SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2019, Vol. 62 052204:1–052204:19
https://doi.org/10.1007/s11432-018-9602-1

Policy iteration based Q-learning for linear nonzero-sum quadratic differential games

Xinxing LI[1,2], Zhihong PENG[1,2*], Li LIANG[1,2] & Wenzhong ZHA[3]

[1]*School of Automation, Beijing Institute of Technology, Beijing* 100081*, China;*
[2]*State Key Laboratory of Intelligent Control and Decision of Complex System, Beijing* 100081*, China;*
[3]*Information Science Academy, China Electronics Technology Group Corporation, Beijing* 100086*, China*

Received 4 July 2018/Accepted 5 September 2018/Published online 2 April 2019

Abstract In this paper, a policy iteration-based Q-learning algorithm is proposed to solve infinite horizon linear nonzero-sum quadratic differential games with completely unknown dynamics. The Q-learning algorithm, which employs off-policy reinforcement learning (RL), can learn the Nash equilibrium and the corresponding value functions online, using the data sets generated by behavior policies. First, we prove equivalence between the proposed off-policy Q-learning algorithm and an offline PI algorithm by selecting specific initially admissible polices that can be learned online. Then, the convergence of the off-policy Q-learning algorithm is proved under a mild rank condition that can be easily met by injecting appropriate probing noises into behavior policies. The generated data sets can be repeatedly used during the learning process, which is computationally effective. The simulation results demonstrate the effectiveness of the proposed Q-learning algorithm.

Keywords adaptive dynamic programming, ADP, Q-learning, reinforcement learning, RL, linear nonzero-sum quadratic differential games, policy iteration, PI, off-policy

Citation Li X X, Peng Z H, Liang L, et al. Policy iteration based Q-learning for linear nonzero-sum quadratic differential games. Sci China Inf Sci, 2019, 62(5): 052204, https://doi.org/10.1007/s11432-018-9602-1

1 Introduction

Differential game theory [1] is an important tool for dealing with the coordination and conflict issues in dynamical systems with multiple decision-makers or control inputs and has been widely applied in social and behavior science, industrial processing control [2], robotic maneuvering control [3], communication [4], H-infinity robust control [5], among others. In differential games, each player tries to pursue the optimal control policy to optimize his own performance objective while taking account of the control policies of other players. The solution of a differential game is always characterized by the Nash equilibrium [1], in which no player can improve its outcome by unilaterally changing its strategy.

Theoretically, to obtain the Nash equilibrium in a nonzero-sum differential game, one should solve coupled Hamilton-Jacobi-Bellman (HJB) equations [1]. However, HJB equations are nonlinear partial differential equations, and it is difficult or even impossible to obtain their analytic solutions. In linear nonzero-sum quadratic differential games, where HJB equations reduce to coupled algebraic Riccati equations (AREs), the analytic solutions are still not readily available. Therefore, many approximate methods have been proposed to solve differential games such as Galerkin's spectral method [6], neural network approximation [7], iterative algorithm [8], successive approximation [9], algebraic methods [10, 11], and

*Corresponding author (email: peng@bit.edu.cn) partial differential inequalities [12]. However, these methods are offline methods that generate fixed control policies; thus, they are unable to meet the need of real-time implementation for controllers. Moreover, they require full knowledge of system dynamics, which makes them vulnerable to changes in system dynamics and inaccuracies in system modeling. In this study, we are interested in developing an intelligent online algorithm to seek the Nash equilibrium of linear nonzero-sum quadratic differential games with completely unknown dynamics.

The motivation of our work comes from reinforcement learning (RL) [13]. The core idea of RL is that an agent can unceasingly improve his action or policy based on the observed response from the environment, with which the agent interacts. In the control field, RL is also called approximate dynamic programming (ADP) [14] or neurodynamic programming [15]. In the seminal work of [16], an adaptive dynamic programming approach combining RL with dynamic programming (DP) was proposed to approximate the solution of a Bellman equation arising in optimal control. Different from DP, which operates backwards in time (offline) and suffers from the curse of dimensionality, ADP operates forwards in time (online), and overcomes the curse of dimensionality [17]. In the past few years, ADP has been widely used to solve optimal control problems [18–22] and differential games [23–36].

The most commonly used ADP method for differential games is policy iteration (PI) [23–37], which is based on a successive approximation technology [38]. PI involves two parts: (1) policy evaluation and (2) policy improvement. Hence, it is an iterative method for solving the coupled HJB equations (nonzero-sum game) or Hamilton-Jacobi-Isaacs (HJI) equation (zero-sum game) by constructing a sequence of admissible control policies [39] that will converge to the Nash equilibrium. An offline PI algorithm with two iteration loops was proposed for zero-sum differential games with constrained control input in [7]. The authors of [29] proposed a model-based PI algorithm to determine the mixed optimal control pairs for nonlinear zero-sum differential games. In [24], a partially model-free PI algorithm based on integral RL (IRL) was proposed to determine the Nash equilibrium solutions of linear zero-sum quadratic differential games. Two data-driven PI algorithms were proposed in [25, 26] for nonlinear zero-sum differential games with unknown dynamics. In [27], a model-free PI algorithm based on off-policy RL was developed to solve the H-infinity control of discrete-time systems.

Compared with zero-sum games [7, 23–27], nonzero-sum games are more difficult to solve, because one needs to solve multiple coupled HJB equations or AREs. A partially model-free PI algorithm was proposed to solve linear nonzero-sum quadratic differential games in [36]. In [39], a novel PI algorithm called synchronous PI was proposed to solve nonlinear nonzero-sum differential games online, where two neural network (NN) structures, i.e., critic NNs and actor NNs, are introduced to approximate the Nash equilibrium and the value functions, respectively, and the critic NNs and actor NNs are tuned synchronously in an adaptive way. Further, the authors of [29] proposed a synchronous PI algorithm that just contains critic NNs. In [30], a synchronous PI algorithm was presented to solve discrete-time nonzero-sum games. Algorithms in [28–30, 39] require full knowledge of system dynamics. To remove PI's dependence on system model, the authors of [32–34] introduced system identification technology; however, adding an identifier NN increases the computational burden and the identification error cannot be eliminated. In [35], a model-free PI algorithm based on off-policy RL was proposed to solve nonzero-sum differential games, however, the input matrices of all players were assumed to be the same. In addition, most of the PI algorithms above are based on on-policy RL [13], because the experience data used for learning is generated by the control policies that are learned about. One of the main drawbacks of on-policy RL is the insufficient exploration ability to the state space. To ensure the convergence of PI algorithms, one should inject exploratory signals or probing noises into the controllers to meet certain persistence of excitation (PE) conditions [39], which may cause deviations from the Nash equilibrium [27].

Q-learning is another powerful tool for solving optimal control problems [40–47] and Nash games [31, 48–50]. For discrete-time systems, Q-learning belongs to off-policy RL, that is, the agent can use the experience data generated by behavior policies to learn the target behaviors (i.e., the optimal control policy or Nash equilibrium); thus, Q-learning has better exploration ability to the state space than on-policy PI algorithms. In addition, Q-learning is completely model-free. Many Q-learning methods have been proposed to solve the optimal control problems of discrete-time systems [18, 40–43, 45, 46] and zero-sum games of linear discrete-time systems [49,50]. However, there are very few studies on continuous-time systems, because it is difficult to construct the Q-functions for continuous-time systems. In [44], a Q-learning method was developed for infinite-horizon discounted cost linear quadratic regulator problems. The author of [31] proposed a model-free synchronous PI algorithm for linear nonzero-sum quadratic differential games by constructing a Q-function for each player. Further, a cooperative Q-learning method was proposed to solve multi-agent differential graphical games in [47]. In fact, the Q-learning methods in [31, 44, 47] belongs to on-policy RL, which has the drawback of insufficient exploration to the state space. Thus, to ensure the convergence to the Nash equilibrium, one should inject probing noises into the controllers to meet certain PE conditions, however, the effect of the probing noises on convergence has not been analyzed in [31, 50].

Inspired by off-policy RL, we develop a novel off-policy Q-learning algorithm to solve linear nonzero-sum quadratic differential games online in this study. The main contributions of this study are threefold. First, the proposed Q-learning algorithm is completely model-free and runs online just using the data sets of the implemented behavior policies and the corresponding system state, and the data sets can be used repeatedly, which is computationally efficient. Second, as the proposed Q-learning algorithm utilizes off-policy RL, it has better exploration ability to the state space compared with the on-policy Q-learning methods in [31, 50]. Third, the convergence of the proposed Q-learning algorithm is proved under the mild rank condition which can be easily met by choosing appropriate probing noises, and the probing noises will cause no deviations from the Nash equilibrium.

The rest of this paper is organized as follows. In Section 2, some preliminaries on linear nonzero-sum quadratic differential games are presented. In Section 3, a model-based offline PI algorithm is proposed. In Section 4, a model-free Q-learning algorithm based on off-policy RL is presented. In Section 5, two simulation examples are given to demonstrate the effectiveness of the proposed Q-learning algorithm. In Section 6, conclusion is drawn, and future studies are presented.

Notations. $\mathbb{R}^n$ denotes n-dimensional Euclidean space. $\mathbb{R}^{n\times m}$ is the set of all real $n \times m$ matrices. I_n denotes the n-dimensional identity matrix. $\mathbb{Z}_+$ denotes a set of positive integers. $\otimes$ stands for the Kronecker product. $\mathrm{vec}(\cdot)$ is the vectorization operator for a matrix that converts the matrix into a column vector. For $x \in \mathbb{R}^n$, $\bar{x} := [x_1^2, x_1x_2, \ldots, x_1x_n, x_2^2, \ldots, x_2x_n, \ldots, x_{n-1}x_n, x_n^2]^{\mathrm{T}}$. For symmetric matrix $M \in \mathbb{R}^{n\times n}$, $\Theta(M)$ is a column vector that stacks the elements of a diagonal and the upper triangle part where the off-diagonal elements are taken as $M_{ij} + M_{ji}$. For a time-varying signal $\theta(t) : \mathbb{R} \to \mathbb{R}^{n\times m}$, $\theta_t := \theta(t)$ and $\theta|_{t_1}^{t_2} := \theta_{t_2} - \theta_{t_1}$. $\|A\| = (\mathrm{tr}(A^{\mathrm{T}}A))^{1/2}$ is the Frobenius norm of matrix A.

2 Problem statement

Consider the following infinite horizon linear nonzero-sum quadratic differential game:

$$\dot{x}(t) = Ax(t) + \sum_{j=1}^{N} B_j u_j(t), \quad x(0) = x_0, \quad t \geqslant 0, \tag{1}$$

where $x \in \mathbb{R}^n$ is the system state vector, $u_j \in \mathbb{R}^{m_j}$, $j \in \Gamma := \{1, 2, \ldots, N\}$ is the control input of player j, and Γ is the set of players. $A \in \mathbb{R}^{n\times n}, B_j \in \mathbb{R}^{n\times m_j}$ are the plant and input matrices that are assumed to be unknown in this study. Each player i has a cost function J_i to be optimized:

$$J_i(x(0); u_1, \ldots, u_N) = \frac{1}{2}\int_0^{\infty} \left(x^{\mathrm{T}} M_i x + \sum_{j=1}^{N} u_j^{\mathrm{T}} R_{ij} u_j \right) \mathrm{d}t, \quad \forall i \in \Gamma, \tag{2}$$

where $M_i \geqslant 0, R_{ij} > 0, \forall i, j \in \Gamma$ are user defined matrices of appropriate dimensions. For fixed stabilizing control inputs $(u_1, \ldots, u_N)$, the value function V_i that evaluates the performances of player i from time t is defined as

$$V_i(x(t); u_1, \ldots, u_N) = \frac{1}{2}\int_t^{\infty} \left(x^{\mathrm{T}} M_i x + \sum_{j=1}^{N} u_j^{\mathrm{T}} R_{ij} u_j \right) \mathrm{d}t, \quad \forall i \in \Gamma. \tag{3}$$

Definition 1 (Admissible policies [39])**.** A set of feedback control policies $u(x) = (u_1(x), \dots, u_N(x))$ is defined as admissible, if $u_i(x)$ is continuous, $u_i(0) = 0$, $u(x)$ stabilizes system (1), and cost function (2) is finite, for $\forall x_0 \in \mathbb{R}^n$.

The objective of the differential game is to find a set of admissible policies called the feedback Nash equilibrium $(u_1^*, \dots, u_N^*)$ [1], such that the following inequalities are satisfied simultaneously:

$$V_i^*(x(t); u_i^*, u_{-i}^*) \leqslant V_i(x(t); u_i, u_{-i}^*), \quad \forall u_i \in \mathbb{R}^{m_i}, \quad \forall i \in \Gamma, \tag{4}$$

where $u_{-i}^* = (u_1^*, \dots, u_{i-1}^*, u_{i+1}^*, \dots, u_N^*)$. Inequalities (4) imply that all the players will cooperate to implement the Nash equilibrium $(u_1^*, \dots, u_N^*)$, because the profit of the player who changes the control policy unilaterally will be damaged.

According to the definition of the feedback Nash equilibrium, solving the differential game above is equivalent to solving the following coupled optimal control problems:

$$\begin{aligned} &V_i^*(x(t); u_i, u_{-i}^*) := \min_{u_i} \int_t^\infty \frac{1}{2}\left(x^{\mathrm{T}} M_i x + u_i^{\mathrm{T}} R_{ii} u_i + \sum_{j\neq i} u_j^{*\mathrm{T}} R_{ij} u_j^*\right) \mathrm{d}t, \\ &\text{s.t. } \dot{x} = Ax + B_i u_i + \sum_{j\neq i} B_j u_j^*, \quad \forall i \in \Gamma. \end{aligned} \tag{5}$$

Define the following Hamiltonian functions:

$$\begin{aligned} H_i\left(x, u_i, u_{-i}^*, \frac{\partial V_i^*}{\partial x}\right) =& \left(\frac{\partial V_i^*}{\partial x}\right)^{\mathrm{T}} (Ax + B_i u_i + \sum_{j\neq i} B_j u_j^*) \\ &+ \frac{1}{2}\left(x^{\mathrm{T}} M_i x + u_i^{\mathrm{T}} R_{ii} u_i + \sum_{j\neq i} u_j^{*\mathrm{T}} R_{ij} u_j^*\right), \quad \forall x, u_i, \quad \forall i \in \Gamma. \end{aligned} \tag{6}$$

Employing Bellman's principle of optimality leads to the following HJB equations:

$$0 = \arg\min_{u_i} H_i\left(x, u_i, u_{-i}^*, \frac{\partial V_i^*}{\partial x}\right), \quad \forall i \in \Gamma. \tag{7}$$

Using stationarity condition, we can obtain

$$\frac{\partial H_i}{\partial u_i}\big|_{u_i^*} = 0 \Rightarrow u_i^* = -R_{ii}^{-1} B_i^{\mathrm{T}} \frac{\partial V_i^*}{\partial x}, \quad \forall i \in \Gamma. \tag{8}$$

Considering that system (1) is linear time-invariant, we can represent the value function $V_i^*(x), \forall i \in \Gamma$ as a quadratic form of the state, i.e., $V_i^*(x) = \frac{1}{2} x^{\mathrm{T}} P_i x$, where $P_i \in \mathbb{R}^{n\times n}$ is the positive definite value matrix. Then, Eq. (8) can be rewritten as

$$u_i^*(x) = -R_{ii}^{-1} B_i^{\mathrm{T}} P_i x, \quad \forall i \in \Gamma. \tag{9}$$

Substituting (9) into (7), the HJB equations reduce to the following coupled AREs:

$$\begin{aligned} &\left(A - \sum_{j=1}^N B_j R_{jj}^{-1} B_j^{\mathrm{T}} P_j\right)^{\mathrm{T}} P_i + P_i \left(A - \sum_{j=1}^N B_j R_{jj}^{-1} B_j^{\mathrm{T}} P_j\right) \\ &+ \sum_{j=1}^N P_j B_j R_{jj}^{-1} R_{ij} R_{jj}^{-1} B_j^{\mathrm{T}} P_j + M_i = 0, \quad \forall i \in \Gamma. \end{aligned} \tag{10}$$

According to [1], if there exist positive definite matrices $P_1, \dots, P_N$ satisfying the coupled AREs in (10), pair $(A - \sum_{j\neq i} B_j R_{jj}^{-1} B_j^{\mathrm{T}} P_i, B_i)$ is stabilizable and pair

$$\left(A - \sum_{j\neq i} B_j R_{jj}^{-1} B_j^{\mathrm{T}} P_i, M_i + \sum_{j\neq i} P_j B_j R_{jj}^{-1} R_{ij} R_{jj}^{-1} B_j^{\mathrm{T}} P_j\right)$$

is detectable for each $i \in \Gamma$. The control polices in (9) constitute a feedback Nash equilibrium and stabilize system (1).

To obtain the feedback Nash equilibrium in (9), one should solve the N coupled AREs in (10), which are nonlinear matrix equations. Moreover, Eqs. (9) and (10) contain the information of the system dynamics A and $B_1, \ldots, B_N$, which are assumed to be unknown. In Section 4, we will propose a novel model-free Q-learning method to solve the linear nonzero-sum quadratic differential game.

3 Offline PI for linear nonzero-sum quadratic differential games

Before deriving the Q-learning algorithm, we first give an offline PI algorithm that will be useful in Section 4. The offline PI algorithm is motivated by Bellman's method of successive approximation [38], which transforms the nonlinear matrix equations in (10) into linear Lyapunov equations. The detailed algorithm is as Algorithm 1.

Algorithm 1 Model-based offline PI algorithm

Step 1: (Initialization) Start with a set of initially stabilizing feedback gains $K_1^1, \ldots, K_N^1$.

Step 2: (Policy evaluation) For a given set of stabilizing feedback gains $K_1^l, \ldots, K_N^l$, solve for the positive definite matrices $P_1^l, \ldots, P_N^l$ using the following Lyapunov equations:

$$\left(A - \sum_{j=1}^{N} B_j K_j^l\right)^{\mathrm{T}} P_i^l + P_i^l \left(A - \sum_{j=1}^{N} B_j K_j^l\right) + M_i + \sum_{j=1}^{N} K_j^{l\mathrm{T}} R_{ij} K_j^l = 0, \quad \forall i \in \Gamma. \tag{11}$$

Step 3: (Policy updating) Update the feedback gains as follows:

$$K_i^{l+1} = R_{ii}^{-1} B_i^{\mathrm{T}} P_i^l, \quad \forall i \in \Gamma. \tag{12}$$

Stop if $\|K_i^l - K_i^{l-1}\| \leqslant \varepsilon, i \in \Gamma$, where ε is a small positive threshold, otherwise set $l = l + 1$ and go to Step 2.

Algorithm 1 is similar with the algorithm developed in [51], which is used to solve the infinite horizon linear quadratic regulator problems for continuous-time systems. It has been proved that the algorithm in [51] is convergent by selecting any initially stabilizing feedback gain. Compared with the algorithm in [51], the convergence analysis of Algorithm 1 is much more complicated, because there exist N coupled value functions, and the sequence of P_i^l for each $i \in \Gamma$ is not necessarily monotonous due to the coupling among matrices $P_1^l, \ldots, P_N^l$. To overcome this problem, the authors of [9] proposed an offline Lyapunov iterative algorithm to solve the infinite horizon linear nonzero-sum quadratic differential game, and proved that P_i^l will converge to P_i for each $i \in \Gamma$ by means of Lyapunov's second method and Bellman's successive approximation. Next, we will prove that Algorithm 1 is equivalent to the algorithm in [9] by choosing $K_i^1 = R_{ii}^{-1} B_i^{\mathrm{T}} P_i^0, \forall i \in \Gamma$, where $P_1^0, \ldots, P_N^0$ are the solutions of the following N AREs:

$$\begin{aligned}
&P_1^0 A + A^{\mathrm{T}} P_1^0 + M_1 - P_1^0 B_1 R_{11}^{-1} B_1^{\mathrm{T}} P_1^0 = 0,\\
&P_2^0 (A - S_1 P_1^0) + (A - S_1 P_1^0)^{\mathrm{T}} P_2^0 + (M_2 + P_1^0 Z_{21} P_1^0) - P_1^0 S_2 P_1^0 = 0,\\
&\vdots\\
&P_N^0 \left(A - \sum_{j=1}^{N-1} S_j P_j^0\right) + \left(A - \sum_{j=1}^{N-1} S_j P_j^0\right)^{\mathrm{T}} P_N^0 + \left(M_N + \sum_{j=1}^{N-1} P_j^0 Z_{Nj} P_j^0\right) - P_N^0 S_N P_N^0 = 0,
\end{aligned} \tag{13}$$

with $S_i = B_i R_{ii}^{-1} B_i^{\mathrm{T}}, Z_{ij} = B_j R_{jj}^{-1} R_{ij} R_{jj}^{-1} B_j^{\mathrm{T}}, i, j = 1, \ldots, N, i \neq j, l = 0, 1, \ldots$.

According to the optimal control theory, solving the AREs in (13) is equivalent to solving the following N infinite horizon linear quadratic regulator (IHLQR) problems successively:

$$\frac{1}{2} x_0^{\mathrm{T}} P_1^0 x_0 = \min_{u_1} \frac{1}{2} \int_0^{\infty} x^{\mathrm{T}} M_1 x + u_1^{\mathrm{T}} R_{11} u_1 \mathrm{d}t \quad \text{s.t. } \dot{x} = Ax + B_1 u_1,$$

$$\vdots$$

$$\frac{1}{2}x_0^{\mathrm{T}}P_N^0x_0 = \min_{u_N}\frac{1}{2}\int_0^{\infty} x^{\mathrm{T}}\left(M_N+\sum_{j=1}^{N-1}P_j^0Z_{Nj}P_j^0\right)x+u_N^{\mathrm{T}}R_{NN}u_N\mathrm{d}t$$
$$\text{s.t. } \dot{x}=\left(A-\sum_{j=1}^{N-1}S_jP_j^0\right)x+B_Nu_N.$$

Remark 1. In fact, the simulation study shows that Algorithm 1 will converge under any initially stabilizing feedback gains. We choose $K_i^1=R_{ii}^{-1}B_i^{\mathrm{T}}P_i^0, \forall i\in\Gamma$ for the sake of the following convergence analysis. The IHLQR problems above can be solved by using existing model-free ADP methods, such as the PI algorithms in [21, 22] or the Q-learning methods in [47], and these methods are computationally efficient. Besides, obtaining $K_1^1,\ldots,K_N^1$ sequentially lowers the difficulty of initialization compared with existing trial and error methods.

Lemma 1. If the initial feedback gains are selected as $K_i^1=R_{ii}^{-1}B_i^{\mathrm{T}}P_i^0$ for each $i\in\Gamma$, where $P_1^0,\ldots,P_N^0$ are the solutions of AREs (13), Algorithm 1 is equivalent to the algorithm proposed in [9]. Thus, $\lim_{l\to\infty}P_i^l=P_i$, $\lim_{l\to\infty}K_i^l=R_{ii}^{-1}B_i^{\mathrm{T}}P_i$ for each $i\in\Gamma$.

Proof. As triples $(A,B_i,\sqrt{M_i})$ are assumed to be stabilizable-detectable for each $i\in\Gamma$, and $M_i+\sum_{j=1}^{i-1}P_j^0Z_{ij}P_j^0$ are positive definite, there exists a unique positive definite solution to every ARE in (13). In Algorithm 1, by setting $K_i^1=R_{ii}^{-1}B_i^{\mathrm{T}}P_i^0$ and substituting $K_i^l=R_{ii}^{-1}B_i^{\mathrm{T}}P_i^{l-1}$ into (11), we can obtain

$$\begin{aligned} &P_i^l\left(A-\sum_{j=1}^{N}B_jR_{jj}^{-1}B_j^{\mathrm{T}}P_j^{l-1}\right)+\left(A-\sum_{j=1}^{N}B_jR_{jj}^{-1}B_j^{\mathrm{T}}P_j^{l-1}\right)^{\mathrm{T}}P_i^l\\ &=-M_i-\sum_{j=1}^{N}(R_{jj}^{-1}B_j^{\mathrm{T}}P_j^{l-1})^{\mathrm{T}}R_{ij}R_{jj}^{-1}B_j^{\mathrm{T}}P_j^{l-1}\\ &=-M_i-\sum_{j=1}^{N}P_j^{l-1}B_jR_{jj}^{-1}R_{ij}R_{jj}^{-1}B_j^{\mathrm{T}}P_j^{l-1}\\ &=-\left(M_i+P_i^{l-1}S_iP_i^{l-1}+\sum_{j\neq i}P_j^{l-1}Z_{ij}P_j^{l-1}\right). \end{aligned}\tag{14}$$

Note that $S_i=B_iR_{ii}^{-1}B_i^{\mathrm{T}}, \forall i\in\Gamma$. We can therefore obtain

$$\begin{aligned} &P_i^l\left(A-\sum_{j=1}^{N}B_jR_{jj}^{-1}B_j^{\mathrm{T}}P_j^{l-1}\right)+\left(A-\sum_{j=1}^{N}B_jR_{jj}^{-1}B_j^{\mathrm{T}}P_j^{l-1}\right)^{\mathrm{T}}P_i^l\\ &\qquad=\left(A-\sum_{j=1}^{N}S_jP_j^{l-1}\right)^{\mathrm{T}}P_i^l+P_i^l\left(A-\sum_{j=1}^{N}S_jP_j^{l-1}\right). \end{aligned}\tag{15}$$

Combining (14) and (15) yields

$$P_i^l\left(A-\sum_{j=1}^{N}S_jP_j^{l-1}\right)+\left(A-\sum_{j=1}^{N}S_jP_j^{l-1}\right)^{\mathrm{T}}P_i^l=-\left(M_i+P_i^{l-1}S_iP_i^{l-1}+\sum_{j\neq i}P_j^{l-1}Z_{ij}P_j^{l-1}\right).\tag{16}$$

From (16), we can know that Algorithm 1 is equivalent to the algorithm in [9]; thus, $\lim_{l\to\infty}P_i^l=P_i$, $\lim_{l\to\infty}K_i^l=R_{ii}^{-1}B_i^{\mathrm{T}}P_i$, $\forall i\in\Gamma$.

Remark 2. Compared with the coupled AREs (10), which are nonlinear matrix equations, Algorithm 1 and the algorithm proposed in [9] operate on the reduced-order linear matrix equations (Lyapunov equations), which reduce the computational difficulty. However, as these algorithms are both offline, they cannot be applied to online controllers. Besides, to run these algorithms, one needs to have accurate knowledge of system matrix A and input matrices $B_1, \ldots, B_N$, which makes these algorithms sensitive to changes in system dynamics.

4 Q-learning method based on off-policy RL

To remove the dependence on system model and increase the exploration ability to state space, in this section, we propose a model-free online Q-learning algorithm based on off-policy RL. First, we implement a set of behavior polices to system (1) to generate sufficient experience data (i.e., input-state data); then, we use the experience data to learn the target policies (i.e., the Nash equilibrium).

First, we provide some properties of the Kronecker product: (i) $\mathrm{vec}(ABC) = (C^{\mathrm{T}} \otimes A)\mathrm{vec}(B)$, (ii) $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$, (iii) $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$, where A, B, C and D are matrices of appropriate dimensions. In addition, let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of A, and $\mu_1, \ldots, \mu_m$ be the eigenvalues of B. Then the eigenvalues of $A \otimes B$ are $\lambda_i \mu_j$, $i = 1, \ldots, n, j = 1, \ldots, m$.

To derive the off-policy Q-learning algorithm, we first give the following Q-functions for each $i \in \Gamma$:

$$\begin{aligned} Q_i^l(x, u_i, u_{-i}) &:= V_i^l + H_i\left(x, u_i, u_{-i}, \frac{\partial V_i^l}{\partial x}\right) \\ &= \frac{1}{2}x^{\mathrm{T}} P_i^l x + \frac{1}{2}x^{\mathrm{T}} M_i x + \frac{1}{2}\sum_{j=1}^{N} u_j^{\mathrm{T}} R_{ij} u_j + \frac{1}{2}x^{\mathrm{T}} P_i^l \left(Ax + B_i u_i + \sum_{j \neq i} B_j u_j\right) \\ &\quad + \frac{1}{2}\left(Ax + B_i u_i + \sum_{j \neq i} B_j u_j\right)^{\mathrm{T}} P_i^l x, \quad \forall x, u_i, u_{-i},\ \forall i \in \Gamma, \end{aligned} \tag{17}$$

where $V_i^l = \frac{1}{2}x^{\mathrm{T}} P_i^l x$ is the value function obtained from (11), $u_{-i} = (u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_N)$, and $H_i(x, u_i, u_{-i}, \frac{\partial V_i^l}{\partial x})$ is the Hamiltonian function defined as

$$H_i\left(x, u_i, u_{-i}, \frac{\partial V_i^l}{\partial x}\right) = \frac{1}{2}x^{\mathrm{T}} M_i x + \frac{1}{2}\sum_{j=1}^{N} u_j^{\mathrm{T}} R_{ij} u_j + \left(\frac{\partial V_i^l}{\partial x}\right)^{\mathrm{T}}\left(Ax + \sum_{j=1}^{N} B_j u_j\right), \quad \forall x, u_i, u_{-i},\ \forall i \in \Gamma. \tag{18}$$

Let $Z^i = [x^{\mathrm{T}}, u_i^{\mathrm{T}}, u_{-i}^{\mathrm{T}}]^{\mathrm{T}}$ for each $i \in \Gamma$, with $u_{-i}^{\mathrm{T}} = [u_1^{\mathrm{T}}, \ldots, u_{i-1}^{\mathrm{T}}, u_{i+1}^{\mathrm{T}}, \ldots, u_N^{\mathrm{T}}]$. Then $Q_i^l(x, u_i, u_{-i})$ can be expressed as the following quadratic form:

$$Q_i^l(x, u_i, u_{-i}) := \frac{1}{2}Z^{i\mathrm{T}} \begin{bmatrix} S_{xx}^{i,l} & S_{xu_i}^l & S_{xu_1}^{i,l} & \cdots & S_{xu_N}^{i,l} \\ S_{u_i x}^l & S_{u_i u_i}^l & S_{u_i u_1}^l & \cdots & S_{u_i u_N}^l \\ S_{u_1 x}^{i,l} & S_{u_1 u_i}^l & S_{u_1 u_1}^{i,l} & \cdots & S_{u_1 u_N}^{i,l} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ S_{u_N x}^{i,l} & S_{u_N u_i}^l & S_{u_N u_1}^{i,l} & \cdots & S_{u_N u_N}^{i,l} \end{bmatrix} Z^i, \tag{19}$$

where $S_{xx}^{i,l} := P_i^l + P_i^l A + A^{\mathrm{T}} P_i^l + M_i$, $S_{xu_i}^l := (S_{u_i x}^l)^{\mathrm{T}} = P_i^l B_i$, $S_{xu_j}^{i,l} := (S_{u_j x}^{i,l})^{\mathrm{T}} = P_i^l B_j$, $\forall j \neq i$, $S_{u_i u_i}^l := R_{ii}$, $S_{u_k u_j}^{i,l} := 0$, $\forall j, k \neq i$, and $S_{u_i u_j}^l := (S_{u_j u_i}^l)^{\mathrm{T}} = \frac{1}{2}R_{ij}$, $\forall j \neq i$.

According to the definition of S_i^l, we can know that the information available to player i is $S_{u_i u_i}^l$ and $S_{u_i u_j}^l$, and the knowledge of A, B_i and P_i^l are embedded in $S_{xx}^{i,l}$, $S_{xu_i}^l$ and $S_{xu_j}^{i,l}$. If we can obtain the accurate Q-function Q_i^l, Eq. (12) can be rewritten as $K_i^{l+1} = (S_{u_i u_i}^l)^{-1} S_{u_i x}^l$, which requires no knowledge of system dynamics. Next, we will employ the off-policy RL technology to solve Q_i^l or S_i^l for each $i \in \Gamma$.

Implementing behavior policies $(\tilde{u}_1(t), \ldots, \tilde{u}_N(t))$ to system (1) yields

$$\dot{x} = \left(A - \sum_{j=1}^{N} B_j K_j^l\right) x(t) + \sum_{j=1}^{N} B_j (\tilde{u}_j(t) + K_j^l x(t)), \tag{20}$$

where the behavior policies are selected as $\tilde{u}_j(t) = -\tilde{K}_j x(t) + e_j(t)$ for each $j \in \Gamma$, $(\tilde{K}_1, \ldots, \tilde{K}_N)$ are feedback gains such that $A - \sum_{j=1}^{N} B_j \tilde{K}_j$ is Hurwitz and $(e_1(t), \ldots, e_N(t))$ are the injected probing noises used to increase the exploratory ability to the state space. $(-K_1^l x, \ldots, -K_N^l x)$ are the target policies to be learned about.

Substituting the values of behavior policies and the corresponding system state at time t into Q_i^l gives

$$\begin{aligned} Q_i^l(x_t, \tilde{u}_{it}, \tilde{u}_{-it}) =& \frac{1}{2} x_t^{\mathrm{T}} P_i^l x_t + \frac{1}{2} \sum_{j=1}^{N} \tilde{u}_{jt}^{\mathrm{T}} R_{ij} \tilde{u}_{jt} + \frac{1}{2} x_t^{\mathrm{T}} M_i x_t \\ &+ \frac{1}{2} x_t^{\mathrm{T}} P_i^l \left(A - \sum_{j=1}^{N} B_j K_j^l\right) x_t + \frac{1}{2} x_t^{\mathrm{T}} \left(A - \sum_{j=1}^{N} B_j K_j^l\right)^{\mathrm{T}} P_i^l x_t \\ &+ x_t^{\mathrm{T}} \sum_{j \neq i} P_i^l B_j (\tilde{u}_j(t) + K_j^l x_t) + x_t^{\mathrm{T}} P_i^l B_i (\tilde{u}_{it} + K_i^l x_t), \quad \forall i \in \Gamma. \end{aligned} \tag{21}$$

According to (11) and (12), Eq. (21) can be rewritten as

$$\begin{aligned} Q_i^l(x_t, \tilde{u}_{it}, \tilde{u}_{-it}) =& \frac{1}{2} x_t^{\mathrm{T}} P_i^l x_t + \frac{1}{2} \sum_{j=1}^{N} \tilde{u}_{jt}^{\mathrm{T}} R_{ij} \tilde{u}_{jt} - \frac{1}{2} x_t^{\mathrm{T}} \left(\sum_{j=1}^{N} K_j^{l\mathrm{T}} R_{ij} K_j^l\right) x_t \\ &+ \sum_{j \neq i} \left(\tilde{u}_j(t) + K_j^l x_t\right)^{\mathrm{T}} B_j^{\mathrm{T}} P_i^l x_t + (\tilde{u}_{it} + K_i^l x_t)^{\mathrm{T}} R_{ii} K_i^{l+1} x_t, \quad \forall i \in \Gamma. \end{aligned} \tag{22}$$

As $B_1, \ldots, B_N$ are unknown, we define the auxiliary variables T_{ij}^l as $T_{ij}^l = B_j^{\mathrm{T}} P_i^l$ for each $j \neq i$. By letting $Z^{i'} = [x^{\mathrm{T}}, \tilde{u}_i^{\mathrm{T}}, \tilde{u}_{-i}^{\mathrm{T}}]^{\mathrm{T}}$ and using the Kronecker product representations, Eq. (22) becomes

$$\begin{aligned} &\frac{1}{2} (\bar{Z}_t^{i'})^{\mathrm{T}} \Theta(S_i^l) - \frac{1}{2} \sum_{j=1}^{N} (\tilde{u}_{jt}^{\mathrm{T}} \otimes \tilde{u}_{jt}^{\mathrm{T}}) \mathrm{vec}(R_{ij}) + \frac{1}{2} (x_t^{\mathrm{T}} \otimes x_t^{\mathrm{T}}) \sum_{j=1}^{N} \mathrm{vec}(K_j^{l\mathrm{T}} R_{ij} K_j^l) \\ &- \left((x_t^{\mathrm{T}} \otimes \tilde{u}_{it}^{\mathrm{T}}) \left(I_n \otimes R_{ii}\right) + (x_t^{\mathrm{T}} \otimes x_t^{\mathrm{T}}) \times (I_n \otimes (K_i^{l\mathrm{T}} R_{ii}))\right) \times \mathrm{vec}(K_i^{l+1}) \\ &- \sum_{j \neq i} \left(x_t^{\mathrm{T}} \otimes \tilde{u}_{jt}^{\mathrm{T}} + (x_t^{\mathrm{T}} \otimes x_t^{\mathrm{T}})(I_n \otimes K_j^{l\mathrm{T}})\right) \times \mathrm{vec}(T_{ij}^l) = V_i^l(x_t), \quad \forall i \in \Gamma. \end{aligned} \tag{23}$$

As V_i^l in (23) is not available, we need to eliminate it. Note that

$$V_i^l(x_{t+\Delta t}) - V_i^l(x_t) = \int_t^{t+\Delta t} \dot{V}_i^l(x_\tau) \mathrm{d}\tau, \ \ \Delta t > 0, \tag{24}$$

along the system trajectory generated by (20). It follows that

$$\begin{aligned} \int_t^{t+\Delta t} \dot{V}_i^l(x) \mathrm{d}\tau =& \frac{1}{2} \int_t^{t+\Delta t} x^{\mathrm{T}} (P_i^l A_l + A_l^{\mathrm{T}} P_i^l) x \, \mathrm{d}\tau \\ &+ \int_t^{t+\Delta t} \left(\tilde{u}_i + K_i^l x\right)^{\mathrm{T}} R_{ii} K_i^{l+1} x \mathrm{d}\tau + \sum_{j \neq i} \int_t^{t+\Delta t} \left(\tilde{u}_j + K_j^l x\right)^{\mathrm{T}} T_{ij}^l x \mathrm{d}\tau, \end{aligned} \tag{25}$$

where $A_l = A - \sum_{j=1}^{N} B_j K_j^l$. According to (11), we have

$$\int_t^{t+\Delta t} x^{\mathrm{T}} (P_i^l A_l + A_l^{\mathrm{T}} P_i^l) x \, \mathrm{d}\tau = -\int_t^{t+\Delta t} x^{\mathrm{T}} \left(M_i + \sum_{j=1}^{N} K_j^{l\mathrm{T}} R_{ij} K_j^l\right) x \, \mathrm{d}\tau. \tag{26}$$

Using Kronecker product representations, Eqs. (24)–(26) yield

$$\begin{aligned} V_i^l(x_{t+\Delta t}) - V_i^l(x_t) =& -\frac{1}{2}\int_t^{t+\Delta t} x^{\mathrm{T}} \otimes x^{\mathrm{T}} \mathrm{d}\tau \times \mathrm{vec}\left(M_i + \sum_{j=1}^{N} K_j^{l\mathrm{T}} R_{ij} K_j^l\right) \\ &+ \left(\int_t^{t+\Delta t} x^{\mathrm{T}} \otimes \tilde{u}_i \mathrm{d}\tau (I_n \otimes R_{ii}) + \int_t^{t+\Delta t} x^{\mathrm{T}} \otimes x^{\mathrm{T}} \mathrm{d}\tau (I_n \otimes (K_i^{l\mathrm{T}} R_{ii}))\right) \\ &\times \mathrm{vec}(K_i^{l+1}) + \sum_{j\neq i}\left(\int_t^{t+\Delta t} x^{\mathrm{T}} \otimes \tilde{u}_i \mathrm{d}\tau + \int_t^{t+\Delta t} x^{\mathrm{T}} \otimes x^{\mathrm{T}} \mathrm{d}\tau (I_n \otimes K_j^{l\mathrm{T}})\right)\mathrm{vec}(T_{ij}^l). \end{aligned} \tag{27}$$

Using (27), we can eliminate V_i^l in (23), and then we obtain

$$(\bar{Z}_{t+\Delta t}^{i'} - \bar{Z}_t^{i'})^{\mathrm{T}}\Theta(S_i^l) - \Lambda_t^{i,l}\mathrm{vec}(K_i^{l+1}) + \sum_{j\neq i}\xi_t^{j,l}\mathrm{vec}(T_{ij}^l) = \Pi_t^{i,l}, \quad \forall i \in \Gamma, \tag{28}$$

where $\Lambda_t^{i,l}$, $\xi_t^{j,l}$, and $\Pi_t^{i,l}$ are defined as follows:

$$\begin{aligned} \Lambda_t^{i,l} =& -2\left(x^{\mathrm{T}} \otimes x^{\mathrm{T}}\Big|_t^{t+\Delta t} + \int_t^{t+\Delta t} x^{\mathrm{T}} \otimes x^{\mathrm{T}}\mathrm{d}\tau\right) \times \left(I_n \otimes (K_i^{l\mathrm{T}} R_{ii})\right) \\ & -2\left(x^{\mathrm{T}} \otimes \tilde{u}_i^{\mathrm{T}}\big|_t^{t+\Delta t} + \int_t^{t+\Delta t} x^{\mathrm{T}} \otimes \tilde{u}_i^{\mathrm{T}}\mathrm{d}\tau\right)(I_n \otimes R_{ii}), \\ \xi_t^{j,l} =& -2\left(x^{\mathrm{T}} \otimes x^{\mathrm{T}}\Big|_t^{t+\Delta t} + \int_t^{t+\Delta t} x^{\mathrm{T}} \otimes x^{\mathrm{T}}\mathrm{d}\tau\right) \\ & \times (I_n \otimes K_j^{l\mathrm{T}}) - 2\left(x^{\mathrm{T}} \otimes \tilde{u}_j^{\mathrm{T}}\big|_t^{t+\Delta t} + \int_t^{t+\Delta t} x^{\mathrm{T}} \otimes \tilde{u}_j^{\mathrm{T}}\mathrm{d}\tau\right), \quad \forall j \neq i, \end{aligned}$$

and

$$\begin{aligned} \Pi_t^{i,l} =& \sum_{j=1}^{N}\left(\tilde{u}_j^{\mathrm{T}} \otimes \tilde{u}_j^{\mathrm{T}}\big|_t^{t+\Delta t}\right)\mathrm{vec}(R_{ij}) + \left(x^{\mathrm{T}} \otimes x^{\mathrm{T}}\big|_t^{t+\Delta t}\right)\sum_{j=1}^{N}\mathrm{vec}(K_j^{l\mathrm{T}} R_{ij} K_j^l) \\ & - \left(\int_t^{t+\Delta t} x^{\mathrm{T}} \otimes x^{\mathrm{T}}\mathrm{d}\tau\right)\mathrm{vec}\left(M_i + \sum_{j=1}^{N} K_j^{l\mathrm{T}} R_{ij} K_j^l\right). \end{aligned}$$

It can be seen that Eq. (28) is a system of linear equations with respect to $\Theta(S_i^l), \mathrm{vec}(K_i^{l+1})$ and T_{ij}^l for each $i \in \Gamma, j \neq i$. In each linear equation, there exist

$$n\sum_{j=1}^{N} m_j + \frac{(1 + n + \sum_{j=1}^{N} m_j)(n + \sum_{j=1}^{N} m_j)}{2}$$

unknown parameters. Thus, to obtain $\Theta(S_i^l), \mathrm{vec}(K_i^{l+1})$ and T_{ij}^l, we need to construct at least

$$n\sum_{j=1}^{N} m_j + \frac{(1 + n + \sum_{j=1}^{N} m_j)(n + \sum_{j=1}^{N} m_j)}{2}$$

linear equations for each $i \in \Gamma$. This can be achieved by collecting data sets along the system trajectory generated by (20). Define the sampling time as $t_k := t_0 + k\Delta t, k \in \mathbb{Z}_+$, where $\Delta t > 0$ is the sampling interval.

Define the following data sets for each $i \in \Gamma$:

$$\Phi_i^l = \begin{bmatrix} (\bar{Z}_{t_1}^{i'} - \bar{Z}_{t_0}^{i'})^{\mathrm{T}} & \Lambda_{t_1}^{i,l} & \xi_{t_1}^{i,l} \\ \vdots & \vdots & \vdots \\ (\bar{Z}_{t_k}^{i'} - \bar{Z}_{t_{k-1}}^{i'})^{\mathrm{T}} & \Lambda_{t_k}^{i,l} & \xi_{t_k}^{i,l} \\ \vdots & \vdots & \vdots \\ (\bar{Z}_{t_s}^{i'} - \bar{Z}_{t_{s-1}}^{i'})^{\mathrm{T}} & \Lambda_{t_s}^{i,l} & \xi_{t_s}^{i,l} \end{bmatrix}, \quad \Pi_i^l = \begin{bmatrix} \Pi_{t_1}^{i,l} \\ \vdots \\ \Pi_{t_k}^{i,l} \\ \vdots \\ \Pi_{t_s}^{i,l} \end{bmatrix},$$

where $\xi_{t_k}^{i,l} = [\xi_{t_k}^{1,l}, \ldots, \xi_{t_k}^{i-1,l}, \xi_{t_k}^{i+1,l}, \ldots, \xi_{t_k}^{N,l}]$.

Thus, we can obtain the following N group of systems of linear equations:

$$\Phi_i^l \Big[(\Theta(S_i^l))^{\mathrm{T}}, \mathrm{vec}^{\mathrm{T}}(K_i^{l+1}), \mathrm{vec}^{\mathrm{T}}(T_i^l) \Big]^{\mathrm{T}} = \Pi_i^l, \quad \forall i \in \Gamma, \tag{29}$$

for each $i \in \Gamma$, data set $\mathrm{vec}(T_i^l) = [\mathrm{vec}^{\mathrm{T}}(T_{i1}^l), \ldots, \mathrm{vec}^{\mathrm{T}}(T_{ii-1}^l), \mathrm{vec}^{\mathrm{T}}(T_{ii+1}^l), \ldots, \mathrm{vec}^{\mathrm{T}}(T_{iN}^l)]^{\mathrm{T}}$. According to the definition of Φ_i^l and Π_i^l, we can obtain $K_i^{l+1}, \forall i \in \Gamma$ without using any knowledge of A and $B_1, \ldots, B_N$, if Eq. (29) is solvable. To ensure that each equation in (29) has a unique solution, the total sampling number s should be larger than

$$n \sum_{j=1}^{N} m_j + \frac{(1 + n + \sum_{j=1}^{N} m_j)(n + \sum_{j=1}^{N} m_j)}{2}$$

and Φ_i^l should have full column rank for each $i \in \Gamma$ and $l \in \mathbb{Z}_+$. If Φ_i^l has a full column rank, $\Phi_i^{l\mathrm{T}} \Phi_i^l$ is invertible and $\Theta(S_i^l), \mathrm{vec}(K_i^{l+1})$ and T_{ij}^l can be obtained directly as follows:

$$\begin{pmatrix} \Theta(S_i^l) \\ \mathrm{vec}(K_i^{l+1}) \\ \mathrm{vec}(T_i^l) \end{pmatrix} = \left(\Phi_i^{l\mathrm{T}} \Phi_i^l \right)^{-1} (\Phi_i^l)^{\mathrm{T}} \Pi_i^l, \quad \forall i \in \Gamma. \tag{30}$$

Next, we will give a sufficient condition under which Φ_i^l has full column rank for each $i \in \Gamma$ and $l \in \mathbb{Z}_+$.

Lemma 2. If there exist a positive integer s_0 and probing noises $(e_1(t), \ldots, e_N(t))$ such that the following condition:

$$\mathrm{rank}(H) = n \sum_{j=1}^{N} m_j + \frac{(1 + n + \sum_{j=1}^{N} m_j)(n + \sum_{j=1}^{N} m_j)}{2}$$

is satisfied for all positive integers $s \geqslant s_0$, Φ_i^l has full column rank for each $i \in \Gamma$ and $l \in \mathbb{Z}_+$, where $H = [H_{\bar{Z}'}, H_{xe_1}, \ldots, H_{xe_N}]$, $Z' = [x^{\mathrm{T}}, \tilde{u}_1^{\mathrm{T}}, \ldots, \tilde{u}_k^{\mathrm{T}}, \ldots, \tilde{u}_N^{\mathrm{T}}]^{\mathrm{T}}$, $H_{\bar{Z}'} = [\bar{Z}'_{t_1} - \bar{Z}'_{t_0}, \ldots, \bar{Z}'_{t_s} - \bar{Z}'_{t_{s-1}}]^{\mathrm{T}}$ and $H_{xe_i} = [\int_{t_0}^{t_1} x \otimes e_i \mathrm{d}\tau, \ldots, \int_{t_{s-1}}^{t_s} x \otimes e_i \mathrm{d}\tau]^{\mathrm{T}}, \forall i \in \Gamma$.

Proof. First, we define the following data sets:

$$H_{xx} = \begin{bmatrix} x^{\mathrm{T}} \otimes x^{\mathrm{T}} \big|_{t_0}^{t_1} \\ \vdots \\ x^{\mathrm{T}} \otimes x^{\mathrm{T}} \big|_{t_{k-1}}^{t_k} \\ \vdots \\ x^{\mathrm{T}} \otimes x^{\mathrm{T}} \big|_{t_{s-1}}^{t_s} \end{bmatrix}, \quad H'_{xx} = \begin{bmatrix} \int_{t_0}^{t_1} x^{\mathrm{T}} \otimes x^{\mathrm{T}} \mathrm{d}\tau \\ \vdots \\ \int_{t_{k-1}}^{t_k} x^{\mathrm{T}} \otimes x^{\mathrm{T}} \mathrm{d}\tau \\ \vdots \\ \int_{t_{s-1}}^{t_s} x^{\mathrm{T}} \otimes x^{\mathrm{T}} \mathrm{d}\tau \end{bmatrix},$$

$$H_{x\tilde{u}_i}=\begin{bmatrix} x^{\mathrm{T}}\otimes\tilde{u}_i^{\mathrm{T}}\big|_{t_0}^{t_1}\\ \vdots\\ x^{\mathrm{T}}\otimes\tilde{u}_i^{\mathrm{T}}\Big|_{t_{k-1}}^{t_k}\\ \vdots\\ x^{\mathrm{T}}\otimes\tilde{u}_i^{\mathrm{T}}\Big|_{t_{s-1}}^{t_s}\end{bmatrix},\quad\forall i\in\Gamma,\quad H_{\tilde{u}_m\tilde{u}_n}=\begin{bmatrix}\tilde{u}_m^{\mathrm{T}}\otimes\tilde{u}_n^{\mathrm{T}}\big|_{t_0}^{t_1}\\ \vdots\\ \tilde{u}_m^{\mathrm{T}}\otimes\tilde{u}_n^{\mathrm{T}}\Big|_{t_{k-1}}^{t_k}\\ \vdots\\ \tilde{u}_m^{\mathrm{T}}\otimes\tilde{u}_n^{\mathrm{T}}\Big|_{t_{s-1}}^{t_s}\end{bmatrix},\quad\forall m,n\in\Gamma.$$

Define matrices $\tilde{H}$ and $\hat{\Phi}_i^l$ as $\tilde{H}=[H_{xx},H_{x\tilde{u}_1},\ldots,H_{x\tilde{u}_N},H_{\tilde{u}_1\tilde{u}_1},\ldots,H_{\tilde{u}_N\tilde{u}_N},H_{xe_1},\ldots,H_{xe_N}]$ and $\hat{\Phi}_i^l=[H_{xx},H_{x\tilde{u}_1},\ldots,H_{x\tilde{u}_N},H_{\tilde{u}_1\tilde{u}_1},\ldots,H_{\tilde{u}_N\tilde{u}_N},\Lambda_i^l,\xi_i^l]$, where $\Lambda_i^l=[(\Lambda_{t_1}^{i,l})^{\mathrm{T}},\ldots,(\Lambda_{t_s}^{i,l})^{\mathrm{T}}]^{\mathrm{T}}$ and $\xi_i^l=[(\xi_{t_1}^{i,l})^{\mathrm{T}},\ldots,(\xi_{t_s}^{i,l})^{\mathrm{T}}]^{\mathrm{T}}$. Using the definitions of $\tilde{H}$ and $\hat{\Phi}_i^l$, we can obtain that $\operatorname{rank}(H)=\operatorname{rank}(\tilde{H})$ and $\operatorname{rank}(\Phi_i^l)=\operatorname{rank}(\hat{\Phi}_i^l)$. Thus, we just need to prove that $\operatorname{rank}(\tilde{H})=\operatorname{rank}(\hat{\Phi}_i^l)$ for each $i\in\Gamma$ and $l\in\mathbb{Z}_+$.

First, we give one useful property of the rank of partitioned matrix. Let $S=[S_1,\ldots,S_i,\ldots,S_M]$ be a partitioned matrix with $S_i\in\mathbb{R}^{p\times q_i}$. Then the rank of S will not change under the following two operations: (i) post-multiplying an arbitrary block S_i by a invertible matrix U_i with $U_i\in\mathbb{R}^{q_i\times q_i}$ and (ii) adding an arbitrary block S_j post-multiplied by a matrix V_j to block S_i with $V_j\in\mathbb{R}^{q_j\times q_i}$, and V_j is not necessarily invertible. Define

$$\tilde{\Lambda}_i^l=-2\begin{bmatrix}\int_{t_0}^{t_1}x^{\mathrm{T}}\otimes x^{\mathrm{T}}\mathrm{d}\tau(I_n\otimes(K_i^{l\mathrm{T}}R_{ii}))\\ \quad+\int_{t_0}^{t_1}x^{\mathrm{T}}\otimes\tilde{u}_i^{\mathrm{T}}\mathrm{d}\tau(I_n\otimes R_{ii})\\ \vdots\\ \int_{t_{s-1}}^{t_s}x^{\mathrm{T}}\otimes x^{\mathrm{T}}\mathrm{d}\tau(I_n\otimes(K_i^{l\mathrm{T}}R_{ii}))\\ \quad+\int_{t_{s-1}}^{t_s}x^{\mathrm{T}}\otimes\tilde{u}_i^{\mathrm{T}}\mathrm{d}\tau(I_n\otimes R_{ii})\end{bmatrix},\quad\forall i\in\Gamma$$

and

$$\tilde{\xi}_j^l=-2\begin{bmatrix}\int_{t_0}^{t_1}x^{\mathrm{T}}\otimes x^{\mathrm{T}}\mathrm{d}\tau(I_n\otimes K_j^{l\mathrm{T}})\\ \quad+\int_{t_0}^{t_1}x^{\mathrm{T}}\otimes\tilde{u}_j^{\mathrm{T}}\mathrm{d}\tau\\ \vdots\\ \int_{t_{s-1}}^{t_s}x^{\mathrm{T}}\otimes x^{\mathrm{T}}\mathrm{d}\tau(I_n\otimes K_j^{l\mathrm{T}})\\ \quad+\int_{t_{s-1}}^{t_s}x^{\mathrm{T}}\otimes\tilde{u}_j^{\mathrm{T}}\mathrm{d}\tau\end{bmatrix},\quad\forall j\neq i.$$

Let $\tilde{\xi}_j^l=[\tilde{\xi}_1^l,\ldots,\tilde{\xi}_{i-1}^l,\tilde{\xi}_{i+1}^l,\ldots,\tilde{\xi}_N^l]$. We can obtain

$$\tilde{\Lambda}_i^l=\Lambda_i^l+H_{x\tilde{u}_i}(2I_s\otimes(I_n\otimes R_{ii}))+H_{xx}(2I_s\otimes(I_n\otimes(K_i^{l\mathrm{T}}R_{ii})))\tag{31}$$

and

$$\tilde{\xi}_j^l=\xi_j^l+H_{x\tilde{u}_j}(2I_s\otimes(I_n\otimes I_{nm_j}))+H_{xx}(2I_s\otimes(I_n\otimes K_j^{l\mathrm{T}})).\tag{32}$$

Let $\tilde{\Phi}_i^l=[H_{xx},H_{x\tilde{u}_1},\ldots,H_{x\tilde{u}_N},H_{\tilde{u}_1\tilde{u}_1},\ldots,H_{\tilde{u}_N\tilde{u}_N},\tilde{\Lambda}_i^l,\tilde{\xi}_i^l]$. Eqs. (31) and (32) imply that $\tilde{\Phi}_i^l$ can be obtained from $\hat{\Phi}_i^l$ under operation (i), which means that $\operatorname{rank}(\hat{\Phi}_i^l)=\operatorname{rank}(\tilde{\Phi}_i^l)$.

Observe that

$$x^{\mathrm{T}}\otimes x^{\mathrm{T}}\Big|_{t_{k-1}}^{t_k}=\int_{t_{k-1}}^{t_k}\dot{x}^{\mathrm{T}}\otimes x^{\mathrm{T}}\mathrm{d}\tau+\int_{t_{k-1}}^{t_k}x^{\mathrm{T}}\otimes\dot{x}^{\mathrm{T}}\mathrm{d}\tau.\tag{33}$$

Along with (20), Eq. (33) becomes

$$\begin{aligned}x^{\mathrm{T}}\otimes x^{\mathrm{T}}\Big|_{t_{k-1}}^{t_k}=&\int_{t_{k-1}}^{t_k}(x^{\mathrm{T}}\hat{A}^{\mathrm{T}})\otimes x^{\mathrm{T}}\mathrm{d}\tau+\int_{t_{k-1}}^{t_k}x^{\mathrm{T}}\otimes(x^{\mathrm{T}}\hat{A}^{\mathrm{T}})\mathrm{d}\tau\\&+\sum_{j=1}^{N}\int_{t_{k-1}}^{t_k}(e_j^{\mathrm{T}}B_j^{\mathrm{T}})\otimes x^{\mathrm{T}}\mathrm{d}\tau+\sum_{j=1}^{N}\int_{t_{k-1}}^{t_k}x^{\mathrm{T}}\otimes(e_j^{\mathrm{T}}B_j^{\mathrm{T}})\mathrm{d}\tau,\end{aligned}\tag{34}$$

where $\hat{A} = A - \sum_{j=1}^{N} B_j\tilde{K}_j$. Considering that $e_j^{\mathrm{T}} \otimes x^{\mathrm{T}}$ and $x^{\mathrm{T}} \otimes e_j^{\mathrm{T}}$ are row vectors, there exists an invertible matrix $W^j \in \mathbb{R}^{nm_j\times nm_j}$ for each $j \in \Gamma$ such that $\int_{t_{k-1}}^{t_k} e_j^{\mathrm{T}} \otimes x^{\mathrm{T}}\mathrm{d}\tau = \int_{t_{k-1}}^{t_k} x^{\mathrm{T}}\otimes e_j^{\mathrm{T}}\mathrm{d}\tau W^j$. Thus, Eq. (34) can be rewritten as

$$x^{\mathrm{T}} \otimes x^{\mathrm{T}} \Big|_{t_{k-1}}^{t_k} = \int_{t_{k-1}}^{t_k} x^{\mathrm{T}} \otimes x^{\mathrm{T}}\mathrm{d}\tau\tilde{A} + \sum_{j=1}^{N}\int_{t_{k-1}}^{t_k} x^{\mathrm{T}}\otimes e_j^{\mathrm{T}}\mathrm{d}\tau\,\tilde{B}_j, \tag{35}$$

where $\tilde{A} = \hat{A}^{\mathrm{T}} \otimes I_n + I_n \otimes \hat{A}^{\mathrm{T}}$ and $\tilde{B}_j = W^j(B_j^{\mathrm{T}} \otimes I_n) + I_n \otimes B_j^{\mathrm{T}}$. According to (35), it follows that

$$H_{xx} = H'_{xx}(I_s \otimes \tilde{A}) + \sum_{j=1}^{N} H_{ex_j}(I_s \otimes \tilde{B}_j). \tag{36}$$

Next, we will prove that matrix $\tilde{A}$ is invertible by proof of contradiction. Assuming that $\tilde{A}$ is not invertible, then $(\hat{A}^{\mathrm{T}} \otimes I_n + I_n \otimes \hat{A}^{\mathrm{T}})X = 0$ has a nontrivial solution $X_1 \neq 0$. Thus, $(I_{n^2} + (\hat{A}^{\mathrm{T}} \otimes I_n)^{-1}(\hat{A}^{-\mathrm{T}} \otimes \hat{A}^{\mathrm{T}}))X_1 = 0$. Using the Kronecker product, we can obtain $(I_{n^2} + \hat{A}^{-\mathrm{T}} \otimes \hat{A}^{\mathrm{T}})X_1 = 0$. It can be seen that none of the eigenvalues of matrix $I_{n^2} + \hat{A}^{-\mathrm{T}} \otimes \hat{A}^{\mathrm{T}}$ is zero, which means that $I_{n^2} + \hat{A}^{-\mathrm{T}} \otimes \hat{A}^{\mathrm{T}}$ is invertible and $X_1 = 0$. This contradicts with the assumption, so $\tilde{A}$ is invertible. As $\tilde{A}$ is invertible, Eq. (36) yields

$$H'_{xx} = \left(H_{xx} - \sum_{j=1}^{N} H_{ex_j}(I_s \otimes \tilde{B}_j)\right)(I_s \otimes \tilde{A})^{-1}. \tag{37}$$

Let $\tilde{H}' = [H'_{xx}, H_{x\tilde{u}_1}, \ldots, H_{x\tilde{u}_N}, H_{\tilde{u}_1\tilde{u}_1}, \ldots, H_{\tilde{u}_N\tilde{u}_N}, H_{xe_1}, \ldots, H_{xe_N}]$. Eq. (37) implies that $\tilde{H}'$ can be obtained from $\tilde{H}$ under operations (i) and (ii), which means $\mathrm{rank}(\tilde{H}') = \mathrm{rank}(\tilde{H})$. As $\tilde{u}_j(t) = -\tilde{K}_jx(t) + e_j(t), \forall j \in \Gamma$, we have

$$\int_{t_{k-1}}^{t_k} x^{\mathrm{T}}\otimes\tilde{u}_j^{\mathrm{T}}\mathrm{d}\tau = \int_{t_{k-1}}^{t_k} x^{\mathrm{T}}\otimes e_j^{\mathrm{T}}\mathrm{d}\tau - \int_{t_{k-1}}^{t_k} x^{\mathrm{T}}\otimes x^{\mathrm{T}}\mathrm{d}\tau(I_n \otimes \tilde{K}_j^{\mathrm{T}}). \tag{38}$$

According to the definitions of $\tilde{\Lambda}_i^l$ and $\tilde{\xi}_j^l$, we can obtain

$$\tilde{\Lambda}_i^l = H_{xe_i}(-2I_s \otimes (I_n \otimes R_{ii})) + H'_{xx}(2I_s \otimes (I_n \otimes (\tilde{K}_i^{\mathrm{T}} - K_i^{l\mathrm{T}}R_{ii}))), \tag{39}$$

and

$$\tilde{\xi}_j^l = H_{xe_j}(-2I_s \otimes I_n) + H'_{xx}(2I_s \otimes (I_n \otimes (\tilde{K}_j^{\mathrm{T}} - K_j^{l\mathrm{T}}))). \tag{40}$$

Obviously, both $-2I_s \otimes (I_n \otimes R_{ii})$ and $-2I_s \otimes I_n$ are invertible. Thus, Eqs. (38) and (39) imply that $\tilde{\Phi}_i^l$ can be obtained from $\tilde{H}'$ under operations (i) and (ii); thus $\mathrm{rank}(\tilde{\Phi}_i^l) = \mathrm{rank}(\tilde{H}')$. We have proved that $\mathrm{rank}(\Phi_i^l) = \mathrm{rank}(\hat{\Phi}_i^l) = \mathrm{rank}(\tilde{\Phi}_i^l)$ and $\mathrm{rank}(H) = \mathrm{rank}(\tilde{H}) = \mathrm{rank}(\tilde{H}')$; thus $\mathrm{rank}(\Phi_i^l) = \mathrm{rank}(H) = n\sum_{j=1}^{N} m_j + \frac{(1+n+\sum_{j=1}^{N} m_j)(n+\sum_{j=1}^{N} m_j)}{2}$. The proof is completed.

Remark 3. According to Lemma 2, we can conclude that, once the rank condition in Lemma 2 is satisfied, the exact Q-function Q_i^l or S_i^l can be obtained from (30) for each $i \in \Gamma$ at each iteration. The rank condition in Lemma 2 is more relaxed than the PE condition. In addition, the rank condition can be satisfied by choosing appropriate probing noises (e.g., harmonic signals containing sufficient frequencies or random signals).

Now, we present the complete off-policy Q-learning algorithm.

Remark 4. Though the initially admissible feedback gains $K_i^1 = R_{ii}^{-1}B_i^{\mathrm{T}}P_i^0, \forall i \in \Gamma$ in Step 2 contain information on $B_1, \ldots, B_N$, they can be obtained using existing methods [21, 22], without using any information on $B_1, \ldots, B_N$. Thus, Algorithm 2 is completely model-free (data-driven) and robust to the inaccuracy in system modeling. Note that once the rank condition is met, the data of the state and behavior polices can be used repeatedly, which is more computationally efficient than the existing Q-learning methods [41, 44, 48, 49], for which new experience data must be regenerated in each new iteration.

Algorithm 2 Model-free off-policy Q-learning algorithm

```
Step1: (Data collection) Apply control inputs (i.e., behavior policies) $\tilde{u}_j(t) = -\tilde{K}_j x(t) + e_j(t), j \in \Gamma$ to system (1) and store the values of the state and behavior policies from $t_0$ to $t_s$.
Step2: (Initialization) Solve the $N$ decoupled IHLQR problems corresponding to AREs (13) to obtain the initially admissible feedback gains $K_i^1 = R_{ii}^{-1} B_i^{\mathrm{T}} P_i^0$ for each $i \in \Gamma$.
Step3: (Policy updating) Given a set of admissible feedback gains $K_1^l, \ldots, K_N^l$, solve for $S_i^l$, $K_i^{l+1}$ and $T_{ij}^l$ for each $i \in \Gamma$, $j \neq i$ using (30).
Stop if $\|K_i^l - K_i^{l-1}\| \leqslant \varepsilon$ for all $i \in \Gamma$, where $\varepsilon$ is a small positive threshold; otherwise set $l = l + 1$ and go to Step 3.
```

Theorem 1. Let the rank condition in Lemma 2 be satisfied. Then Algorithm 2 is equivalent to Algorithm 1; thus the target policies $(-K_i^l x, \ldots, -K_N^l x)$ obtained from Algorithm 2 will converge to the Nash equilibrium, that is, $\lim_{l\to\infty} K_i^l = R_{ii}^{-1} B_i^{\mathrm{T}} P_i, \forall i \in \Gamma$.

Proof. Let $K_1^l, \ldots, K_N^l$ be the admissible feedback gains at the lth iteration in Algorithm 1 and $K_i^1 = R_{ii}^{-1} B_i^{\mathrm{T}} P_i^0$ for each $i \in \Gamma$. As $A - \sum_{i=1}^N B_i K_i^l$ is Hurwitz, $P_1^l, \ldots, P_N^l$ are the unique solutions of the Lyapunov equations (11), which means $K_1^{l+1}, \ldots, K_N^{l+1}$ are uniquely determined by $K_i^{l+1} = R_{ii}^{-1} B_i^{\mathrm{T}} P_i^l$ for each $i \in \Gamma$. From the definition of Q_i^l and the derivation of Algorithm 2, we can know that K_i^l, P_i^l, and K_i^{l+1} satisfy (30) for each $i \in \Gamma$ (K_i^l is embedded in Φ_i^l and Π_i^l, and P_i^l is embedded in S_i^l). If the rank condition in Lemma 2 is satisfied, Eq. (30) has a unique solution $[(\psi_i^l)^{\mathrm{T}}, (\chi_i^l)^{\mathrm{T}}, (\zeta_i^l)^{\mathrm{T}}]^{\mathrm{T}}$ for each $i \in \Gamma$, where

$$\psi_i^l \in \mathbb{R}^{\frac{(1+n+\sum_{j=1}^N m_j)(n+\sum_{j=1}^N m_j)}{2}}, \quad \chi_i^l \in \mathbb{R}^{nm_i}$$

and $\zeta_i^l \in \mathbb{R}^{n\sum_{j\neq i} m_j}$. Thus, we have $\chi_i^l = \mathrm{vec}(K_i^{l+1})$. Using the result in Lemma 1, we can obtain $\lim_{l\to\infty} K_i^l = R_{ii}^{-1} B_i^{\mathrm{T}} P_i, \forall i \in \Gamma$.

Employing Algorithm 2, we can obtain the Nash equilibrium $(u_1^*, \ldots, u_N^*)$, but the corresponding value matrices $(P_1, \ldots, P_N)$ are still unknown. Next, we will turn to obtaining the value matrices $P_i, \forall i \in \Gamma$.

Define the following optimal Q-functions:

$$Q_i^*(x, u_i^*, u_{-i}^*) := V_i^* + H_i\left(x, u_i^*, u_{-i}^*, \frac{\partial V_i^*}{\partial x}\right), \quad \forall x, \forall i \in \Gamma, \tag{41}$$

where the Hamiltonian $H_i(x, u_i^*, u_{-i}^*, \frac{\partial V_i^*}{\partial x})$ is defined as

$$H_i(x, u_i^*, u_{-i}^*, \frac{\partial V_i^*}{\partial x}) := \frac{1}{2} x^{\mathrm{T}} M_i x + \frac{1}{2}\sum_{j=1}^N u_j^{*\mathrm{T}} R_{ij} u_j^* + \left(\frac{\partial V_i^*}{\partial x}\right)^{\mathrm{T}} \left(Ax + \sum_{j=1}^N B_j u_j^*\right), \quad \forall x, \forall i \in \Gamma. \tag{42}$$

Obviously, $Q_i^*(x, u_i^*, u_{-i}^*)$ can be obtained after the Nash equilibrium is obtained. By substituting $(u_1^*, \ldots, u_N^*)$ into (42) and using AREs (10), we have $H_i(x, u_i^*, u_{-i}^*, \frac{\partial V_i^*}{\partial x}) = 0, \forall x, \forall i \in \Gamma$. Thus,

$$Q_i^*(x, u_i^*, u_{-i}^*) = V_i^* = \frac{1}{2} x^{\mathrm{T}} P_i x, \quad \forall x, \forall i \in \Gamma. \tag{43}$$

Let $\{x(t), t \geqslant 0\}$ be the system trajectory generated by (20); then we can obtain

$$Q_i^*(x_{t_k}, u_{it_k}^*, u_{-it_k}^*) - Q_i^*(x_{t_{k-1}}, u_{it_{k-1}}^*, u_{-it_{k-1}}^*) = \frac{1}{2}\bar{x}_{t_k}^{\mathrm{T}} \Theta(P_i) - \frac{1}{2}\bar{x}_{t_{k-1}}^{\mathrm{T}} \Theta(P_i), \quad \forall i \in \Gamma. \tag{44}$$

Let $\tilde{Q}_i^* = [Q_i^*(x_{t_1}, u_{it_1}^*, u_{-it_1}^*) - Q_i^*(x_{t_0}, u_{it_0}^*, u_{-it_0}^*), \ldots, Q_i^*(x_{t_s}, u_{it_s}^*, u_{-it_s}^*) - Q_i^*(x_{t_{s-1}}, u_{it_{s-1}}^*, u_{-it_{s-1}}^*)]^{\mathrm{T}}$ and $\tilde{X} = [\bar{x}_{t_1} - \bar{x}_{t_0}, \ldots, \bar{x}_{t_s} - \bar{x}_{t_{s-1}}]^{\mathrm{T}}$. If the rank condition in Lemma 2 is satisfied, we can easily verify that matrix $\tilde{X}$ has full column rank. Thus, we have

$$\Theta(P_i) = 2(\tilde{X}^{\mathrm{T}}\tilde{X})^{-1}\tilde{X}^{\mathrm{T}}\tilde{Q}_i^*, \quad \forall i \in \Gamma. \tag{45}$$

Now, the value matrices $(P_1, \ldots, P_N)$ corresponding to the Nash equilibrium $(u_1^*, \ldots, u_N^*)$ are obtained.

5 Simulation study

In this section, two simulation examples are provided to show the effectiveness of Algorithm 2.

Example 1 (Government debt stabilization game [11])**.** In this game, it is assumed that government debt accumulation, $\dot{d}$, is the sum of interest payments on government debt, $\mathrm{rd}(t)$, and primary fiscal deficits, $f(t)$, minus the seignorage, $m(t)$:

$$\dot{d}(t) = \mathrm{rd}(t) + f(t) - m(t),\ \ d(0) = d_0. \tag{46}$$

The objective of the fiscal authority is to minimize the sum of time profiles of the primary fiscal deficit, base-money growth, and government debt:

$$J_1 = \int_0^{\infty} \left\{ \big(f(t) - \bar{f}\big)^2 + \eta(m(t) - \bar{m})^2 \ +\lambda\big(d(t) - \bar{d}\big)^2 \mathrm{e}^{-\delta t} \right\} \mathrm{d}t, \tag{47}$$

whereas the monetary authorities set the growth of base money so as to minimize the loss function:

$$J_2 = \int_0^{\infty} \left\{ (m(t) - \bar{m})^2 \ +\kappa\big(d(t) - \bar{d}\big)^2 \mathrm{e}^{-\delta t} \right\} \mathrm{d}t, \tag{48}$$

where $\bar{f}$, $\bar{m}$ and $\bar{d}$ are assumed to be fixed targets that are given a priori.

Introducing the auxiliary variables $x_1(t) := \big(d(t) - \bar{d}\big)\,\mathrm{e}^{-\frac{1}{2}\delta t}$, $x_2(t) := \big(r\bar{d} + \bar{f} - \bar{m}\big)\,\mathrm{e}^{-\frac{1}{2}\delta t}$, $u_1(t) := \big(f(t) - \bar{f}\big)\,\mathrm{e}^{-\frac{1}{2}\delta t}$ and $u_2(t) := (m(t) - \bar{m})\,\mathrm{e}^{-\frac{1}{2}\delta t}$, the author of [11] shows that the game above can be rewritten as the following differential game:

$$\dot{x} = Ax + B_1u_1 + B_2u_2, \tag{49}$$

with $A = \left[\begin{smallmatrix} r - \frac{1}{2}\delta & 1 \\ 0 & -\frac{1}{2}\delta \end{smallmatrix}\right]$, $B_1 = \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$, $B_2 = \left[\begin{smallmatrix} -1 \\ 0 \end{smallmatrix}\right]$, $\delta = 0.04$, $r = 0.06$ and $x_0 = [\,-1.4\ 0.2\,]^{\mathrm{T}}$.

The cost functions (47) and (48) become

$$J_1 = \int_0^{\infty} \lambda x_1^2(t) + u_1^2(t) + \eta u_2^2(t)\mathrm{d}t, \tag{50}$$

and

$$J_2 = \int_0^{\infty} \kappa x_1^2(t) + \eta u_2^2(t)\mathrm{d}t, \tag{51}$$

with $\eta = 0.02$, $\lambda = \kappa = 1$, $M_1 = \mathrm{diag}(\lambda, 0)$, $M_2 = \mathrm{diag}(\kappa, 0)$, $R_{11} = 1$, $R_{12} = \eta$, $R_{22} = \kappa$ and $R_{21} = 0$.

Employing the algorithm proposed in [9], we obtain the solutions of the AREs:

$$P_1 = \begin{bmatrix} 0.6295 & 0.4082 \\ 0.4082 & 10.8105 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 0.5713 & 0.2863 \\ 0.2863 & 6.4225 \end{bmatrix}.$$

The Nash equilibrium feedback gains are

$$K_1^* = \begin{bmatrix} 0.6295 & 0.4082 \end{bmatrix}, \quad K_2^* = \begin{bmatrix} -0.5713 & -0.2863 \end{bmatrix}.$$

Now we implement Algorithm 2 to solve this differential game. We can obtain the initial admissible feedback gains: $K_1^1 = [\,1.0408\ 1.0196\,]$, $K_2^1 = [\,-0.4140\ 0.0057\,]$. The behavior policies are selected as $\tilde{u}_1(t) = -K_1^1x(t) + e_1(t)$ and $\tilde{u}_2(t) = -K_2^1x(t) + e_2(t)$, where $e_1(t) = 6\sum_{n=1}^{40}\sin(nt)$ and $e_2(t) = \sin(3t) + 6\sin(4t^3) + \sin(6t) + \sin(10t)$ are the injected probing noises to maintain the rank condition in Lemma 2. During the learning process, data of the state and behavior policies are collected over intervals of 0.1 s, i.e., $\Delta t = 0.1$, and the number of sampling intervals is $s = 15$. After twelve iterations, the feedback gains K_1^l, K_2^l converge with

$$K_1^{13} = \begin{bmatrix} 0.6283 & 0.4101 \end{bmatrix}, \quad K_2^{13} = \begin{bmatrix} -0.5730 & -0.2851 \end{bmatrix}.$$

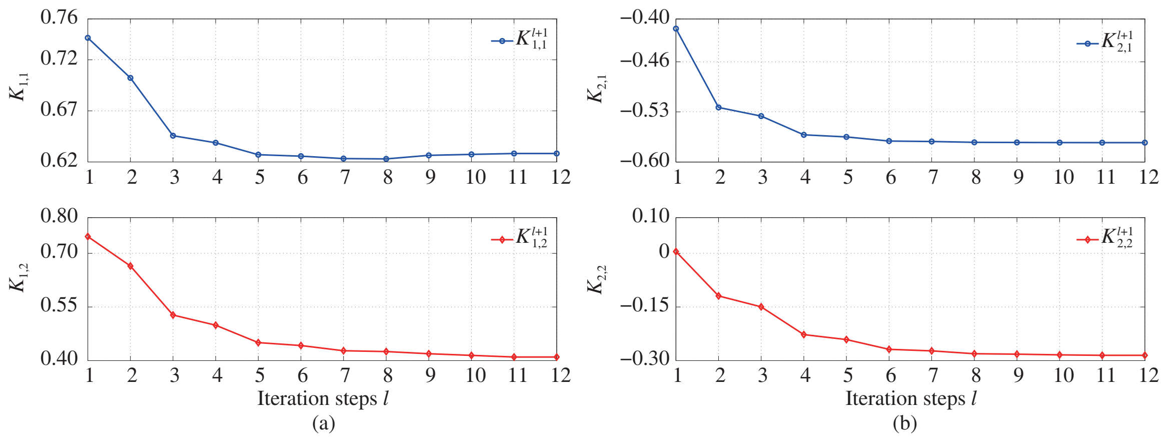

Figure 1 (Color online) Convergence of feedback gains for both players. (a) The convergence of feedback gains K_1^l to K_1^*, where $K_1^{l+1} = [K_{1,1}^{l+1}, K_{1,2}^{l+1}]$; (b) the convergence of feedback gains K_2^l to K_2^*, where $K_2^{l+1} = [K_{2,1}^{l+1}, K_{2,2}^{l+1}]$.

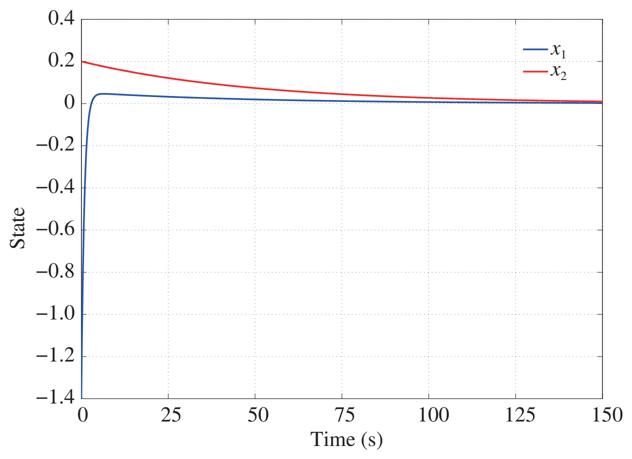

Figure 2 (Color online) State evolution of system (49) by implementing $u_1(t) = -K_1^{13}x(t)$ and $u_2(t) = -K_2^{13}x(t)$.

By substituting $u_1 = -K_1^{13}x$ and $u_2 = -K_2^{13}x$ into (45), we can obtain the value matrices

$$P_1^{13} = \begin{bmatrix} 0.6289 & 0.4105 \\ 0.4105 & 10.8216 \end{bmatrix}, \quad P_2^{13} = \begin{bmatrix} 0.5704 & 0.2893 \\ 0.2893 & 6.3941 \end{bmatrix}.$$

The feedback gain errors $\|K_1^{13} - K_1^*\| = 0.0022$, $\|K_2^{13} - K_2^*\| = 0.0021$, and the value matrix errors $\|P_1^{13} - P_1\| = 0.0116$, $\|P_2^{13} - P_2\| = 0.0287$.

Figure 1 shows that feedback gains K_1^l and K_2^l will converge to the Nash equilibrium feedback gains K_1^* and K_2^*. Figure 2 shows that $u_1(t) = -K_1^{13}x(t)$ and $u_2(t) = -K_2^{13}x(t)$ stabilize the system (49).

Example 2. Consider a more complicated linear nonzero-sum quadratic differential game with two players [8]:

$$\dot{x} = Ax + B_1u_1 + B_2u_2, \tag{52}$$

with

$$A = \begin{bmatrix} -0.0366 & 0.0271 & 0.0188 & -0.4555 \\ 0.0482 & -1.01 & 0.0024 & -4.0208 \\ 0.1002 & 0.2855 & -0.707 & 1.3229 \\ 0 & 0 & 1 & 0 \end{bmatrix}, B_1 = \begin{bmatrix} 0.4422 \\ 3.0447 \\ -5.52 \\ 0 \end{bmatrix}, B_2 = \begin{bmatrix} 0.1761 \\ -7.5922 \\ 4.99 \\ 0 \end{bmatrix}, x(0) = \begin{bmatrix} 10 \\ 20 \\ 15 \\ 5 \end{bmatrix}.$$

The cost functions are

$$J_1 = \frac{1}{2}\int_0^\infty (x^{\mathrm{T}} M_1 x + u_1^{\mathrm{T}} R_{11} u_1 + u_2^{\mathrm{T}} R_{12} u_2)\mathrm{d}t \tag{53}$$

and

$$J_2 = \frac{1}{2}\int_0^\infty (x^{\mathrm{T}} M_2 x + u_2^{\mathrm{T}} R_{22} u_2 + u_1^{\mathrm{T}} R_{21} u_1)\mathrm{d}t, \tag{54}$$

where $M_1 = \mathrm{diag}(3.5, 2, 4, 5)$, $M_2 = \mathrm{diag}(1.5, 6, 3, 1)$, $R_{11} = 1$, $R_{12} = 0.25$, $R_{21} = 0.6$, and $R_{22} = 2$.

Employing the algorithm proposed in [9], we obtain the solutions of the AREs:

$$P_1 = \begin{bmatrix} 7.6566 & 0.6438 & 0.6398 & -3.0811 \\ 0.6438 & 0.2878 & 0.2855 & -0.0945 \\ 0.6398 & 0.2855 & 0.5620 & 0.2270 \\ -3.0811 & -0.0945 & 0.2270 & 6.6987 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 3.4579 & 0.1568 & 0.2047 & -1.8480 \\ 0.1568 & 0.6235 & 0.2889 & -0.0711 \\ 0.2047 & 0.2889 & 0.4020 & 0.0729 \\ -1.8480 & -0.0711 & 0.0729 & 3.7850 \end{bmatrix}.$$

The Nash equilibrium feedback gains are

$$K_1^* = \begin{bmatrix} 1.8151 & -0.4150 & -1.9501 & -2.9041 \end{bmatrix},$$

$$K_2^* = \begin{bmatrix} 0.2200 & -1.6323 & -0.0772 & 0.2891 \end{bmatrix}.$$

Now we implement Algorithm 2 to solve the differential game above. We can obtain the initially admissible feedback gains: $K_1^1 = [1.5621\ 0.6371\ -1.8146\ -3.6510]$ and $K_2^1 = [0.1446\ -1.5486\ -0.0681\ 0.4211]$. The behavior policies are chosen as $\tilde{u}_1(t) = -K_1^1 x(t) + e_1(t)$ and $\tilde{u}_2(t) = -K_2^1 x(t) + e_2(t)$, where $e_1(t) = 6\sum_{n=1}^{50}\sin(nt)$ and $e_2(t) = \sin(3t) + 6\sin(7t) + \cos(5t) + 6\cos(11t)$ are probing noises to meet the rank condition in Lemma 2. During the learning process, data of the state and behavior policies are collected over intervals of $0.1s$, i.e., $\Delta t = 0.1$ and the number of sampling intervals is $s = 30$. After seven iterations, the feedback gains K_1^l and K_2^l converge with

$$K_1^8 = \begin{bmatrix} 1.8164 & -0.4130 & -1.9611 & -2.9090 \end{bmatrix},$$

$$K_2^8 = \begin{bmatrix} 0.2185 & -1.6303 & -0.0780 & 0.2885 \end{bmatrix}.$$

By substituting $u_1 = -K_1^8 x$ and $u_2 = -K_2^8 x$ into (45), we can obtain the value matrices

$$P_1^8 = \begin{bmatrix} 7.6662 & 0.6433 & 0.6382 & -3.0911 \\ 0.6433 & 0.2884 & 0.2862 & -0.0926 \\ 0.6382 & 0.2862 & 0.5648 & 0.2314 \\ -3.0911 & -0.0926 & 0.2314 & 6.7120 \end{bmatrix}, \quad P_2^8 = \begin{bmatrix} 3.4574 & 0.1560 & 0.2032 & -1.8552 \\ 0.1560 & 0.6238 & 0.2904 & -0.0681 \\ 0.2032 & 0.2904 & 0.4032 & 0.0757 \\ -1.8552 & -0.0681 & 0.0757 & 3.8063 \end{bmatrix}.$$

The feedback gain errors are $\|K_1^8 - K_1^*\| = 0.0123$ and $\|K_2^8 - K_2^*\| = 0.0027$. The value matrix errors are $\|P_1^8 - P_1\| = 0.0206$ and $\|P_2^8 - P_2\| = 0.0276$.

Figure 3 shows that feedback gains K_1^l and K_2^l will converge to the Nash equilibrium feedback gains K_1^* and K_2^*. Figure 4 shows that $u_1(t) = -K_1^8 x(t)$ and $u_2(t) = -K_2^8 x(t)$ stabilize the system (52).

6 Conclusion

This paper proposes an online Q-learning algorithm based on off-policy RL to solve infinite horizon linear nonzero-sum quadratic differential games with completely unknown dynamics. By selecting a set of appropriate initially admissible control policies, we prove the equivalence between the Q-learning algorithm and an offline PI algorithm. A rank condition on probing noises is established to ensure the

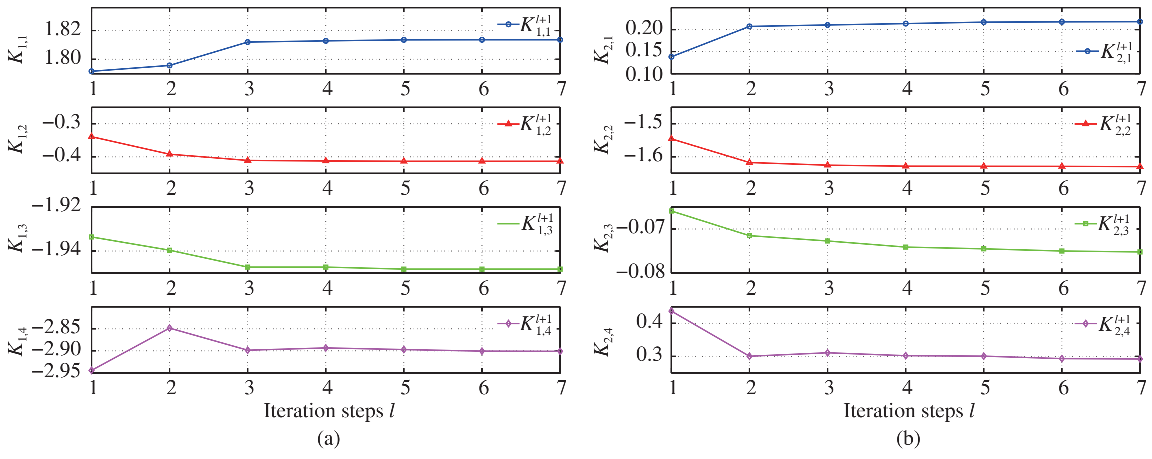

Figure 3 (Color online) Convergence of feedback gains for both players. (a) The convergence of feedback gains K_1^l to K_1^*, where $K_1^{l+1} = [K_{1,1}^{l+1}, K_{1,2}^{l+1}, K_{1,3}^{l+1}, K_{1,4}^{l+1}]$; (b) the convergence of feedback gains K_2^l to K_2^*, where $K_2^{l+1} = [K_{2,1}^{l+1}, K_{2,2}^{l+1}, K_{2,3}^{l+1}, K_{2,4}^{l+1}]$.

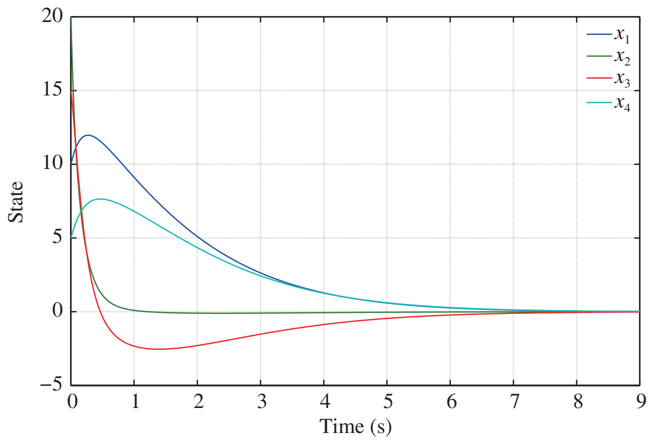

Figure 4 (Color online) State evolution of system (52) by implementing $u_1(t) = -K_1^8 x(t)$ and $u_2(t) = -K_2^8 x(t)$.

convergence of the proposed Q-learning method. The simulation study shows the effectiveness of the proposed Q-learning algorithm. In the future studies, we will focus on developing off-policy Q-learning methods to solve nonlinear nonzero-sum Nash differential games with unknown dynamics.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61203078) and the Key Project of Shenzhen Robotics Research Center NSFC (Grant No. U1613225).

References

1 Basar T, Olsder G J. Dynamic Noncooperative Game Theory (Classics in Applied Mathematics). 2nd ed. Philadelphia: SIAM, 1999

2 Falugi P, Kountouriotis P A, Vinter R B. Differential games controllers that confine a system to a safe region in the state space, with applications to surge tank control. IEEE Trans Automat Contr, 2012, 57: 2778–2788

3 Zha W Z, Chen J, Peng Z H, et al. Construction of barrier in a fishing game with point capture. IEEE Trans Cybern, 2017, 47: 1409–1422

4 Lin F H, Liu Q, Zhou X W, et al. Towards green for relay in InterPlaNetary Internet based on differential game model. Sci China Inf Sci, 2014, 57: 042306

5 Luo B, Wu H N, Huang T. Off-policy reinforcement learning for H_∞ control design. IEEE Trans Cybern, 2015, 45: 65–76

6 Bea R W. Successive Galerkin approximation algorithms for nonlinear optimal and robust control. Int J Control, 1998, 71: 717–743

7 Abu-Khalaf M, Lewis F L, Huang J. Neurodynamic programming and zero-sum games for constrained control systems. IEEE Trans Neural Netw, 2008, 19: 1243–1252

8 Freiling G, Jank G, Abou-Kandil H. On global existence of solutions to coupled matrix Riccati equations in closed-loop

Nash games. IEEE Trans Automat Contr, 1996, 41: 264–269
9 Li T Y, Gajic Z. Lyapunov iterations for solving coupled algebraic riccati equations of nash differential games and algebraic riccati equations of zero-sum game. In: New Trends in Dynamic Games and Applications. Boston: Birkhäuser, 1995. 333–351
10 Possieri C, Sassano M. An algebraic geometry approach for the computation of all linear feedback Nash equilibria in LQ differential games. In: Proceedings of the 54th IEEE Conference on Decision and Control, Osaka, 2015. 5197–5202
11 Engwerda J C. LQ Dynamic Optimization and Differential Games. New York: Wiley, 2005
12 Mylvaganam T, Sassano M, Astolfi A. Constructive ϵ-Nash equilibria for nonzero-sum differential games. IEEE Trans Automat Contr, 2015, 60: 950–965
13 Sutton R S, Barto A G. Reinforcement Learning: an Introduction. Cambridge: MIT Press, 1998
14 Werbos P J. Approximate dynamic programming for real-time control and neural modeling. In: Handbook of Intelligent Control. New York: Van Nostrand, 1992
15 Bertsekas D P, Tsitsiklis J N. Neuro-Dynamic Programming. Belmont: Athena Scientific, 1996
16 Werbos P J. The elements of intelligence. Cybernetica, 1968, 11: 131
17 Doya K. Reinforcement learning in continuous time and space. Neural Computation, 2000, 12: 219–245
18 Wei Q L, Lewis F L, Sun Q Y, et al. Discrete-time deterministic Q-learning: a novel convergence analysis. IEEE Trans Cyber, 2016, 47: 1–14
19 Wang D, Mu C X. Developing nonlinear adaptive optimal regulators through an improved neural learning mechanism. Sci China Inf Sci, 2017, 60: 058201
20 Vrabie D, Pastravanu O, Abu-Khalaf M, et al. Adaptive optimal control for continuous-time linear systems based on policy iteration. Automatica, 2009, 45: 477–484
21 Jiang Y, Jiang Z P. Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. Automatica, 2012, 48: 2699–2704
22 Luo B, Wu H N, Huang T W, et al. Data-based approximate policy iteration for affine nonlinear continuous-time optimal control design. Automatica, 2014, 50: 3281–3290
23 Zhang H G, Wei Q L, Liu D R. An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games. Automatica, 2011, 47: 207–214
24 Vrabie D, Lewis F L. Adaptive dynamic programming for online solution of a zero-sum differential game. J Control Theor Appl, 2011, 9: 353–360
25 Zhu Y H, Zhao D B, Li X G. Iterative adaptive dynamic programming for solving unknown nonlinear zero-sum game based on online data. IEEE Trans Neural Netw Learn Syst, 2017, 28: 714–725
26 Modares H, Lewis F L, Jiang Z P. H_∞ tracking control of completely unknown continuous-time systems via off-policy reinforcement learning. IEEE Trans Neural Netw Learn Syst, 2015, 26: 2550–2562
27 Kiumarsi B, Lewis F L, Jiang Z P. H_∞ control of linear discrete-time systems: off-policy reinforcement learning. Automatica, 2017, 78: 144–152
28 Vamvoudakis K G, Lewis F L, Hudas G R. Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality. Automatica, 2012, 48: 1598–1611
29 Zhang H G, Cui L L, Luo Y H. Near-optimal control for nonzero-sum differential games of continuous-time nonlinear systems using single-network ADP. IEEE Trans Cybern, 2013, 43: 206–216
30 Zhang H G, Jiang H, Luo C M, et al. Discrete-time nonzero-sum games for multiplayer using policy-iteration-based adaptive dynamic programming algorithms. IEEE Trans Cybern, 2017, 47: 3331–3340
31 Vamvoudakis K G. Non-zero sum Nash Q-learning for unknown deterministic continuous-time linear systems. Automatica, 2015, 61: 274–281
32 Zhao D B, Zhang Q C, Wang D, et al. Experience replay for optimal control of nonzero-sum game systems with unknown dynamics. IEEE Trans Cybern, 2016, 46: 854–865
33 Johnson M, Kamalapurkar R, Bhasin S, et al. Approximate N-player nonzero-sum game solution for an uncertain continuous nonlinear system. IEEE Trans Neural Netw Learn Syst, 2015, 26: 1645–1658
34 Liu D R, Li H L, Wang D. Online synchronous approximate optimal learning algorithm for multi-player non-zero-sum games with unknown dynamics. IEEE Trans Syst Man Cybern Syst, 2014, 44: 1015–1027
35 Song R Z, Lewis F L, Wei Q L. Off-policy integral reinforcement learning method to solve nonlinear continuous-time multiplayer nonzero-sum games. IEEE Trans Neural Netw Learn Syst, 2017, 28: 704–713
36 Vrabie D, Lewis F L. Integral reinforcement learning for online computation of feedback Nash strategies of nonzero-sum differential games. In: Proceedings of the 49th IEEE Conference on Decision and Control, Atlanta, 2010: 3066–3071
37 Vamvoudakis K G, Modares H, Kiumarsi B, et al. Game theory-based control system algorithms with real-time reinforcement learning: how to solve multiplayer games online. IEEE Control Syst, 2017, 37: 33–52
38 Leake R J, Liu R W. Construction of suboptimal control sequences. SIAM J Control, 1967, 5: 54–63
39 Vamvoudakis K G, Lewis F L. Multi-player non-zero-sum games: online adaptive learning solution of coupled Hamilton-Jacobi equations. Automatica, 2011, 47: 1556–1569
40 Watkins C, Dayan P. Q-Learning. Mach Learn, 1992, 8: 279–292
41 Bradtke S J, Ydstie B E, Barto A G. Adaptive linear quadratic control using policy iteration. In: Proceedings of American Control Conference, Baltimore, 1994. 3475–3479
42 Chen C L, Dong D Y, Li H X, et al. Hybrid MDP based integrated hierarchical Q-learning. Sci China Inf Sci, 2011, 54: 2279–2294
43 Wei Q L, Liu D R. A novel policy iteration based deterministic Q-learning for discrete-time nonlinear systems. Sci

China Inf Sci, 2015, 58: 122203

44 Palanisamy M, Modares H, Lewis F L, et al. Continuous-time Q-learning for infinite-horizon discounted cost linear quadratic regulator problems. IEEE Trans Cybern, 2015, 45: 165–176

45 Yan P F, Wang D, Li H L, et al. Error bound analysis of Q-function for discounted optimal control problems with policy iteration. IEEE Trans Syst Man Cybern Syst, 2017, 47: 1207–1216

46 Luo B, Liu D R, Wu H N, et al. Policy gradient adaptive dynamic programming for data-based optimal control. IEEE Trans Cybern, 2017, 47: 3341–3354

47 Vamvoudakis K G. Q-learning for continuous-time linear systems: a model-free infinite horizon optimal control approach. Syst Control Lett, 2017, 100: 14–20

48 Vamvoudakis K G, Hespanha J P. Cooperative Q-learning for rejection of persistent adversarial inputs in networked linear quadratic systems. IEEE Trans Automat Contr, 2018, 63: 1018–1031

49 Rizvi S A A, Lin Z L. Output feedback Q-learning for discrete-time linear zero-sum games with application to the H-infinity control. Automatica, 2018, 95: 213–221

50 Li J N, Chai T Y, Lewis F L, et al. Off-policy Q-learning: set-point design for optimizing dual-rate rougher flotation operational processes. IEEE Trans Ind Electron, 2018, 65: 4092–4102

51 Kleinman D. On an iterative technique for Riccati equation computations. IEEE Trans Automat Contr, 1968, 13: 114–115