numerical Algebra, Control and Optimization*. He is the Secretariat of Adaptive Dynamic Programming and Reinforcement Learning Technical Committee, the Chinese Association of Automation.

Yin Yang received the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong, in 2009.

He is currently an Assistant Professor with the Division of Information and Communication Technologies, College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar. He also holds adjunct positions with Qatar Computing Research Institute, Doha, and Advanced Digital Sciences Center, Singapore. He has published extensively in top venues on differentially private data publication and analysis, and on query authentication in outsourced databases. In addition, he has designed efficient query processing methods in various contexts, including data streams, relational keyword search, spatial databases, Web portals, and wireless sensor networks. He is currently researching actively on cloud-based big data analytics, with a focus on real-time stream processing. His current research interests include big data analytics, cloud computing, database security and privacy.

Derong Liu (S'91–M'94–SM'96–F'05) received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, USA, in 1994.

He became a Full Professor of electrical and computer engineering and of computer science with the University of Illinois at Chicago, Chicago, IL, USA, in 2006. He was selected for the "100 Talents Program" by the Chinese Academy of Sciences, Beijing, China, in 2008, and he served as the Associate Director of the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation from 2010 to 2015. He has published 18 books.

Dr. Liu is the Editor-in-Chief of *Artificial Intelligence Review* (Springer). He was the Editor-in-Chief of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2010 to 2015. He is a Fellow of the International Neural Network Society and the International Association of Pattern Recognition.