Information Sciences 222 (2013) 472–485

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Simultaneous policy update algorithms for learning the solution of linear continuous-time H_∞ state feedback control

Huai-Ning Wu *, Biao Luo

Science and Technology on Aircraft Control Laboratory, School of Automation Science and Electrical Engineering, Beihang University (Beijing University of Aeronautics and Astronautics), Beijing 100191, PR China

ARTICLE INFO

Article history:
Received 20 November 2011
Received in revised form 4 August 2012
Accepted 12 August 2012
Available online 22 August 2012

Keywords:
Simultaneous policy update algorithm
H_∞ state feedback control
Algebra Riccati equation
Lyapunov equation
Offline
Online

ABSTRACT

It is well known that the H_∞ state feedback control problem can be viewed as a two-player zero-sum game and reduced to find a solution of the algebra Riccati equation (ARE). In this paper, we propose a simultaneous policy update algorithm (SPUA) for solving the ARE, and develop offline and online versions. The offline SPUA is a model-based approach, which obtains the solution of the ARE by solving a sequence of Lyapunov equations (LEs). Its convergence is established rigorously by constructing a Newton's sequence for the fixed point equation. The online SPUA is a partially model-free approach, which takes advantage of the thought of reinforcement learning (RL) to learn the solution of the ARE online without requiring the internal system dynamics, wherein both players update their action policies simultaneously. The convergence of the online SPUA is proved by demonstrating that it is mathematically equivalent to the offline SPUA. Finally, by conducting comparative simulation studies on an F-16 aircraft plant and a power system, the results show that both the offline SPUA and the online SPUA can find the solution of the ARE, and achieve much better convergence than the existing methods.

1. Introduction

Over the past decades, reinforcement learning (RL) and approximate dynamic programming (ADP) have appeared to be promising methodologies for solving H_2 optimal control problems and H_∞ control problems [17,28,31]. RL is a kind of machine learning method [12,13,22,28,32], which refers to an actor or agent that interacts with its environment and aims to learn the optimal actions, or control policies, by observing their responses from the environment. ADP [18,31] uses RL schemes for approximately solving the dynamic programming problem forward-in-time, thus, it affords a methodology for learning the feedback control laws online in real time based on system performance without necessarily knowing the system dynamics. This overcomes the *curse of dimensionality* [18] of dynamic programming, which is an offline technique that requires a backward-in-time solution procedure.

In recent years, the bulk of researches on feedback control with ADP and RL are mainly focused on optimal control problems (e.g., see [17,31] and references therein). However, for practical systems, the control performance is often affected by the presence of unknown disturbances. In this situation, an H_∞ controller can be designed to ensure that a ratio between the energy of output and disturbance is less than a prespecified level. The synthesis of the H_∞ controller relies on solving the Hamilton–Jacobi–Isaacs (HJI) equation for nonlinear systems and the algebraic Riccati equation (ARE) for linear systems [27]. Compared with the H_2 ARE (for optimal control) where the quadratic term is negative semidefinite, the H_∞ ARE is a

* Corresponding author.
E-mail address: whn@buaa.edu.cn (H.-N. Wu).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.08.012

nonlinear matrix equation with indefinite quadratic term [16]. Kleinman [15] developed a well known iterative method for the solution of the H_2 ARE by solving a sequence of linear Lyapunov equations (LEs), but it cannot be directly used for solving the H_∞ ARE. In [16], the H_∞ ARE was successively approximated by a sequence of H_2 AREs. Based on this work, Vrabie and Lewis [29] used value iteration (VI) to solve each H_2 ARE, and obtained the solution of the H_∞ ARE in an online fashion without requiring complete knowledge of the system dynamics. The method in [16] was also extended to solve the AREs that arise from linear quadratic (LQ) stochastic zero-game problems [10], the stochastic H_∞ control problems [7,8], and the periodic H_∞ control problem [9]. It is worth mentioning that the method in [16] is similar with the iterative algorithm in [27] where the HJI equation was successively approximated by a series of Hamilton–Jacobi–Bellman (HJB) equations. Thereafter in [2], the works in [3,4,20] were extended to solve the HJB equations in [27], where each HJB equation was further successively approximated with a sequence of generalized HJB (GHJB) equations. This method was realized in [24] by employing the so-called synchronous zero-sum game policy iteration, which is the extension of the method in [25] for optimal control. In [1,33], by using neural networks as value function approximators, ADP techniques were also developed to find solution of HJI equation subject to input constraints. Notice that the mentioned methods are in fact the extensions of Kleinman's algorithm [15] to two-player zero-sum games, where two iterative loops are needed, such as in [2], inner loop with iterations on the control and outer loop on disturbance. This may lead to redundant iterations and thus waste of sources and result in low efficiency.

To overcome this deficiency, we propose a simultaneous policy update algorithm (SPUA) for finding the solution of the H_∞ ARE, where two players update their policies simultaneously. Both offline and online algorithms based on the SPUA are developed and their convergences are established rigorously. Unlike the solution procedure in [29], where the H_∞ ARE was initially converted to a sequence of H_2 AREs based on [16] and VI was then used to solve each H_2 ARE online, the proposed SPUA transforms directly the H_∞ ARE into a sequence of LEs, and thus avoids the solutions of H_2 AREs.

The remainder of this paper is organized as follows. In Section 2, we give the problem description and briefly present the existing methods. The SPUA is proposed for the solution of the H_∞ ARE and related issues are considered in Section 3. Comparative simulation studies on an F-16 aircraft and a power system with offline and online algorithms are provided in Section 4. Finally, a brief conclusion is drawn in Section 5.

Notations: $\mathbb{R}, \mathbb{R}^n$ and $\mathbb{R}^{n\times m}$ are the set of real numbers, the n-dimensional Euclidean space and the set of all real $n \times m$ matrices, respectively. $\|\cdot\|$ denotes the vector norm or matrix norm in $\mathbb{R}^n$ or $\mathbb{R}^{n\times m}$, respectively. For a symmetric matrix $\boldsymbol{M}$, $\boldsymbol{M} > 0 (\boldsymbol{M} \geqslant 0)$ means that it is a positive (semi-)definite matrix. $\lambda(\cdot), \bar{\sigma}(\cdot)$ denote the eigenvalue and maximum singular value of a matrix, respectively. The superscript T is used for the transpose and $\boldsymbol{I}$ denotes the identity matrix of appropriate dimension. $\mathbb{C}^-$ is the open left-half of the complex plane.

2. Backgrounds

In this section, we give the problem description and recall some preliminary results.

2.1. Problem description

Let us consider the following linear time-invariant (LTI) continuous-time system with external disturbances:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_1\boldsymbol{w}(t) + \boldsymbol{B}_2\boldsymbol{u}(t) \tag{1}$$

$$\boldsymbol{z}(t) = \begin{bmatrix} \boldsymbol{C}\boldsymbol{x}(t) \\ \boldsymbol{D}\boldsymbol{u}(t) \end{bmatrix} \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state, $\boldsymbol{u} \in \mathbb{R}^m$ is the control input, $\boldsymbol{w} \in \mathbb{R}^q$ is the external disturbance and $\boldsymbol{w}(t) \in L_2[0,\infty)$, and $\mathbf{z} \in \mathbb{R}^p$ is the objective output. $\boldsymbol{A}$, $\boldsymbol{B}_1$, $\mathbf{B}_2$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are known constant matrices with appropriate dimensions.

We consider the following linear state feedback control law:

$$\boldsymbol{u}(t) = \boldsymbol{K}\boldsymbol{x}(t) \tag{3}$$

where $\boldsymbol{K} \in \mathbb{R}^{m\times n}$ is a state feedback gain matrix to be determined.

Substituting the control law (3) into (1) and (2) leads to the following closed-loop system:

$$\dot{\boldsymbol{x}}(t) = (\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{K})\boldsymbol{x}(t) + \boldsymbol{B}_1\boldsymbol{w}(t) \tag{4}$$

$$\boldsymbol{z}(t) = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{D}\boldsymbol{K} \end{bmatrix}\boldsymbol{x}(t). \tag{5}$$

The H_∞ control problem under consideration is to find a state feedback control law of form (3) such that the closed-loop system (4) and (5) is asymptotically stable, and satisfies

$$\frac{\int_0^\infty \|\boldsymbol{z}(t)\|^2 dt}{\int_0^\infty \|\boldsymbol{w}(t)\|^2 dt} = \frac{\int_0^\infty (\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u})dt}{\int_0^\infty (\boldsymbol{w}^T\boldsymbol{w})dt} \leqslant \gamma^2 \tag{6}$$

for all nonzero $\boldsymbol{w}(t) \in L_2[0,\infty)$, where $\boldsymbol{Q} = \boldsymbol{C}^T\boldsymbol{C}$, $\boldsymbol{R} = \boldsymbol{D}^T\boldsymbol{D}$ and $\gamma > 0$ is some prescribed level of disturbance attenuation.

In order that a stabilizing controller exists, the following standard assumptions are necessary:

(1) The pair $(\boldsymbol{A},\boldsymbol{B}_2)$ is stabilizable.
(2) The pair $(\boldsymbol{C},\boldsymbol{A})$ has no unobservable modes on the imaginary axis.

Lemma 1 [11]. *Suppose the standard assumptions hold. Then there exists a linear control law (3), such that the closed-loop system (4) and (5) is asymptotically stable and satisfies (6) for a prescribed level $\gamma > 0$ if and only if the ARE*

$$\mathcal{F}(\boldsymbol{P}) \triangleq \boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} + \gamma^{-2}\boldsymbol{P}\boldsymbol{B}_1\boldsymbol{B}_1^T\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P} = 0 \tag{7}$$

has a stabilizing solution $\boldsymbol{P} \geqslant 0$, i.e., $\lambda(\boldsymbol{A} + \gamma^{-2}\boldsymbol{B}_1\boldsymbol{B}_1^T\boldsymbol{P} - \boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}) \in \mathbb{C}^-$. In this case, the gain matrix of a suitable state-feedback controller is given by

$$\boldsymbol{K} = -\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}.$$

2.2. Offline method for the solution of H_∞ ARE

Note from Lemma 1 that the H_∞ control problem is reduced to find a stabilizing solution $\boldsymbol{P} \geqslant 0$ of ARE (7). In [16], the H_∞ ARE was converted to a sequence of H_2 AREs of LQ optimal control problems. As mentioned in [16], Kleinman's algorithm [15] can be used to solve each H_2 ARE. Here, we first review the Kleinman's method for solving an H_2 ARE.

Let us consider the following H_2 ARE for the LQ optimal control:

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} - \boldsymbol{P}\boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P} = 0. \tag{8}$$

Then Kleinman's method is given as follows:

Algorithm 1 (*Kleinman's method*).

Step 1 . Give an initial stabilizing matrix $\boldsymbol{P}^{(0)}$, i.e., $\lambda(\boldsymbol{A} - \boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}^{(0)}) \in \mathbb{C}^-$. Let $i = 0$.

Step 2 . Solve the following LE for $\boldsymbol{P}^{(i+1)}$:

$$\widetilde{\boldsymbol{A}}_i^T\boldsymbol{P}^{(i+1)} + \boldsymbol{P}^{(i+1)}\widetilde{\boldsymbol{A}}_i + \boldsymbol{Q} + \boldsymbol{P}^{(i)}\boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}^{(i)} = 0$$

where $\widetilde{\boldsymbol{A}}_i = \boldsymbol{A} - \boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}^{(i)}$.

Step 3 . Set $i = i + 1$. If $\|\boldsymbol{P}^{(i)} - \boldsymbol{P}^{(i-1)}\| \leqslant \varepsilon$($\varepsilon$ is a small positive real number), stop and output $\boldsymbol{P}^{(i)}$ as the solution $\boldsymbol{P}$ of H_2 ARE (8), else, go to Step 2 and continue.

Remark 1. It is seen from Algorithm 1 that the nonlinear H_2 ARE can be iteratively solved with a sequence of linear LEs.

Using the same manipulations as in Proposition 1 of [29], the iterative algorithm in [16] for solving the ARE (7) is to solve the following sequence of H_2 AREs.

$$\boldsymbol{A}_i^T\boldsymbol{P}_{\boldsymbol{u}}^{(i+1)} + \boldsymbol{P}_{\boldsymbol{u}}^{(i+1)}\boldsymbol{A}_i - \boldsymbol{P}_{\boldsymbol{u}}^{(i+1)}\boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}_{\boldsymbol{u}}^{(i+1)} + \boldsymbol{Q}^{(i)} = 0, \quad i = 0,\ldots,\infty \tag{9}$$

where $\boldsymbol{Q}^{(i)} = \boldsymbol{Q} - \gamma^{-2}\boldsymbol{P}_{\boldsymbol{w}}^{(i)}\boldsymbol{B}_1\boldsymbol{B}_1^T\boldsymbol{P}_{\boldsymbol{w}}^{(i)}$ and $\boldsymbol{A}_i = \boldsymbol{A} + \gamma^{-2}\boldsymbol{B}_1\boldsymbol{B}_1^T\boldsymbol{P}_{\boldsymbol{w}}^{(i)}$.

As described in [16], applying Algorithm 1 for solving each H_2 ARE (9) yields the following algorithm.

Algorithm 2.

Step 1. Let $\boldsymbol{P}_{\boldsymbol{w}}^{(0)} = 0$, and $i = 0$.

Step 2. Let $\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,0)} = \boldsymbol{P}_{\boldsymbol{w}}^{(i)}$, and $j = 0$.

Step 3. Solve the following LE for $\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j+1)}$:

$$\boldsymbol{A}_{i,j}^T\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j+1)} + \boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j+1)}\boldsymbol{A}_{i,j} + \boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j)}\boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j)} + \boldsymbol{Q}^{(i)} = 0 \tag{10}$$

where $\boldsymbol{A}_{i,j} = \boldsymbol{A}_i - \boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j)}$.

Step 4. Set $j = j + 1$. If $\|\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j)} - \boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j-1)}\| \leqslant \varepsilon_1$($\varepsilon_1$ is a small positive real number), go to Step 5, else, go to Step 3.

Step 5. Set $\boldsymbol{P}_{\boldsymbol{w}}^{(i+1)} = \boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j)}$ and $i = i + 1$. If $\left\|\boldsymbol{P}_{\boldsymbol{w}}^{(i)} - \boldsymbol{P}_{\boldsymbol{w}}^{(i-1)}\right\| \leqslant \varepsilon$($\varepsilon$ is a small positive real number), stop and output $\boldsymbol{P}_{\boldsymbol{w}}^{(i)}$ as the solution $\boldsymbol{P}$ of ARE (7), else, go to Step 2 and continue.

Remark 2. It is observed from Algorithm 2 that the solution of H_∞ ARE (7) is firstly converted to the solution of a series of H_2 AREs (9), each of which is further reduced to a sequence of LEs (10) according to Algorithm 1. Thus, Algorithm 2 has two iterative loops: (i) the inner loop (i.e., from Step 3 to 4) for the iterative solution of an H_2 ARE (9) by solving a series of LEs (10), and (ii) the outer loop (i.e., from Step 2 to 5) for the iterative solution of ARE (7) by solving a series of H_2 AREs (9).

We notice that Algorithm 2 is an offline method, which requires full system dynamics of (1). In [29], by employing VI to solve each H_2 ARE (9), an online ADP algorithm was developed for the case that the system dynamic is partially unknown. The details of this algorithm are presented in Subsection 2.3.

2.3. Online method for the solution of H_∞ ARE

It is well known that the H_∞ control problem can be equivalently viewed as a two-player zero-sum game, where the control input $\boldsymbol{u}(t)$ is a minimizing player and the disturbance $\boldsymbol{w}(t)$ is a maximizing one. The solution of the H_∞ control problem is the saddle point stabilizing equilibrium of the two-player zero-sum game. Define the following infinite horizon quadratic performance index:

$$V(\boldsymbol{u},\boldsymbol{w}) = \int_0^\infty (\|\boldsymbol{z}(t)\|^2 - \gamma^2\|\boldsymbol{w}(t)\|^2)dt = \int_0^\infty (\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u} - \gamma^2\boldsymbol{w}^T\boldsymbol{w})dt. \tag{11}$$

The goal of the two-player zero-sum game is to find the following saddle point stabilizing equilibrium $(\boldsymbol{u}^*,\boldsymbol{w}^*)$:

$$\boldsymbol{u}^*(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}\boldsymbol{x}, \tag{12}$$

$$\boldsymbol{w}^*(t) = \gamma^{-2}\boldsymbol{B}_1^T\boldsymbol{P}\boldsymbol{x}, \tag{13}$$

such that

$$V(\boldsymbol{u}^*,\boldsymbol{w}) \leqslant V(\boldsymbol{u}^*,\boldsymbol{w}^*) \leqslant V(\boldsymbol{u},\boldsymbol{w}^*),$$

where matrix $\boldsymbol{P}$ satisfies the ARE (7).

In [29], a VI based ADP method was used to online solve the H_2 ARE (9), where two persons play online to learn the solution of ARE (7), i.e., the saddle point stabilizing equilibrium $(\boldsymbol{u}^*,\boldsymbol{w}^*)$. Once the solution is obtained, the H_∞ controller $\boldsymbol{u}^*(t)$ can be used for real time control. Here, we briefly present this method without deduction as follows.

Algorithm 3.

Step 1. Let $\boldsymbol{P}_{\boldsymbol{w}}^{(0)} = 0, \boldsymbol{w}^{(0)} = 0$ and $i = 0$.

Step 2. Let $\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,0)} = \boldsymbol{P}_{\boldsymbol{w}}^{(i)}$, $\boldsymbol{u}^{(i+1,0)} = -\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,0)}\boldsymbol{x}$ and $j = 0$.

Step 3. Online solve the following equation for matrix $\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j+1)}$:

$$\begin{aligned}\boldsymbol{x}^T(t)\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j+1)}\boldsymbol{x}(t) = {} & \boldsymbol{x}^T(t+\Delta t)\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j)}\boldsymbol{x}(t+\Delta t) + \int_t^{t+\Delta t}(\boldsymbol{x}^T(\tau)\boldsymbol{Q}\boldsymbol{x}(\tau) + (\boldsymbol{u}^{(i+1,j)}(\tau))^T\boldsymbol{R}\boldsymbol{u}^{(i+1,j)}(\tau) \\ & - \gamma^2(\boldsymbol{w}^{(i)}(\tau))^T\boldsymbol{w}^{(i)}(\tau))d\tau\end{aligned} \tag{14}$$

where $\boldsymbol{x}(\tau)$ is the state of the system $\dot{\boldsymbol{x}}(\tau) = \boldsymbol{A}\boldsymbol{x}(\tau) + \boldsymbol{B}_1\boldsymbol{w}^{(i)}(\tau) + \boldsymbol{B}_2\boldsymbol{u}^{(i+1,j)}(\tau)$.

Step 4. Update $\boldsymbol{u}^{(i+1,j+1)}$ with

$$\boldsymbol{u}^{(i+1,j+1)} = -\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j+1)}\boldsymbol{x}. \tag{15}$$

Step 5. Set $j = j + 1$. If $\left\|\boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j)} - \boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j-1)}\right\| \leqslant \varepsilon_1$ (ε_1 is a small positive real number), go to Step 6, else, go to Step 3.

Step 6. Set $\boldsymbol{P}_{\boldsymbol{w}}^{(i+1)} = \boldsymbol{P}_{\boldsymbol{u}}^{(i+1,j)}$. Update $\boldsymbol{w}^{(i+1)}$ with

$$\boldsymbol{w}^{(i+1)} = \gamma^{-2}\boldsymbol{B}_1^T\boldsymbol{P}_{\boldsymbol{w}}^{(i+1)}\boldsymbol{x}. \tag{16}$$

Step 7. Set $i = i + 1$. If $\left\|\boldsymbol{P}_{\boldsymbol{w}}^{(i)} - \boldsymbol{P}_{\boldsymbol{w}}^{(i-1)}\right\| \leqslant \varepsilon$ (ε is a small positive real number), stop and output $\boldsymbol{P}_{\boldsymbol{w}}^{(i)}$ as the solution $\boldsymbol{P}$ of ARE (7), go to Step 2 and continue.

In Algorithm 3, Steps 3–5 is a VI procedure for online solution of an H_2 ARE (9) without requiring the internal system dynamics.

Remark 3. To simplify the notation, we describe all algorithms in uniform formulation by using the H_2 ARE with expression (9) instead of that in [16]. Although the descriptions of Algorithms 2 and 3 are seemingly different from those in [16] (by using Kleinman's algorithm to solve each H_2 ARE therein) and [29], they are the same procedures essentially.

Remark 4. In Algorithms 2 and 3, only one of the two players is actively learning and improving its policy. This results in two iterative loops in the iteration process for the solution of ARE. The algorithms are built on interplay between a learning phase, performed by the control player $\boldsymbol{u}(t)$ that is learning in order to optimize its behavior, and a policy update phase, performed by the disturbance player $\boldsymbol{w}(t)$ that is gradually increasing its detrimental effect. That is to say, when one player's policy is updated, the other remains invariant. Therefore, these methods are often time-consuming and inefficient.

3. SPUA for the solution of H_∞ ARE

This section presents the main results of this paper. To overcome the low efficiency of the above algorithms, we propose a SPUA that involves only one iterative loop since it updates policies of both players simultaneously for the solution of H_∞ ARE (7). Firstly, an offline SPUA is proposed and its convergence is established rigorously. Then, an online SPUA without requiring the internal system dynamics is developed, whose convergence is guaranteed by proving that it is equivalent to the offline SPUA mathematically.

3.1. Offline SPUA for the solution of H_∞ ARE

In Algorithm 2, the ARE (7) is firstly converted into a sequence of H_2 AREs (9), which are corresponding to a sequence of optimal control problems. Then, based on Algorithm 1, the solution of each H_2 ARE is obtained by solving a series of LEs. The SPUA proposed in this paper directly transforms the ARE (7) into a sequence of LEs. Therefore, the SPUA is much simpler than Algorithm 2 for implementation.

Algorithm 4 (*Offline SPUA*).

Step 1. Give an initial matrix $\boldsymbol{P}^{(0)} \in \mathbb{P}_0$ ($\mathbb{P}_0 \subset \mathbb{P}$ is determined by Lemma 3). Let $i = 0$.

Step 2. Solve the following LE for $\boldsymbol{P}^{(i+1)}$:

$$\overline{\boldsymbol{A}}_i^T \boldsymbol{P}^{(i+1)} + \boldsymbol{P}^{(i+1)} \overline{\boldsymbol{A}}_i + \overline{\boldsymbol{Q}}^{(i)} = 0 \tag{17}$$

where $\overline{\boldsymbol{A}}_i = \boldsymbol{A} + \gamma^{-2}\boldsymbol{B}_1\boldsymbol{B}_1^T\boldsymbol{P}^{(i)} - \boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}^{(i)}$ and

$$\overline{\boldsymbol{Q}}^{(i)} = \boldsymbol{Q} - \gamma^{-2}\boldsymbol{P}^{(i)}\boldsymbol{B}_1\boldsymbol{B}_1^T\boldsymbol{P}^{(i)} + \boldsymbol{P}^{(i)}\boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}^{(i)}.$$

Step 3. Set $i = i + 1$. If $\|\boldsymbol{P}^{(i)} - \boldsymbol{P}^{(i-1)}\| \leqslant \varepsilon$ (ε is a small positive real number), stop and output $\boldsymbol{P}^{(i)}$, else, go to Step 2 and continue.

Remark 5. It is noticed that Algorithm 4 solves ARE (7) by iteratively solving a sequence of LEs (17), similarly, an inner loop of Algorithm 2 also solves an H_2 ARE (9) by iteratively solving a sequence of LEs (10). Since both LEs (10) and (17) are linear matrix equations of the same form, the computational loads for solving them are fundamentally the same. Thus, the computational load of Algorithm 4 is nearly the same as that of an inner loop in Algorithm 2. However, Algorithm 2 needs an outer loop to solve a series of H_2 ARE rather than one. Therefore, Algorithm 4 is naturally more efficient than Algorithm 2.

Remark 6. The obvious difference between Algorithms 2 and 4 is that, Algorithm 2 updates the control and disturbance policies nonsimultaneously (i.e., one player updates its policy while the other remains invariant), which brings two iterative loops, while SPUA (i.e., Algorithm 4) updates the control and disturbance policies simultaneously, in which only one iterative loop is needed. Moreover, we notice that the convergence of Algorithm 2 can be proved by showing that the ARE (7) is successively approximated by a sequence of H_2 AREs (9), and then demonstrating that each H_2 ARE is successively approximated by a sequence of LEs (10) (i.e., Kleinman's method). However, the ARE (7) is directly successively approximated by a sequence of LEs (17) in Algorithm 4, thus new tools are needed to establish its convergence.

Next, we will establish the convergence of Algorithm 4. Namely, we want to show that the solution of LEs (17) converges to the solution of ARE (7) when i goes to infinity.

To this end, let us consider such a Banach space $\mathbb{P} \subset \{\boldsymbol{P} \in \mathbb{R}^{n\times n} \mid \boldsymbol{P} \geqslant 0, \boldsymbol{P}^T = \boldsymbol{P}\}$ equipped with a norm $\|\boldsymbol{P}\| = \bar{\sigma}(\boldsymbol{P})$, and consider the mapping $\mathcal{F} : \mathbb{P} \to \mathbb{P}$ defined in (7). Then, the Fréchet differential [6] of $\mathcal{F}(\boldsymbol{P})$ at $\boldsymbol{P}$ can be obtained as

$$\boldsymbol{d}\mathcal{F}(\boldsymbol{P};\boldsymbol{M}) = \mathcal{F}'(\boldsymbol{P})\boldsymbol{M} = \overline{\boldsymbol{A}}^T\boldsymbol{M} + \boldsymbol{M}\overline{\boldsymbol{A}}, \tag{18}$$

where $\overline{\boldsymbol{A}} = \boldsymbol{A} + \gamma^{-2}\boldsymbol{B}_1\boldsymbol{B}_1^T\boldsymbol{P} - \boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}$, $\boldsymbol{M} \in \mathbb{P}$. $\mathcal{F}'(\boldsymbol{P})$ called the Fréchet derivative of $\mathcal{F}(\boldsymbol{P})$ at $\boldsymbol{P}$, is a linear operator mapping $\mathbb{P}$ into itself.

Define a mapping $\mathcal{T} : \mathbb{P} \to \mathbb{P}$ as follows:

$$\mathcal{T}\boldsymbol{P} = \boldsymbol{P} - (\mathcal{F}'(\boldsymbol{P}))^{-1}\mathcal{F}(\boldsymbol{P}). \tag{19}$$

We construct a Newton iterative sequence $\{\boldsymbol{P}^{(i)}\}$ as

$$\boldsymbol{P}^{(i+1)} = \mathcal{T}\boldsymbol{P}^{(i)}, \quad i = 0, 1, 2, \ldots. \tag{20}$$

Under some proper assumptions (as shown in Lemma 2 below), the sequence $\{\boldsymbol{P}^{(i)}\}$ given in (20) can converge to the unique solution of the fixed-point equation $\mathcal{T}\boldsymbol{P} = \boldsymbol{P}$, i.e., the solution of ARE $\mathcal{F}(\boldsymbol{P}) = 0$. The convergence and the error bound of sequence $\{\boldsymbol{P}^{(i)}\}$ are guaranteed by the following Kantorovtich's Theorem [14,23].

Lemma 2 (*Kantorovtich's Theorem*). *Assume for some* $\boldsymbol{P}^{(0)} \in \mathbb{P}_1 \subset \mathbb{P}$ *such that* $(\mathcal{F}'(\boldsymbol{P}^{(0)}))^{-1}$ *exists and that*

(1) $\|(\mathcal{F}'(\boldsymbol{P}^{(0)}))^{-1}\| \leqslant B_0$, (21)

(2) $\|(\mathcal{F}'(\boldsymbol{P}^{(0)}))^{-1}\mathcal{F}(\boldsymbol{P}^{(0)})\| \leqslant \eta$, (22)

(3) $\|\mathcal{F}'(\boldsymbol{P}^{(1)}) - \mathcal{F}'(\boldsymbol{P}^{(2)})\| \leqslant K\|\boldsymbol{P}^{(1)} - \boldsymbol{P}^{(2)}\|$, for all $\boldsymbol{P}^{(1)}, \boldsymbol{P}^{(2)} \in \mathbb{P}_1$, (23)

with $h = B_0 K\eta \leqslant \frac{1}{2}$. *Let*

$$\mathbb{P}_2 = \{\boldsymbol{P} \,|\, \|\boldsymbol{P} - \boldsymbol{P}^{(0)}\| \leqslant \sigma\}, \quad \text{where } \sigma = \left(\frac{1 - \sqrt{1 - 2h}}{h}\right)\eta. \tag{24}$$

Now, if $\mathbb{P}_2 \subset \mathbb{P}_1$, then, the sequence $\{\boldsymbol{P}^{(i)}\}$ given in (20) is well defined, remains in $\mathbb{P}_2$, and converges to $\boldsymbol{P} \in \mathbb{P}_2$ such that $\mathcal{F}(\boldsymbol{P}) = 0$. In addition,

$$\|\boldsymbol{P} - \boldsymbol{P}^{(i)}\| \leqslant \frac{\eta}{h}\left(\frac{(1 - \sqrt{1 - 2h})^{2^i}}{2^i}\right), \quad i = 0, 1, 2, \ldots \tag{25}$$

It is observed from Lemma 2 that $\mathbb{P}_1$ is required to be suitably chosen. The following lemma gives a method to determine a suitable $\mathbb{P}_0$ satisfying $\mathbb{P}_0 \subset \mathbb{P}_1$, so that $\mathbb{P}_0$ conversely guarantees the hypotheses of Lemma 2.

Lemma 3. (19) *Suppose* $\boldsymbol{P} \geqslant 0$ *is the stabilizing solution of equation* $\mathcal{F}(\boldsymbol{P}) = 0$. *If* $\|(\mathcal{F}'(\boldsymbol{P}))^{-1}\| \leqslant B^*$, *and*

$$\mathbb{P}_3 = \{\widehat{\boldsymbol{P}} \,|\, \|\widehat{\boldsymbol{P}} - \boldsymbol{P}\| \leqslant (1/B^*K)\} \subset \mathbb{P}_1 \tag{26}$$

then, the hypotheses of Lemma 2 is satisfied, i.e., for each $\boldsymbol{P}^{(0)} \in \mathbb{P}_0$, $h \leqslant 1/2$, B_0 *and* η *in conditions (21) and (22) are*

$$B_0 = \frac{B^*}{1 - B^*K\|\boldsymbol{P}^{(0)} - \boldsymbol{P}\|} \geqslant \|(\mathcal{F}'(\boldsymbol{P}^{(0)}))^{-1}\| \quad \text{and}$$
$$\eta = \frac{1 - \frac{1}{2}B^*K\|\boldsymbol{P}^{(0)} - \boldsymbol{P}\|}{1 - B^*K\|\boldsymbol{P}^{(0)} - \boldsymbol{P}\|}\|\boldsymbol{P}^{(0)} - \boldsymbol{P}\| \geqslant \|(\mathcal{F}'(\boldsymbol{P}^{(0)}))^{-1}\mathcal{F}(\boldsymbol{P}^{(0)})\|,$$

where $\mathbb{P}_0$ *is defined as*

$$\mathbb{P}_0 = \{\widehat{\boldsymbol{P}} \,|\, \|\widehat{\boldsymbol{P}} - \boldsymbol{P}\| \leqslant (2 - \sqrt{2})/(2B^*K)\}. \tag{27}$$

Lemmas 2 and 3 imply that if $\boldsymbol{P}^{(0)}$ is chosen in $\mathbb{P}_0$ (that defined by (27)), the Newton sequence $\{\boldsymbol{P}^{(i)}\}$ generated by (20) can converge to the unique fixed point of (19), i.e., the positive definite stabilizing solution of ARE $\mathcal{F}(\boldsymbol{P}) = 0$, and the error bound is given in (25). With Lemmas 2 and 3, we prove that the sequence $\{\boldsymbol{P}^{(i)}\}$ generated by Algorithm 4 is essentially a Newton sequence in Theorem 1, and give the convergence of Algorithm 4 in Theorem 2.

Theorem 1. *Let* $\mathcal{T}$ *be a mapping defined by (19). Then, the sequence* $\{\boldsymbol{P}^{(i)}\}$ *generated by Algorithm 4 and the Newton iteration (20) are equivalent.*

Proof. It follows from (20) that

$$\boldsymbol{P}^{(i+1)} = \mathcal{T}\boldsymbol{P}^{(i)} = \boldsymbol{P}^{(i)} - (\mathcal{F}'(\boldsymbol{P}^{(i)}))^{-1}\mathcal{F}(\boldsymbol{P}^{(i)})$$

which is equivalently rewritten as

$$\mathcal{F}'(\boldsymbol{P}^{(i)})\boldsymbol{P}^{(i+1)} = \mathcal{F}'(\boldsymbol{P}^{(i)})\boldsymbol{P}^{(i)} - \mathcal{F}(\boldsymbol{P}^{(i)}). \tag{28}$$

From (18) and (7), we have

$$\mathcal{F}'(\boldsymbol{P}^{(i)})\boldsymbol{P}^{(i+1)} = \overline{\boldsymbol{A}}_i^T\boldsymbol{P}^{(i+1)} + \boldsymbol{P}^{(i+1)}\overline{\boldsymbol{A}}_i \tag{29}$$
$$\mathcal{F}'(\boldsymbol{P}^{(i)})\boldsymbol{P}^{(i)} = \overline{\boldsymbol{A}}_i^T\boldsymbol{P}^{(i)} + \boldsymbol{P}^{(i)}\overline{\boldsymbol{A}}_i \tag{30}$$
$$\mathcal{F}(\boldsymbol{P}^{(i)}) = \boldsymbol{A}^T\boldsymbol{P}^{(i)} + \boldsymbol{P}^{(i)}\boldsymbol{A} + \overline{\boldsymbol{Q}}^{(i)}. \tag{31}$$

Substituting (29)–(31) into (28), we can obtain

$$\overline{\boldsymbol{A}}_i^T \boldsymbol{P}^{(i+1)} + \boldsymbol{P}^{(i+1)} \overline{\boldsymbol{A}}_i = -\overline{\boldsymbol{Q}}^{(i)}$$

This completes the proof. □

Theorem 2. *Let $\boldsymbol{P} \geqslant 0$ is the stabilizing solution of ARE (7) and $\{\boldsymbol{P}^{(i)}\}$ be the sequence generated by Algorithm 4. Then, when $i \to \infty$, $\boldsymbol{P}^{(i)} \to \boldsymbol{P}$.*

Proof. Theorem 1 shows that the sequence $\{\boldsymbol{P}^{(i)}\}$ generated by Algorithm 4 is equivalent to the Newton sequence obtained by (20). According to Algorithm 4, we have the initial matrix $\boldsymbol{P}^{(0)} \in \mathbb{P}_0$ where $\mathbb{P}_0$ is defined by (27). It follows from Lemma 3 that the $\mathbb{P}_0$ can conversely guarantee the hypotheses of Lemma 2. Therefore, we can conclude from Lemma 2 that the sequence $\{\boldsymbol{P}^{(i)}\}$ is convergent, i.e., $\boldsymbol{P}^{(i)} \to \boldsymbol{P}$, when $i \to \infty$. □

3.2. Online SPUA for the solution of H_∞ ARE

The SPUA of Algorithm 4 is an offline approach, which requires the full knowledge of the system dynamics. In this subsection, we make use of the thought of RL techniques, and propose an online SPUA to learn the solution of ARE (7) in an online scheme without requiring the knowledge of internal system dynamics.

It follows from (11) that, given arbitrary control action $\boldsymbol{u}(t)$ and disturbance signal $\boldsymbol{w}(t)$ with initial system state $\boldsymbol{x}(t)$, the cost function is

$$V(\boldsymbol{x}(t)) = \int_t^\infty \Xi(\tau) d\tau,$$

which can be rewritten as

$$V(\boldsymbol{x}(t)) = \int_t^{t+\Delta t} \Xi(\tau) d\tau + V(\boldsymbol{x}(t+\Delta t)), \tag{32}$$

where $\Xi(\tau) \triangleq \boldsymbol{x}^T(\tau)\boldsymbol{Q}\boldsymbol{x}(\tau) + \boldsymbol{u}^T(\tau)\boldsymbol{R}\boldsymbol{u}(\tau) - \gamma^2 \boldsymbol{w}^T(\tau)\boldsymbol{w}(\tau)$. For a linear two-player zero-sum game, $V(\boldsymbol{x}(t))$ can be parameterized as

$$V(\boldsymbol{x}(t)) = \boldsymbol{x}^T(t)\boldsymbol{P}\boldsymbol{x}(t), \tag{33}$$

then, (32) is rewritten as

$$\boldsymbol{x}^T(t)\boldsymbol{P}\boldsymbol{x}(t) = \boldsymbol{x}^T(t+\Delta t)\boldsymbol{P}\boldsymbol{x}(t+\Delta t) + \int_t^{t+\Delta t} \Xi(\tau) d\tau. \tag{34}$$

Based on (34), an online SPUA is developed for the solution of ARE (7) as follows.

Algorithm 5 (*Online SPUA*).

Step 1. Give an initial matrix $\boldsymbol{P}^{(0)} \in \mathbb{P}_0$, and let $\boldsymbol{u}^{(0)} = -\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}^{(0)}\boldsymbol{x}$, $\boldsymbol{w}^{(0)} = \gamma^{-2}\boldsymbol{B}_1^T\boldsymbol{P}^{(0)}\boldsymbol{x}$. Set $i = 0$.

Step 2. With policies $\boldsymbol{u}^{(i)}$ and $\boldsymbol{w}^{(i)}$, solve the following equation for $\boldsymbol{P}^{(i+1)}$:

$$\boldsymbol{x}^T(t)\boldsymbol{P}^{(i+1)}\boldsymbol{x}(t) = \boldsymbol{x}^T(t+\Delta t)\boldsymbol{P}^{(i+1)}\boldsymbol{x}(t+\Delta t) + \int_t^{t+\Delta t} \Xi^{(i)}(\tau) d\tau \tag{35}$$

where $\Xi^{(i)}(\tau) \triangleq \boldsymbol{x}^T(\tau)\boldsymbol{Q}\boldsymbol{x}(\tau) + (\boldsymbol{u}^{(i)}(\tau))^T\boldsymbol{R}\boldsymbol{u}^{(i)}(\tau) - \gamma^2(\boldsymbol{w}^{(i)}(\tau))^T\boldsymbol{w}^{(i)}(\tau)$ and $\boldsymbol{x}(\tau)$ is the state of the system

$$\dot{\boldsymbol{x}}(\tau) = \boldsymbol{A}\boldsymbol{x}(\tau) + \boldsymbol{B}_1\boldsymbol{w}^{(i)}(\tau) + \boldsymbol{B}_2\boldsymbol{u}^{(i)}(\tau). \tag{36}$$

Step 3. Update control policy $\boldsymbol{u}^{(i+1)}$ and disturbance policy $\boldsymbol{w}^{(i+1)}$ as follows:

$$\boldsymbol{u}^{(i+1)} = -\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}^{(i+1)}\boldsymbol{x} \tag{37}$$

$$\boldsymbol{w}^{(i+1)} = \gamma^{-2}\boldsymbol{B}_1^T\boldsymbol{P}^{(i+1)}\boldsymbol{x}. \tag{38}$$

Step 4. Set $i = i + 1$. If $\|\boldsymbol{P}^{(i)} - \boldsymbol{P}^{(i-1)}\| \leqslant \varepsilon$ (ε is a small positive real number), stop and output $\boldsymbol{P}^{(i)}$, else, go to Step 2 and continue.

Remark 7. Observe that the online SPUA (i.e., Algorithm 5) does not require the internal system dynamic matrix $\boldsymbol{A}$, whose information is embedded in the online measurement of the states $\boldsymbol{x}(t)$ and $\boldsymbol{x}(t+\Delta t)$, and evaluation of the cost $\int_t^{t+\Delta t}\Xi^{(i)}(\tau)d\tau$. The online SPUA follows the basic procedure of policy iteration in RL, which involves policy evaluation (in Step 2) and policy improvement (in Step 3). Hence, it can also be viewed as a RL technique for two players to learn their optimal actions in the unknown environment.

Notice from Algorithm 5 that the control and disturbance policies update simultaneously with (37) and (38), and thus only one iterative loop is needed. The convergence of Algorithm 5 is guaranteed by the following theorem.

Theorem 3. *The iteration in Algorithm 5 is equivalent to the iteration in Algorithm 4, i.e., the iteration of* Eqs. (35)–(38) *is equivalent to the iteration of* Eq. (17).

Proof. With policies $\boldsymbol{u}^{(i)}$ and $\boldsymbol{w}^{(i)}$, it follows from (36) that the closed-loop system is $\dot{\boldsymbol{x}}(\tau)=\overline{\boldsymbol{A}}_i\boldsymbol{x}(\tau)$. Consider the following derivative:

$$\begin{aligned}\frac{d}{d\tau}\boldsymbol{x}^T(\tau)\boldsymbol{P}^{(i+1)}\boldsymbol{x}(\tau)&=\dot{\boldsymbol{x}}^T(\tau)\boldsymbol{P}^{(i+1)}\boldsymbol{x}(\tau)+\boldsymbol{x}^T(\tau)\boldsymbol{P}^{(i+1)}\dot{\boldsymbol{x}}(\tau)=\boldsymbol{x}^T(\tau)\overline{\boldsymbol{A}}_i^T\boldsymbol{P}^{(i+1)}\boldsymbol{x}(\tau)+\boldsymbol{x}^T(\tau)\boldsymbol{P}^{(i+1)}\overline{\boldsymbol{A}}_i\boldsymbol{x}(\tau)\\&=\boldsymbol{x}^T(\tau)\left(\overline{\boldsymbol{A}}_i^T\boldsymbol{P}^{(i+1)}+\boldsymbol{P}^{(i+1)}\overline{\boldsymbol{A}}_i\right)\boldsymbol{x}(\tau).\end{aligned}\tag{39}$$

Integrating (39) from $\tau=t$ to $\tau=t+\Delta t$ yields

$$\boldsymbol{x}^T(t+\Delta t)\boldsymbol{P}^{(i+1)}\boldsymbol{x}(t+\Delta t)-\boldsymbol{x}^T(t)\boldsymbol{P}^{(i+1)}\boldsymbol{x}(t)=\int_t^{t+\Delta t}\boldsymbol{x}^T(\tau)\left(\overline{\boldsymbol{A}}_i^T\boldsymbol{P}^{(i+1)}+\boldsymbol{P}^{(i+1)}\overline{\boldsymbol{A}}_i\right)\boldsymbol{x}(\tau)d\tau.\tag{40}$$

It follows from (35)–(38) that

$$\begin{aligned}\boldsymbol{x}^T(t+\Delta t)\boldsymbol{P}^{(i+1)}\boldsymbol{x}(t+\Delta t)-\boldsymbol{x}^T(t)\boldsymbol{P}^{(i+1)}\boldsymbol{x}(t)&=-\int_t^{t+\Delta t}\Xi^{(i)}(\tau)d\tau\\&=-\int_t^{t+\Delta t}\boldsymbol{x}^T(\tau)\left(\boldsymbol{Q}-\gamma^{-2}\boldsymbol{P}^{(i)}\boldsymbol{B}_1\boldsymbol{B}_1^T\boldsymbol{P}^{(i)}+\boldsymbol{P}^{(i)}\boldsymbol{B}_2\boldsymbol{R}^{-1}\boldsymbol{B}_2^T\boldsymbol{P}^{(i)}\right)\boldsymbol{x}(\tau)d\tau\\&=-\int_t^{t+\Delta t}\boldsymbol{x}^T(\tau)\overline{\boldsymbol{Q}}^{(i)}\boldsymbol{x}(\tau)d\tau.\end{aligned}\tag{41}$$

From (40) and (41), we can get $\overline{\boldsymbol{A}}_i^T\boldsymbol{P}^{(i+1)}+\boldsymbol{P}^{(i+1)}\overline{\boldsymbol{A}}_i=-\overline{\boldsymbol{Q}}^{(i)}$, i.e., (17) holds. □

According to Theorems 2 and 3, it is immediate that Algorithm 5 is convergent. Obviously, the convergence of offline SPUA provides the mathematical foundation for the online SPUA.

Remark 8. Theorem 3 shows that Algorithm 5 is mathematically equivalent to Algorithm 4, which also needs only one iterative loop. In contrast, the VI algorithm is used to solve H_2 AREs in Algorithm 3, which involves two loops. Therefore, as mentioned in Remark 6, Algorithms 3 and 5 are also essentially different.

Remark 9. It is necessary to discuss the relationship and differences between Algorithms 3 and 5 for solving ARE (7). The main common feature of them is that, both of them are online methods without requiring the internal system dynamics. However, just as mentioned in Remark 8, they are essentially different. The main differences are reflected in two aspects: (1) In Algorithm 3, an inner loop (i.e., the VI algorithm) mathematically solves an H_2 ARE (9) online, while Algorithm 5 directly solves ARE (7) online; (2) Algorithm 5 is mathematically equivalent to a Newton's iteration method as shown above, which has a quadratic convergence rate [6], while the VI algorithm used in Algorithm 3 for solving H_2 AREs only has a linear convergence rate [5]. This means that the convergence of the whole Algorithm 5 is faster than an inner loop in Algorithm 3. Therefore, it is natural that Algorithm 5 converges faster than Algorithm 3.

Remark 10. It is worth mentioning that the proposed SPUAs (both Algorithms 4 and 5) are essentially Newton's methods, which are not global approaches. Thus, they need a suitable choice of initial matrix. In a word, the SPUAs have the same advantages and disadvantages as the Newton's method. Similarly, the existing methods (i.e., Algorithms 2 and 3) also have the problem of suitably choosing initial matrices. In Algorithm 2, the famous Kleinman's method (i.e., Algorithm 1) was used to solve H_2 AREs (9) at each inner iterative loop, which was proven to be also a local approach in [15] and requires an initial stabilizing matrix. However, the work in [16] did not provide a method for finding initial stabilizing matrices at each inner iterative loop. This means that if not providing an initial stabilizing matrix for arbitrary inner iterative loop of Algorithm 2, it may fail to obtain the solution of ARE (7). As to Algorithm 3, we have conducted simulations, and the results have shown that

Algorithm 3 also failed to obtain the solution of ARE (7) when it is far from the initial matrix. Therefore, all of Algorithms 2–5 in our paper are valid only in neighborhood of the solution of the ARE (7). In fact, this problem also widely arises in many existing works for solving optimal or H_∞ control problems of either linear or nonlinear systems through the observation from computer simulations, such as [1–4,7–10,16,24–26,29,30,33]. Till present, it is still a difficult issue for finding proper initializations or developing global approaches. To overcome this difficult, we are intending to introduce some global schemes to attain this goal in future investigations.

3.3. Implement of online SPUA

The methods for computing $\boldsymbol{P}^{(i+1)}$ in online Algorithms 3 and 5 are similar. In this subsection, we use Algorithm 5 as an example to derive the implement approach (one can refer to [30] for similar method of optimal control problem) for online algorithms. The term $\boldsymbol{x}^T(t)\boldsymbol{P}^{(i+1)}\boldsymbol{x}(t)$ in (35) can be written as

$$\boldsymbol{x}^T(t)\boldsymbol{P}^{(i+1)}\boldsymbol{x}(t) = (\bar{\boldsymbol{p}}^{(i+1)})^T\bar{\boldsymbol{x}}(t) \tag{42}$$

where $\bar{\boldsymbol{x}}(t)$ denotes the Kronecker product quadratic polynomial basis vector with the elements $\{x_i(t)x_j(t)\}(i = 1,\ldots,n; j = i,\ldots,n)$, and $\bar{\boldsymbol{p}}^{(i+1)}$ denotes the a column vector by stacking the elements of the diagonal and upper triangular part of the symmetric matrix $\boldsymbol{P}$ into a vector where the off-diagonal elements are taken as $2p_{ij}$. Then, the Eq. (35) can be written as

$$(\bar{\boldsymbol{p}}^{(i+1)})^T(\bar{\boldsymbol{x}}(t) - \bar{\boldsymbol{x}}(t+\Delta t)) = \int_t^{t+\Delta t} \Xi^{(i)}(\tau)d\tau. \tag{43}$$

In (43), $\bar{\boldsymbol{p}}^{(i+1)}$ is a vector of unknown parameters, $\bar{\boldsymbol{x}}(t) - \bar{\boldsymbol{x}}(t+\Delta t)$ acts as a regression vector and the right hand of (43) is the target function. We notice that the symmetric matrix $\boldsymbol{P}^{(i+1)}$ has $n(n+1)/2$ unknown independent parameters. This means that, in order to solve for $\boldsymbol{P}^{(i+1)}$, we should construct at least $N(N \geqslant n(n+1)/2)$ equations. Therefore, in each time interval Δt, we should sample N state vectors. Let $\delta t = \Delta t/N$, and $\bar{\boldsymbol{x}}(t+k\delta t)(k = 0,\ldots,N-1)$ be the Kronecker product quadratic polynomial basis vector of the sampled state vector at time $t + k\delta t$. Then, $\boldsymbol{P}^{(i+1)}$ can be found in the least-square (LS) sense as follows:

$$\bar{\boldsymbol{p}}^{(i+1)} = (\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{Y} \tag{44}$$

where

$$\boldsymbol{X} = [\,(\bar{\boldsymbol{x}}(t) - \bar{\boldsymbol{x}}(t+\delta t)) \quad \cdots \quad (\bar{\boldsymbol{x}}(t+(N-1)\delta t)) - \bar{\boldsymbol{x}}(t+N\delta t)\,],$$

$$\boldsymbol{Y} = \begin{bmatrix} y_0^{(i)} & \cdots & y_{N-1}^{(i)} \end{bmatrix}^T$$

with $y_k^{(i)} = \int_{t+k\delta t}^{t+(k+1)\delta t} \Xi^{(i)}(\tau)d\tau, k = 0,\ldots,N-1$.

Clearly, the LS method (44) requires a nonsingular matrix $\boldsymbol{X}\boldsymbol{X}^T$. To attain the goal, we can inject probing noises into inputs or reset system states.

Remark 11. It should be pointed out that the proposed SPUAs (i.e., Algorithms 4 and 3) need to give an initial matrix $\boldsymbol{P}^{(0)}$ in $\mathbb{P}_0$, which maybe unavailable in some cases. For the offline SPUA (i.e., Algorithm 4), since the system model is completely known, a practical method is to select the matrix $\boldsymbol{P}^{(0)}$ such that $\lambda(\overline{\boldsymbol{A}}_0) \in \mathbb{C}^-$. Thus, if the open-loop system (i.e., $\boldsymbol{u} = 0$ and $\boldsymbol{w} = 0$) is stable, we can simply select $\boldsymbol{P}^{(0)} = 0$. As to the online SPUA (i.e., Algorithm 5), we can simply select $\boldsymbol{P}^{(0)} = \beta\boldsymbol{I}$, where $\beta \geqslant 0$ is some given scalar. Initially, we can run Algorithm 5 with $\beta = 0$. If the online SPUA does not converge to a positive-definite matrix solution, then, increase β gradually until the online SPUA converges to a positive-definite matrix solution. It should be pointed out that the methods presented here for the choice of $\boldsymbol{P}^{(0)}$ in SPUAs are on the basis of experience, and this issue will also be pursued in future work.

4. Comparative simulation studies

To test the efficiency of the two proposed SPUAs, we use two examples for comparative simulation, i.e., an F-16 aircraft plant and a power system. For each example, the following two cases are considered:

- **Case 1**: The system model is completely known. In this case, comparative studies are conducted with two offline algorithms, i.e., Algorithms 2 and 4 are used to solve the ARE.
- **Case 2**: The internal system dynamics is unknown. In this case, comparative studies are conducted with two online algorithms, i.e., Algorithms 3 and 5 are used to online learn the solution of the ARE.

It is noted that both Algorithms 2 and 3 require an initial zero matrix. In order to be fair, we also select the same initial matrices for our algorithms (i.e., Algorithms 4 and 5).

4.1. F-16 aircraft plant

Consider an example of the following F-16 aircraft plant that studied in [21,26]:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -1 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} w + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \tag{45}$$

where the system state vector is $\boldsymbol{x} = [\alpha \quad q \quad \delta_e]^T$, α denotes the angle of attack, q is the pitch rate and δ_e is the elevator deflection angle. The control input u is the elevator actuator voltage and the disturbance w is wind gusts on angle of attack.

Select the matrices $\boldsymbol{C}$ and $\boldsymbol{D}$ in the output vector so that the matrices $\boldsymbol{Q} = \boldsymbol{C}^T\boldsymbol{C}$ and $\boldsymbol{R} = \boldsymbol{D}^T\boldsymbol{D}$ in the cost function are identity matrices, and $\gamma = 5$. Solve the associated ARE (7) with the MATLAB command CARE, we obtain

$$\boldsymbol{P} = \begin{bmatrix} 1.6573 & 1.3954 & -0.1661 \\ 1.3954 & 1.6573 & -0.1804 \\ -0.1661 & -0.1804 & 0.4371 \end{bmatrix}. \tag{46}$$

4.1.1. Simulation for Case 1

The parameters in Algorithms 2 and 4 are given as follows. The value of ε_1 in Algorithm 2 is selected as 10^{-5}, and the value of stop criterion ε in Algorithms 2 and 4 is set as 10^{-7}.

Note that the matrix $\boldsymbol{P}$ in the associated ARE is a 3×3 symmetrical matrix, and thus it has 6 different parameters, i.e., p_{ij}, $(i,j \in \{1,2,3\}, i \leqslant j)$, where p_{ij} denotes the (i,j)th block of $\boldsymbol{P}$. By using Algorithms 2 and 4 to solve the ARE, Fig. 1 shows these parameters in each iterative step, where the dash lines represent their true values, the star markers denote the parameters obtained by Algorithm 2, the circle markers denote the parameters obtained by Algorithm 4, and i denotes the iterative steps (i.e., the times of solving the associated LE (10) in Algorithm 2, or the associated LE (17) in Algorithm 4). It can be seen from Fig. 1 that both Algorithms 2 and 4 converge to the solution of the associated ARE (7). However, Algorithm 4 achieves a faster convergence than Algorithm 2. Moreover, Algorithm 2 terminates at the 14th iterative step (i.e., Algorithm 2 obtains the solution of ARE with a precision of ε at the 14th iterative step), while Algorithm 4 is convergent at the 5th iterative step.

Remark 12. At each iterative step, both Algorithms 2 and 4 solve an LE (i.e., the Eqs. (10) and (17), respectively), thus, it is fair to compare Algorithms 2 and 4 based on the times for solving LEs.

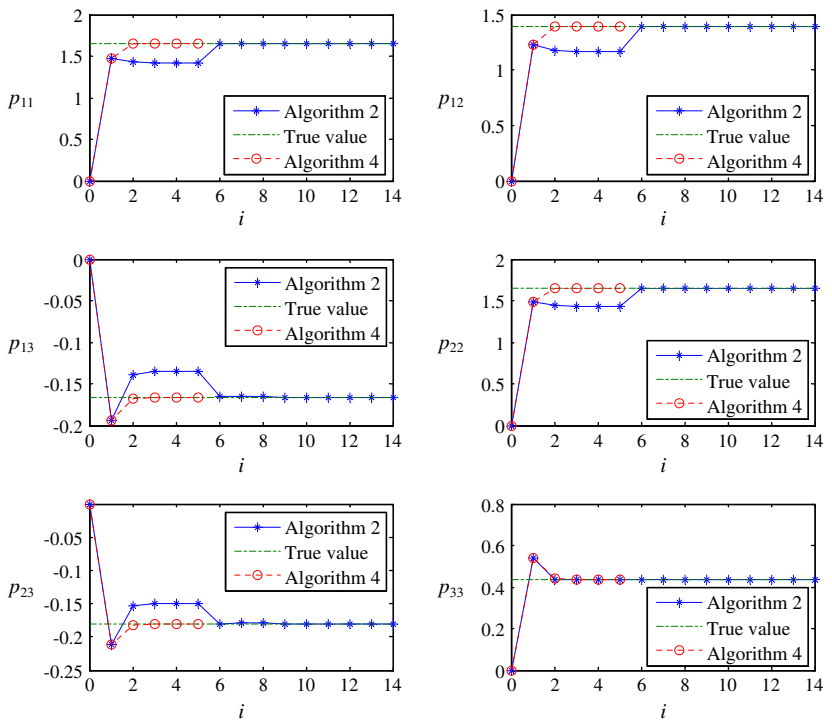

Fig. 1. The parameters p_{ij}, $(i,j \in \{1,2,3\}, i \leqslant j)$ obtained by Algorithms 2 and 4 for the F-16 aircraft plant.

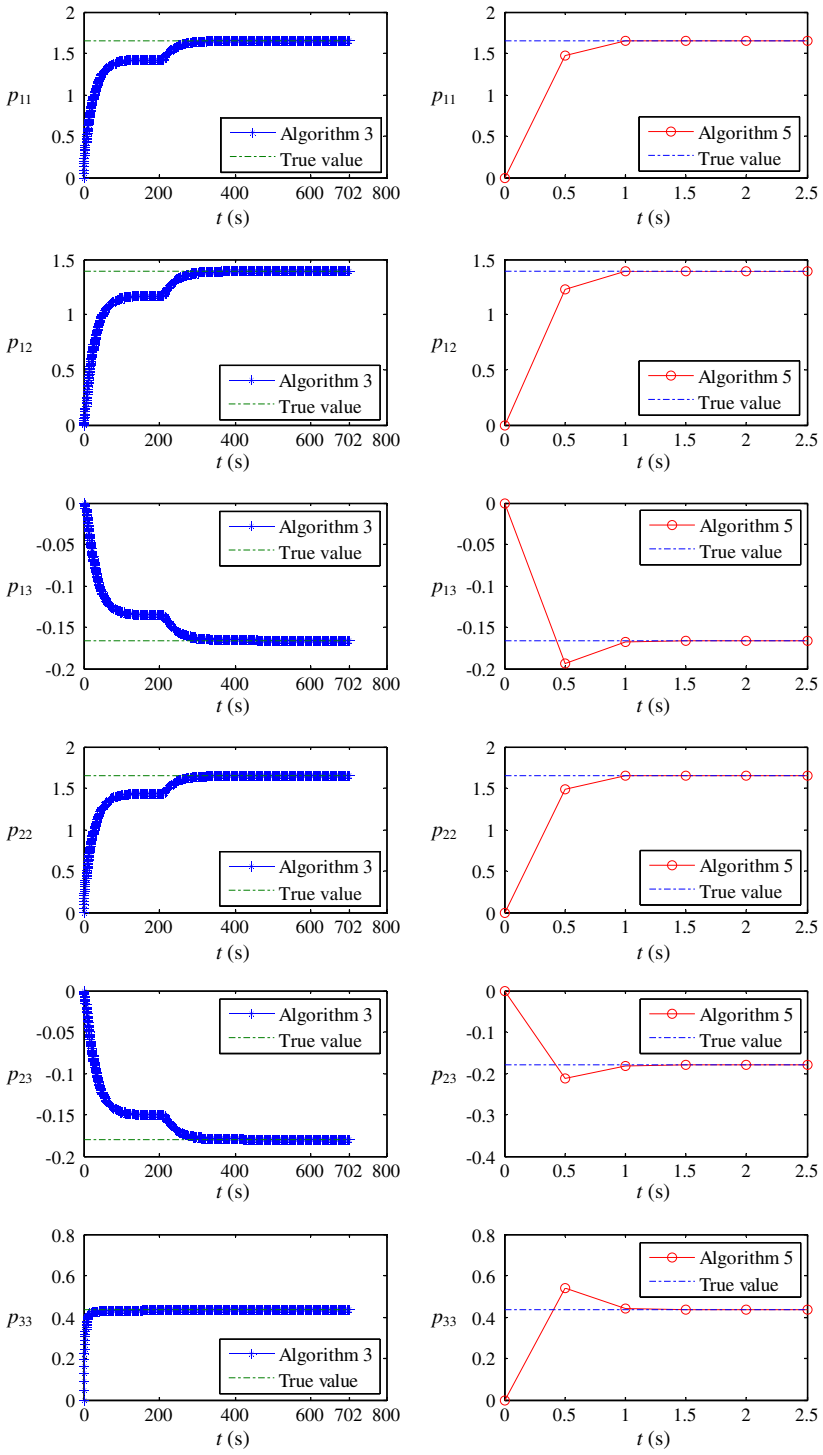

Fig. 2. The parameters p_{ij}, $(i,j \in \{1,2,3\}, i \leqslant j)$ obtained by Algorithms 3 and 5 for the F-16 aircraft plant.

4.1.2. Simulation for Case 2

The selection of parameters in Algorithms 3 and 5 are given as follows. Select the value of ε_1 in Algorithm 3 as 10^{-5}, the value of stop criterion ε in Algorithms 3 and 5 as 10^{-7}, initial system state as $\boldsymbol{x}_0 = [0.1 \quad 0.1 \quad 0.1]^T$, update time for matrix $\boldsymbol{P}$ as $\Delta t = 0.5(\mathrm{s})$ and number of sampling states for the LS method (44) as $N = 10$.

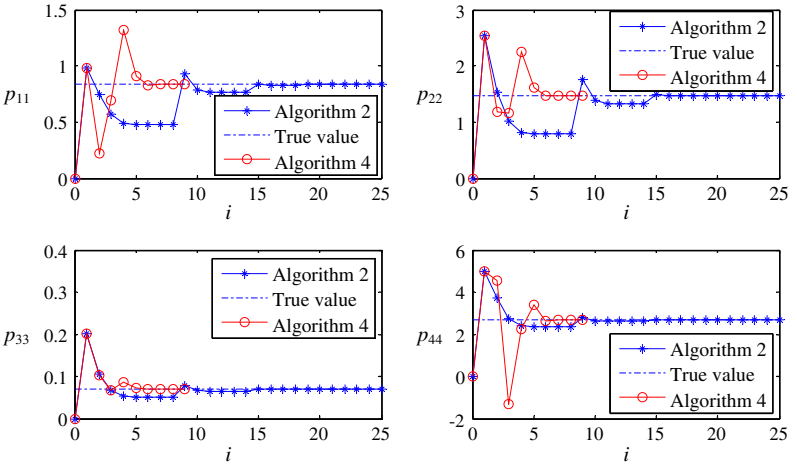

Fig. 3. The parameters p_{ii}, ($i \in \{1,2,3,4\}$) obtained by Algorithms 2 and 4 for the power system.

Remark 13. With the above parameters, Algorithms 3 and 5 sample states every 0.05 seconds, i.e., $\delta t = 0.05$(s). After collecting 10 system state measurements, the LS method (44) is used to solve Eqs. (14) and (35) for new matrices $\boldsymbol{P}$ in Algorithms 3 and 5. After each update, we reset the system state as initial state $\boldsymbol{x}_0$.

By applying Algorithms 3 and 5 for learning the solution of the associated ARE online, Fig. 2 shows the update of 6 different parameters $p_{ij}(i,j \in \{1,2,3\},\ i \leqslant j)$, where t denotes the time, the dash lines represent the true values of matrix parameters. From Fig. 2 and (46), we can see that both Algorithms 3 and 5 converge to the solution of the associated ARE (7). It is also obvious that the convergence of Algorithm 5 is significantly better than Algorithm 3. Algorithm 3 terminates at time $t = 702$ s (i.e., it obtains the solution of ARE with a precision of ε at $t = 702$ s), while Algorithm 5 stops at $t = 2.5$ s.

4.2. Power system

Consider an example of the following power system that studied in [29]:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -0.0665 & 8 & 0 & 0 \\ 0 & -3.663 & 3.663 & 0 \\ -6.86 & 0 & -13.736 & -13.736 \\ 0.6 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} -8 \\ 0 \\ 0 \\ 0 \end{bmatrix} w + \begin{bmatrix} 0 \\ 0 \\ 13.736 \\ 0 \end{bmatrix} u, \tag{47}$$

where the system state vector is $\boldsymbol{x} = \begin{bmatrix} \Delta f & \Delta P_g & \Delta X_g & \Delta E \end{bmatrix}^T$, Δf denotes the incremental frequency deviation, ΔP_g denotes the incremental change in generator output, ΔX_g is the incremental change in governor value position and ΔE is the incremental change in integral control.

Select the matrices $\boldsymbol{C}$ and $\boldsymbol{D}$ in the output vector so that $\boldsymbol{Q} = \boldsymbol{C}^T\boldsymbol{C} = \boldsymbol{I}$ and $\boldsymbol{R} = \boldsymbol{D}^T\boldsymbol{D} = \boldsymbol{I}$, and $\gamma = 3.5$. By using the MATLAB command CARE to solve the associated ARE (7), we obtain

$$\boldsymbol{P} = \begin{bmatrix} 0.8335 & 0.9649 & 0.1379 & 0.8005 \\ 0.9649 & 1.4751 & 0.2358 & 0.8046 \\ 0.1379 & 0.2358 & 0.0696 & 0.0955 \\ 0.8005 & 0.8046 & 0.0955 & 2.6716 \end{bmatrix}. \tag{48}$$

4.2.1. Simulation for Case 1

The parameters in Algorithms 2 and 4 are given as $\varepsilon_1 = 10^{-5}$, and $\varepsilon = 10^{-7}$. By using Algorithms 2 and 4 to solve the associated ARE (7), Fig. 3 gives the diagonal elements of matrix $\boldsymbol{P}$, i.e., p_{ii}, ($i \in \{1,2,3,4\}$), where the dash lines represent the true values, the star markers denote the parameters obtained by Algorithm 2, the circle markers denote the parameters obtained by Algorithm 4, and i denotes the iterative steps.

It is observed from Fig. 3 and (48) that Algorithm 2 converges to the solution of the associated ARE (7) with a precision of ε at the 25th iterative step, while Algorithm 4 is convergent at the 9th iterative step. This means that Algorithm 4 converges faster than Algorithm 2.

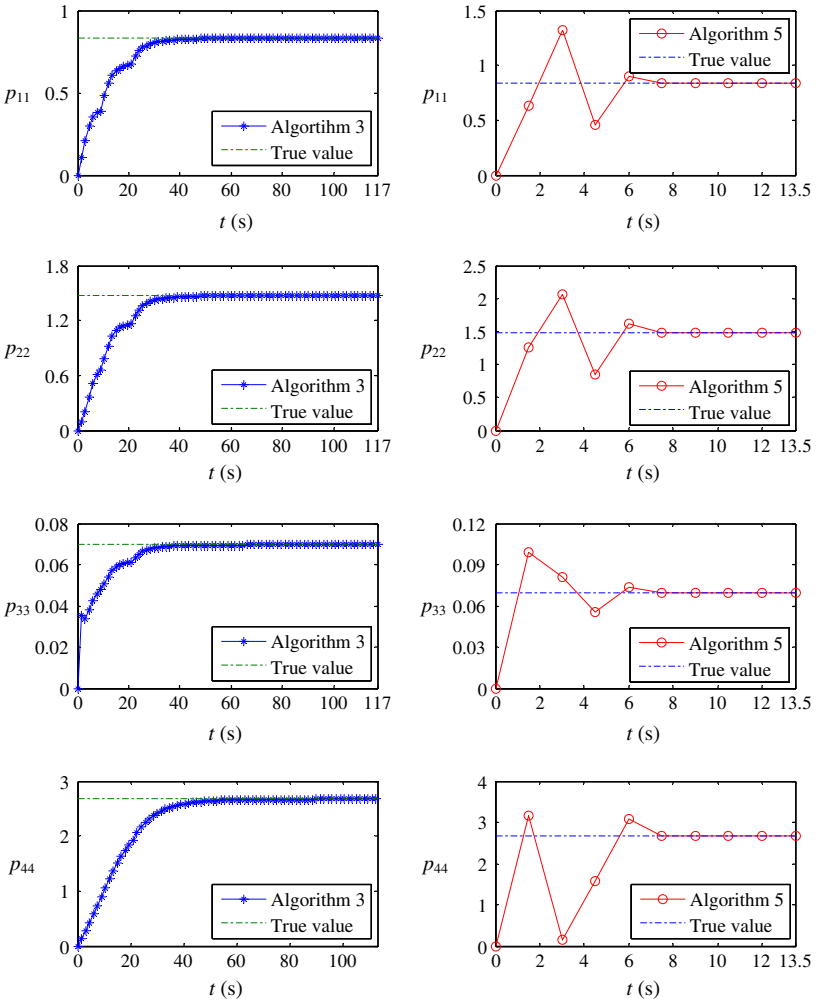

Fig. 4. The parameters p_{ii}, ($i \in \{1,2,3,4\}$) obtained by Algorithms 3 and 5 for the power system.

4.2.2. Simulation for Case 2

Select the value of ε_1 in Algorithm 3 as 10^{-5}, the value of stop criterion ε in Algorithms 3 and 5 as 10^{-7}, initial system state as $\boldsymbol{x}_0 = [0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T$, update time for matrix $\boldsymbol{P}$ as $\Delta t = 1.5$(s) and number of sampling states for the LS method (44) as $N = 15$. With the above parameters, Algorithms 3 and 5 sample states every 0.1 s, i.e., $\delta t = 0.1$(s). After collecting 15 system state measurements, the LS method (44) is used to solve equations (14) and (35) for new matrices $\boldsymbol{P}$ in Algorithms 3 and 5. After each update, the system state is reset as initial state $\boldsymbol{x}_0$.

By using Algorithms 3 and 5 to learn the solution of the associated ARE online, Fig. 4 shows the update of diagonal elements of matrix $\boldsymbol{P}$, i.e., p_{ii}, ($i \in \{1,2,3,4\}$), where t denotes the time, the dash lines represent the true values of matrix parameters. It can be seen from Fig. 4 that Algorithm 3 obtains the solution of the associated ARE (7) with a precision of ε at $t = 117$ s, while Algorithm 5 at $t = 13.5$ s. This implies that the convergence of Algorithm 5 is also much better than Algorithm 3 for the power system.

5. Conclusions

In this paper, we have introduced two methods to solve the ARE of the H_∞ control problem of continuous-time LTI system, the offline SPUA and the online SPUA. In both SPUAs, only one iterative loop is needed. We find that the offline SPUA is essentially a Newton's method to find a solution of a fixed point equation, thus, its convergence is proven with the help of the Kantorovtich's Theorem. Moreover, an online SPUA is also developed, which learns the solution of the ARE without requiring the knowledge of internal system dynamics. The convergence of the online SPUA guaranteed by showing that it is equivalent to the offline SPUA. Through the comparative studies with the existing methods on an F-16 aircraft plant and a power system, the results show that both offline SPUA and online SPUA can greatly improve the convergence.

Our future works on SPUA include: (1) find a more efficient method for the selection of initial matrix on $\mathbb{P}_0$, (2) extend the proposed SPUAs to H_∞ control of nonlinear systems.

Acknowledgments

This work was supported in part by the National Basic Research Program of China (973 Program) (2012CB720003); in part by the National Natural Science Foundation of China under Grants 61074057, 61121003, 61004023 and 91016004. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

[1] M. Abu-Khalaf, F.L. Lewis, J. Huang, Neurodynamic programming and zero-sum games for constrained control systems, IEEE Transactions on Neural Networks 19 (7) (2008) 1243–1252.
[2] R.W. Beard, T.W. Mclain, Successive Galerkin approximation algorithms for nonlinear optimal and robust control, International Journal of Control 71 (5) (1998) 717–743.
[3] R.W. Beard, G.N. Saridis, J. Wen, Galerkin approximations of the generalized Hamilton–Jacobi–Bellman equation, Automatica 33 (12) (1997) 2159–2177.
[4] R.W. Beard, G.N. Saridis, J. Wen, Approximate solutions to the time-invariant Hamilton–Jacobi–Bellman equation, Journal of Optimization Theory and Applications 96 (3) (1998) 589–626.
[5] D.P. Bertsekas, Approximate policy iteration a survey and some new methods, Journal of Control Theory and Applications 9 (3) (2011) 310–335.
[6] W. Cheney, Analysis for Applied Mathematics, Springer-Verlag, New York, 2001.
[7] V. Dragan, I.G. Ivanov, Computation of the stabilizing solution of game theoretic Riccati equation arising in stochastic H_∞ control problems, Numerical Algorithms 57 (3) (2011) 357–375.
[8] V. Dragan, I. Ivanov, A numerical procedure to compute the stabilising solution of game theoretic Riccati equations of stochastic control, International Journal of Control 84 (4) (2011) 783–800.
[9] Y.T. Feng, A. Varga, B.D.O. Anderson, M. Lovera, A new iterative algorithm to solve periodic Riccati differential equations with sign indefinite quadratic terms, IEEE Transactions on Automatic Control 56 (4) (2011) 929–934.
[10] Y.T. Feng, B.D.O. Anderson, An iterative algorithm to solve state-perturbed stochastic algebraic Riccati equations in LQ zero-sum games, Systems & Control Letters 59 (1) (2010) 50–56.
[11] M. Green, D.J.N. Limebeer, Linear Robust Control, Prentice-Hall, Englewood Cliffs, NJ, 1995.
[12] K.-S. Hwang, H.-Y. Lin, Y. -P Hsu, H.-H. Yu, Self-organizing state aggregation for architecture design of Q-learning, Information Sciences 181 (13) (2011) 2813–2822.
[13] K.-S. Hwang, Y.-J. Chen, W.-C. Jiang, T.-W. Yang, Induced states in a decision tree constructed by Q-learning, Information Sciences 213 (5) (2012) 39–49.
[14] L. Kantorovitch, The method of successive approximation for functional equations, Acta Mathematica 71 (1) (1939) 63–97.
[15] D.L. Kleinman, On an iterative technique for Riccati equation computations, IEEE Transactions on Automatic Control 13 (1) (1968) 114–115.
[16] A. Lanzon, Y. Feng, B.D.O. Anderson, M. Rotkowitz, Computing the positive stabilizing solution to algebraic Riccati equations with an indefinite quadratic term via a recursive method, IEEE Transactions on Automatic Control 53 (10) (2008) 2280–2291.
[17] F.L. Lewis, D. Vrabie, Reinforcement learning and adaptive dynamic programming for feedback control, IEEE Circuits and Systems Magazine 9 (3) (2009) 32–50.
[18] W.B. Powell, Approximate Dynamic Programming: Solving the Curses of Dimensionality, Wiley-Interscience, Hoboken, NJ, 2007.
[19] L.B. Rall, A note on the convergence of Newton's method, SIAM Journal on Numerical Analysis 11 (1) (1974) 34–36.
[20] G.N. Saridis, C.G. Lee, An approximation theory of optimal control for trainable manipulators, IEEE Transactions on Systems, Man and Cybernetics Part B-Cybernetics 9 (3) (1979) 152–159.
[21] B. Stevens, F.L. Lewis, Aircraft Control and Simulation, second ed., John Wiley, New Jersey, 2003.
[22] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, Mass, 1998.
[23] R.A. Tapia, The Kantorovich theorem for Newton's method, The American Mathematical Monthly 78 (4) (1971) 389–392.
[24] K.G. Vamvoudakis, F.L. Lewis, Online solution of nonlinear two-player zero-sum games using synchronous policy iteration, in: Proceedings of the 49th IEEE Conference on Decision and Control (CDC), 2010, pp. 3040–3047.
[25] K.G. Vamvoudakis, F.L. Lewis, Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem, Automatica 46 (5) (2010) 878–888.
[26] K.G. Vamvoudakis, Online Learning Algorithms for Differential Dynamic Games and Optimal Control, Ph.D dissertation, Faculty of the Graduate School, University of Texas at Arlington, 2011.
[27] A.J. Van Der Schaft, L_2-gain analysis of nonlinear systems and nonlinear state-feedback H_∞ control, IEEE Transactions on Automatic Control 37 (6) (1992) 770–784.
[28] N.A. Vien, H. Yu, T.C. Chung, Hessian matrix distribution for Bayesian policy gradient reinforcement learning, Information Sciences 181 (9) (2011) 1671–1685.
[29] D. Vrabie, F.L. Lewis, Adaptive dynamic programming for online solution of a zero-sum differential game, Journal of Control Theory and Applications 9 (3) (2011) 353–360.
[30] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, F.L. Lewis, Adaptive optimal control for continuous-time linear systems based on policy iteration, Automatica 45 (2) (2009) 477–484.
[31] F. Wang, H. Zhang, D. Liu, Adaptive dynamic programming: an introduction, IEEE Computational Intelligence Magazine 4 (2) (2009) 39–47.
[32] X. Wang, Y. Cheng, J. Yi, A fuzzy actor-critic reinforcement learning network, Information Sciences 177 (18) (2007) 3764–3781.
[33] H. Zhang, Q. Wei, D. Liu, An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games, Automatica 47 (1) (2011) 207–214.