Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics [⋆]

Yu Jiang [a], Zhong-Ping Jiang [a]

[a]*Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, NY 11201, U.S.A.*

Abstract

This paper presents a novel policy iteration approach for finding online adaptive optimal controllers for continuous-time linear systems with completely unknown system dynamics. The proposed approach employs the approximate/adaptive dynamic programming technique to iteratively solve the algebraic Riccati equation using the online information of state and input, without requiring the a priori knowledge of the system matrices. In addition, all iterations can be conducted by using repeatedly the same state and input information on some fixed time intervals. A practical online algorithm is developed in this paper, and is applied to the controller design for a turbocharged diesel engine with exhaust gas recirculation. Finally, several aspects of the future work are discussed.

Key words: Adaptive optimal control; Policy iterations; Linear-quadratic regulator (LQR).

1 Introduction

The adaptive controller design for unknown linear systems has been intensively studied in the past literature (*e.g.* Ionnaou & Sun, 1996; Mareels & Polderman, 1996; Tao, 2003). A conventional way to design an adaptive optimal control law can be pursued by identifying the system parameters first and then solving the related algebraic Riccati equation. However, adaptive systems designed this way are known to respond slowly to parameter variations from the plant.

Inspired by the learning behavior from biological systems, reinforcement learning (Sutton & Barto, 1998) and approximate/adaptive dynamic programming (ADP) (Werbos, 1974) theories have been broadly applied for solving optimal control problems for uncertain systems in recent years. See, for example, Lewis and Vrabie (2009) and Wang, Zhang, and Liu (2009) for two review papers, and Ferrari, Steck, and Chandramohan (2008), Al-Tamimi, Lewis, and Abu-Khalaf (2007), Bhasin, Sharma, Patre, and Dixon (2011), Dierks and Jagannathan (2011), Xu, Jagannathan, and Lewis (2011), Jiang and Jiang (2011), Vamvoudakis and Lewis (2011), Vrabie, Pastravanu, Abu-Khalaf, and Lewis (2009), Zhang, Wei, and Liu (2011), and Werbos (2009) for some recently developed results.

Among all the different ADP approaches, for discrete-time systems, the action-dependent heuristic dynamic programming (ADHDP) (Werbos, 1989), or Q-learning (Watkins, 1989), is an online iterative scheme that does not depend on the model to be controlled. Recently, the methodology has been extended and applied to many different areas, such as nonzero-sum games (Al-Tamimi et al., 2007), networked control systems (Xu et al., 2011), optimal output feedback control designs (Lewis & Vamvoudakis, 2011), as well as multi-objective optimal control problems (Wei, Zhang, & Dai, 2009).

Due to the different structures of the algebraic Riccati equations between discrete-time and continuous-time systems, results developed for the discrete-time setting cannot be directly applied for solving continuous-time problems. In Baird (1994), the advantage updating technique for sampled continuous-times system was proposed. In Murray, Cox, Lendaris, and Saeks (2002), two model-free algorithms were developed, but the measurements of state derivatives must be used. An exact method was developed in Vrabie et al. (2009), where neither knowing the internal system dynamics nor measuring the state derivatives was necessary. However, a common feature of all the existing ADP-based results is that partial knowledge of the system dynamics is assumed to be exactly known in the setting of continuous-time systems.

The primary objective of this paper is to remove this assump-

[⋆] This work has been supported in part by the National Science Foundation grants DMS-090665 and ECCS-1101401. This paper was not presented at any IFAC meeting. Corresponding author Y. Jiang. Tel. +1-718-260-3976.

Email addresses: yjiang06@students.poly.edu (Yu Jiang), zjiang@poly.edu (Zhong-Ping Jiang).

tion on partial knowledge of the system dynamics, and thus to develop a truly model-free ADP algorithm. More specifically, we propose a novel computational adaptive optimal control methodology that employs the approximate/adaptive dynamic programming technique to iteratively solve the algebraic Riccati equation using the online information of state and input, without requiring the a priori knowledge of the system matrices. In addition, all iterations can be conducted by using repeatedly the same state and input information on some fixed time intervals. It should be noticed that our approach may serve as a computational tool to study ADP related problems for continuous-time systems.

This paper is organized as follows: In Section 2, we briefly introduce the standard linear optimal control problem for continuous-time systems and the policy iteration technique. In Section 3, we develop our computational adaptive optimal control method and show its convergence. A practical online algorithm will be provided. In Section 4, we apply the proposed approach to the optimal controller design problem of a turbocharged diesel engine with exhaust gas recirculation. Concluding remarks as well as potential future extensions are contained in Section 5.

Notation: Throughout this paper, we use $\mathbb{R}$ and $\mathbb{Z}_+$ to denote the sets of real numbers and non-negative integers, respectively. Vertical bars $\|\cdot\|$ represent the Euclidean norm for vectors, or the induced matrix norm for matrices. We use $\otimes$ to indicate Kronecker product, and $\text{vec}(A)$ is defined to be the *mn*-vector formed by stacking the columns of $A \in \mathbb{R}^{n\times m}$ on top of one another, i.e., $\text{vec}(A) = [a_1^T \; a_2^T \; \cdots \; a_m^T]^T$, where $a_i \in \mathbb{R}^n$ are the columns of A. A control law is also called a *policy*. A feedback gain matrix $K \in \mathbb{R}^{m\times n}$ is said to be *stabilizing* for linear systems $\dot{x} = Ax + Bu$ if the feedback matrix $A - BK$ is Hurwitz.

2 Problem formulation and preliminaries

Consider a continuous-time linear system described by

$$\dot{x} = Ax + Bu \tag{1}$$

where $x \in \mathbb{R}^n$ is the system state fully available for feedback control design; $u \in \mathbb{R}^m$ is the control input; $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are unknown constant matrices. In addition, the system is assumed to be stabilizable.

The design objective is to find a linear optimal control law in the form of

$$u = -Kx \tag{2}$$

which minimizes the following performance index

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{3}$$

where $Q = Q^T \geq 0$, $R = R^T > 0$, with $(A, Q^{1/2})$ observable. By linear optimal control theory (Lewis & Syrmos, 1995), when both A and B are accurately known, solution to this problem can be found by solving the following well-known algebraic Riccati equation (ARE)

$$A^T P + PA + Q - PBR^{-1}B^T P = 0. \tag{4}$$

By the assumptions mentioned above, (4) has a unique symmetric positive definite solution P^*. The optimal feedback gain matrix K^* in (2) can thus be determined by

$$K^* = R^{-1}B^T P^*. \tag{5}$$

Since (4) is nonlinear in P, it is usually difficult to directly solve P^* from (4), especially for large-size matrices. Nevertheless, many efficient algorithms have been developed to numerically approximate the solution of (4). One of such algorithms was developed in Kleinman (1968), and is introduced in the following:

Theorem 1 (Kleinman, 1968) *Let $K_0 \in \mathbb{R}^{m\times n}$ be any stabilizing feedback gain matrix, and let P_k be the symmetric positive definite solution of the Lyapunov equation*

$$(A - BK_k)^T P_k + P_k(A - BK_k) + Q + K_k^T R K_k = 0 \tag{6}$$

where K_k, with $k = 1, 2, \cdots$, are defined recursively by:

$$K_k = R^{-1}B^T P_{k-1}. \tag{7}$$

Then, the following properties hold:

1. *$A - BK_k$ is Hurwitz,*
2. *$P^* \leq P_{k+1} \leq P_k$,*
3. *$\lim_{k\to\infty} K_k = K^*$, $\lim_{k\to\infty} P_k = P^*$.*

In Kleinman (1968), by iteratively solving the Lyapunov equation (6), which is linear in P_k, and updating K_k by (7), solution to the nonlinear equation (4) is numerically approximated.

For the purpose of solving (6) without the knowledge of A, in Vrabie et al. (2009), (6) was implemented online by

$$\begin{aligned} & x^T(t)P_k x(t) - x^T(t+\delta t)P_k x(t+\delta t) \\ = & \int_t^{t+\delta t} \left(x^T Q x + u_k^T R u_k\right) d\tau \end{aligned} \tag{8}$$

where $u_k = -K_k x$ is the control input of the system on the time interval $[t, t+\delta t]$.

Since both x and u_k can be measured online, a symmetric solution P_k can be uniquely determined under certain persistent excitation (PE) condition (Vrabie et al. 2009). However, as we can see from (7), the exact knowledge of system matrix B is still required for the iterations. Also, to guarantee the

PE condition, the state may need to be reset at each iteration step, but this may cause technical problems for stability analysis of the closed loop system (Vrabie et al., 2009). An alternative way is to add exploration noise (*e.g.* Bradtke, Ydstie, & Barto, 1994; Vamvoudakis & Lewis, 2011; Al-Tamimi et al., 2007; Xu et al., 2011) such that $u_k = -K_k x + e$, with e the exploration noise, is used as the true control input in (8). As a result, P_k solved from (8) and the one solved from (6) are not exactly the same. In addition, after each time the control policy is updated, information of the state and input must be re-collected for the next iteration. This may slow down the learning process, especially for high-dimensional systems.

3 Computational adaptive optimal control design with completely unknown dynamics

In this section, we will present our new online learning strategy that does not rely on either A or B. First, we assume a stabilizing K_0 is known. Then, for each $k \in \mathbb{Z}_+$, we seek to solve a symmetric positive definite matrix P_k satisfying (6), and obtain a feedback gain matrix $K_{k+1} \in \mathbb{R}^{m\times n}$ using $K_{k+1} = R^{-1}B^T P_k$.

To this end, we rewrite the original system (1) as

$$\dot{x} = A_k x + B(K_k x + u) \tag{9}$$

where $A_k = A - BK_k$.

Then, along the solutions of (9), by (6) and (7) it follows that

$$\begin{aligned} & x(t+\delta t)^T P_k x(t+\delta t) - x(t)^T P_k x(t) \\ = & \int_t^{t+\delta t} \left[x^T(A_k^T P_k + P_k A_k)x + 2(u+K_k x)^T B^T P_k x\right] d\tau \\ = & -\int_t^{t+\delta t} x^T Q_k x \, d\tau + 2\int_t^{t+\delta t} (u+K_k x)^T R K_{k+1} x \, d\tau \end{aligned} \tag{10}$$

where $Q_k = Q + K_k^T R K_k$.

Remark 2 *Notice that in* (10)*, the term $x^T(A_k^T P_k + P_k A_k)x$ depending on the unknown matrices A and B is replaced by $-x^T Q_k x$, which can be obtained by measuring the state online. Also, the term $B^T P_k$ involving B is replaced by RK_{k+1}, in which K_{k+1} is treated as another unknown matrix to be solved together with P_k. Therefore,* (10) *plays an important role in separating the system dynamics from the iterative process. As a result, the requirement of the system matrices in (6) and (7) can be replaced by the state and input information measured online.*

Remark 3 *It is also noteworthy that in* (10) *we always have exact equality if P_k, K_{k+1} satisfy (6), (7), and x is the solution of system (9) with arbitrary control input u. This fact enables us to employ $u = -K_0 x + e$, with e the exploration noise, as the input signal for learning, without affecting the convergence of the learning process.*

Next, we show that given a stabilizing K_k, a pair of matrices (P_k, K_{k+1}), with $P_k = P_k^T > 0$, satisfying (6) and (7) can be uniquely determined without knowing A or B , under certain condition. To this end, we define the following two operators:

$$P \in \mathbb{R}^{n\times n} \to \hat{P} \in \mathbb{R}^{\frac{1}{2}n(n+1)}, \text{ and } x \in \mathbb{R}^n \to \bar{x} \in \mathbb{R}^{\frac{1}{2}n(n+1)}$$

where

$$\begin{aligned} \hat{P} &= [p_{11}, 2p_{12}, \cdots, 2p_{1n}, p_{22}, 2p_{23}, \cdots, 2p_{n-1,n}, p_{nn}]^T, \\ \bar{x} &= [x_1^2, x_1x_2, \cdots, x_1x_n, x_2^2, x_2x_3, \cdots, x_{n-1}x_n, x_n^2]^T. \end{aligned}$$

In addition, by Kronecker product representation, we have

$$x^T Q_k x = \left(x^T \otimes x^T\right) \ \mathrm{vec}(Q_k),$$

and

$$\begin{aligned} & (u+K_k x)^T R K_{k+1} x \\ = & \left[(x^T \otimes x^T)(I_n \otimes K_k^T R) + (x^T \otimes u^T)(I_n \otimes R)\right] \mathrm{vec}(K_{k+1}). \end{aligned}$$

Further, for positive integer l, we define matrices $\delta_{xx} \in \mathbb{R}^{l\times\frac{1}{2}n(n+1)}$, $I_{xx} \in \mathbb{R}^{l\times n^2}$, $I_{xu} \in \mathbb{R}^{l\times mn}$, such that

$$\begin{aligned} \delta_{xx} &= \left[\ \bar{x}(t_1) - \bar{x}(t_0),\ \bar{x}(t_2) - \bar{x}(t_1),\ \cdots,\ \bar{x}(t_l) - \bar{x}(t_{l-1})\ \right]^T, \\ I_{xx} &= \left[\ \textstyle\int_{t_0}^{t_1} x\otimes x \, d\tau,\ \int_{t_1}^{t_2} x\otimes x \, d\tau,\ \cdots,\ \int_{t_{l-1}}^{t_l} x\otimes x \, d\tau\ \right]^T, \\ I_{xu} &= \left[\ \textstyle\int_{t_0}^{t_1} x\otimes u \, d\tau,\ \int_{t_1}^{t_2} x\otimes u \, d\tau,\ \cdots,\ \int_{t_{l-1}}^{t_l} x\otimes u \, d\tau\ \right]^T, \end{aligned}$$

where $0 \le t_0 < t_1 < \cdots < t_l$.

Then, for any given stabilizing gain matrix K_k, (10) implies the following matrix form of linear equations

$$\Theta_k \begin{bmatrix} \hat{P}_k \\ \mathrm{vec}\,(K_{k+1}) \end{bmatrix} = \Xi_k \tag{11}$$

where $\Theta_k \in \mathbb{R}^{l\times[\frac{1}{2}n(n+1)+mn]}$ and $\Xi_k \in \mathbb{R}^l$ are defined as:

$$\begin{aligned} \Theta_k &= \left[\delta_{xx}, -2I_{xx}(I_n \otimes K_k^T R) - 2I_{xu}(I_n \otimes R)\right], \\ \Xi_k &= -I_{xx} \ \mathrm{vec}(Q_k). \end{aligned}$$

Notice that if Θ_k has full column rank, (11) can be directly solved as follows:

$$\begin{bmatrix} \hat{P}_k \\ \mathrm{vec}\,(K_{k+1}) \end{bmatrix} = (\Theta_k^T \Theta_k)^{-1}\Theta_k^T \Xi_k. \tag{12}$$

Now, we are ready to give the following computational adaptive optimal control algorithm for practical online implementation. A flowchart of Algorithm 1 is shown in Figure 1.

Algorithm 1 *(Computational adaptive optimal control algorithm)*

1. *Employ $u = -K_0 x + e$ as the input on the time interval $[t_0,\ t_l]$, where K_0 is stabilizing and e is the exploration noise. Compute δ_{xx}, I_{xx} and I_{xu} until the rank condition in (13) below is satisfied. Let $k = 0$.*
2. *Solve P_k and K_{k+1} from (12).*
3. *Let $k \leftarrow k+1$, and repeat Step 2 until $\|P_k - P_{k-1}\| \leq \varepsilon$ for $k \geq 1$, where the constant $\varepsilon > 0$ is a predefined small threshold.*
4. *Use $u = -K_k x$ as the approximated optimal control policy.*

Remark 4 *Computing the matrices I_{xx} and I_{xu} carries the main burden in performing Algorithm 1. The two matrices can be implemented using $\frac{1}{2}n(n+1)+mn$ integrators in the learning system to collect information of the state and the input.*

Remark 5 *In practice, numerical error may occur when computing I_{xx} and I_{xu}. As a result, the solution of (11) may not exist. In that case, the solution of (12) can be viewed as the least squares solution of (11).*

Next, we show that the convergence of Algorithm 1 can be guaranteed under certain condition. The proof of the following lemma is postponed to the Appendix.

Lemma 6 *If there exists an integer $l_0 > 0$, such that, for all $l \geq l_0$,*

$$\text{rank}\left(\begin{bmatrix} I_{xx}, & I_{xu} \end{bmatrix}\right) = \frac{n(n+1)}{2} + mn, \tag{13}$$

then Θ_k has full column rank for all $k \in \mathbb{Z}_+$.

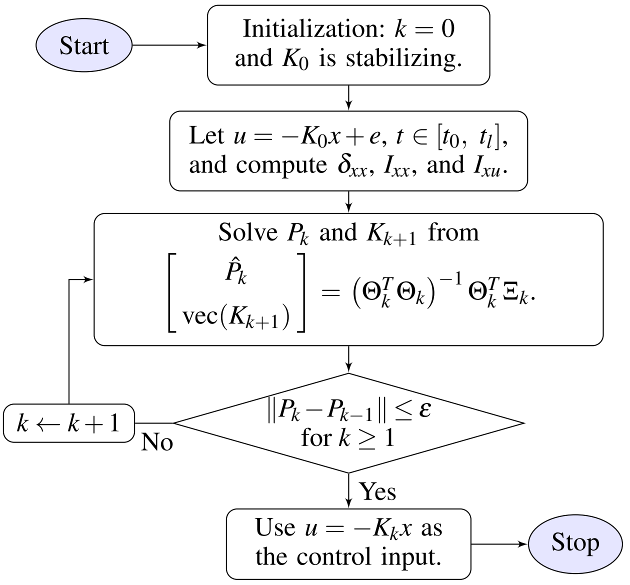
Start
Initialization: $k = 0$ and K_0 is stabilizing.
Let $u = -K_0 x + e$, $t \in [t_0,\ t_l]$, and compute δ_{xx}, I_{xx}, and I_{xu}.
Solve P_k and K_{k+1} from
$\begin{bmatrix} \hat{P}_k \\ \text{vec}(K_{k+1}) \end{bmatrix} = \left(\Theta_k^T \Theta_k\right)^{-1} \Theta_k^T \Xi_k.$
$\|P_k - P_{k-1}\| \leq \varepsilon$ for $k \geq 1$
No
$k \leftarrow k+1$
Yes
Use $u = -K_k x$ as the control input.
Stop

Fig. 1. Flowchart of Algorithm 1.

Theorem 7 *Starting from a stabilizing $K_0 \in \mathbb{R}^{m\times n}$, when the condition of Lemma 6 is satisfied, the sequences $\{P_i\}_{i=0}^{\infty}$ and $\{K_j\}_{j=1}^{\infty}$ obtained from solving (12) converge to the optimal values P^* and K^*, respectively.*

Proof: Given a stabilizing feedback gain matrix K_k, if $P_k = P_k^T$ is the solution of (6), K_{k+1} is uniquely determined by $K_{k+1} = R^{-1}B^T P_k$. By (10), we know that P_k and K_{k+1} satisfy (12). On the other hand, let $P = P^T \in \mathbb{R}^{n\times n}$ and $K \in \mathbb{R}^{m\times n}$, such that

$$\Theta_k \begin{bmatrix} \hat{P} \\ \text{vec}(K) \end{bmatrix} = \Xi_k.$$

Then, we immediately have $\hat{P} = \hat{P}_k$ and $\text{vec}(K) = \text{vec}(K_{k+1})$. By Lemma 6, $P = P^T$ and K are unique. In addition, by the definitions of $\hat{P}$ and $\text{vec}(K)$, $P_k = P$ and $K_{k+1} = K$ are uniquely determined.

Therefore, the policy iteration (12) is equivalent to (6) and (7). By Theorem 1, the convergence is thus proved. ■

Remark 8 *It can be seen that Algorithm 1 contains two separated phases: First, an initial stabilizing control policy with exploration noise is applied and the online information is recorded in matrices δ_{xx}, I_{xx}, and I_{xu} until the rank condition in (13) is satisfied. Second, without requiring additional system information, the matrices δ_{xx}, I_{xx}, and I_{xu} are repeatedly used to implement the iterative process. A sequence of controllers, that converges to the optimal control policy, can be obtained.*

Remark 9 *The choice of exploration noise is not a trivial task for general reinforcement learning problems and other related machine learning problems, especially for high dimensional systems. In solving practical problems, several types of exploration noise have been adopted, such as random noise (Al-Tamimi et al., 2007; Xu et al., 2011), exponentially decreasing probing noise (Vamvoudakis & Lewis, 2011). For the simulations in the next section, we will use the sum of sinusoidal signals with different frequencies, as in Jiang and Jiang (2011).*

Remark 10 *In some sense, our approach is related to the ADHDP (Werbos, 1989), or Q-learning (Watkins, 1989) method for discrete-time systems. Indeed, it can be viewed that we solve the following matrix H_k at each iteration step*

$$H_k = \begin{bmatrix} H_{11,k} & H_{12,k} \\ H_{21,k} & H_{22,k} \end{bmatrix} = \begin{bmatrix} P_k & P_k B \\ B^T P_k & R \end{bmatrix}. \tag{14}$$

Once this matrix is obtained, the control policy can be updated by $K_{k+1} = H_{22,k}^{-1} H_{21,k}$. The discrete-time version of the H_k matrix can be found in Bradtke et al. (1994) and Lewis and Vrabie (2009).

4 Application to a turbocharged diesel engine with exhaust gas recirculation

In this section, we study the controller design for a turbocharged diesel engine with exhaust gas recirculation (Jung, Glover, & Christen, 2005). The open loop model is a six-th order continuous-time linear system. The system matrices A and B are directly taken from Jung et al. (2005) and shown as follows:

$$A = \begin{bmatrix} -0.4125 & -0.0248 & 0.0741 & 0.0089 & 0 & 0 \\ 101.5873 & -7.2651 & 2.7608 & 2.8068 & 0 & 0 \\ 0.0704 & 0.0085 & -0.0741 & -0.0089 & 0 & 0.0200 \\ 0.0878 & 0.2672 & 0 & -0.3674 & 0.0044 & 0.3962 \\ -1.8414 & 0.0990 & 0 & 0 & -0.0343 & -0.0330 \\ 0 & 0 & 0 & -359.0000 & 187.5364 & -87.0316 \end{bmatrix},$$

$$B = \begin{bmatrix} -0.0042 & -1.0360 & 0.0042 & 0.1261 & 0 & 0 \\ 0.0064 & 1.5849 & 0 & 0 & -0.0168 & 0 \end{bmatrix}^T.$$

In order to illustrate the efficiency of the proposed computational adaptive optimal control strategy, the precise knowledge of A and B is not used in the design of optimal controllers. Since the physical system is already stable, the initial stabilizing feedback gain can be set as $K_0 = 0$.

The weighting matrices are selected to be

$$Q = \mathrm{diag}\Big(\,1,\ \ 1,\ \ 0.1,\ \ 0.1,\ \ 0.1,\ \ 0.1\,\Big),\ R = I_2.$$

In the simulation, the initial values for the state variables are randomly selected around the origin. From $t = 0s$ to $t = 2s$, the following exploration noise is used as the system input

$$e = 100\sum_{i=1}^{100}\sin(\omega_i t) \tag{15}$$

where ω_i, with $i = 1, \cdots, 100$, are randomly selected from $[-500, 500]$.

State and input information is collected over each interval of $0.01s$. The policy iteration started at $t = 2s$, and convergence is attained after 16 iterations, when the stopping criterion $\|P_k - P_{k-1}\| \le 0.03$ is satisfied. The formulated controller is used as the actual control input to the system starting from $t = 2s$ to the end of the simulation. The trajectory of the Euclidean norm of all the state variables is shown in Figure 2. The system output variables $y_1 = 3.6x_6$ and $y_2 = x_4$, denoting the mass air flow (MAF) and the intake manifold absolute pressure (MAP) (Jung et al., 2005), are plotted in Figure 3.

The proposed algorithm gives the cost and the feedback gain matrices as shown below:

$$P_{15} = \begin{bmatrix} 127.5331 & 0.5415 & 16.8284 & 1.8305 & 1.3966 & 0.0117 \\ 0.5415 & 0.0675 & 0.0378 & 0.0293 & 0.0440 & 0.0001 \\ 16.8284 & 0.0378 & 18.8105 & -0.3317 & 4.1648 & 0.0012 \\ 1.8305 & 0.0293 & -0.3317 & 0.5041 & -0.1193 & -0.0001 \\ 1.3966 & 0.0440 & 4.1648 & -0.1193 & 3.3985 & 0.0004 \\ 0.0117 & 0.0001 & 0.0012 & -0.0001 & 0.0004 & 0.0006 \end{bmatrix},$$

$$K_{15} = \begin{bmatrix} -0.7952 & -0.0684 & -0.0725 & 0.0242 & -0.0488 & -0.0002 \\ 1.6511 & 0.1098 & 0.0975 & 0.0601 & 0.0212 & 0.0002 \end{bmatrix}.$$

By solving directly the algebraic Riccati equation (4), we obtain the optimal solutions:

$$P^* = \begin{bmatrix} 127.5325 & 0.5416 & 16.8300 & 1.8307 & 1.4004 & 0.0117 \\ 0.5416 & 0.0675 & 0.0376 & 0.0292 & 0.0436 & 0.0001 \\ 16.8300 & 0.0376 & 18.8063 & -0.3323 & 4.1558 & 0.0012 \\ 1.8307 & 0.0292 & -0.3323 & 0.5039 & -0.1209 & -0.0001 \\ 1.4004 & 0.0436 & 4.1558 & -0.1209 & 3.3764 & 0.0004 \\ 0.0117 & 0.0001 & 0.0012 & -0.0001 & 0.0004 & 0.0006 \end{bmatrix},$$

$$K^* = \begin{bmatrix} -0.7952 & -0.0684 & -0.0726 & 0.0242 & -0.0488 & -0.0002 \\ 1.6511 & 0.1098 & 0.0975 & 0.0601 & 0.0213 & 0.0002 \end{bmatrix}.$$

The convergence of P_k and K_k to their optimal values is illustrated in Figure 4.

Notice that if B is accurately known, the problem can also be solved using the method in Vrabie et al. (2009). However, that method requires a total learning time of $32s$ for 16 iterations, if the state and input information within $2s$ is collected for each iteration. In addition, the method in Vrabie et al. (2009) may need to reset the state at each iteration step, in order to satisfy the PE condition.

5 Conclusions and future work

A novel computational policy iteration approach for finding online adaptive optimal controllers for continuous-time linear systems with completely unknown system dynamics has been presented in this paper. This method solves the algebraic Riccati equation iteratively using system state and input information collected online, without knowing the system matrices. A practical online algorithm was proposed and has been applied to the control design for a turbocharged diesel engine with unknown parameters. The methodology developed in this paper may serve as a computational tool to study the adaptive optimal control of uncertain nonlinear systems.

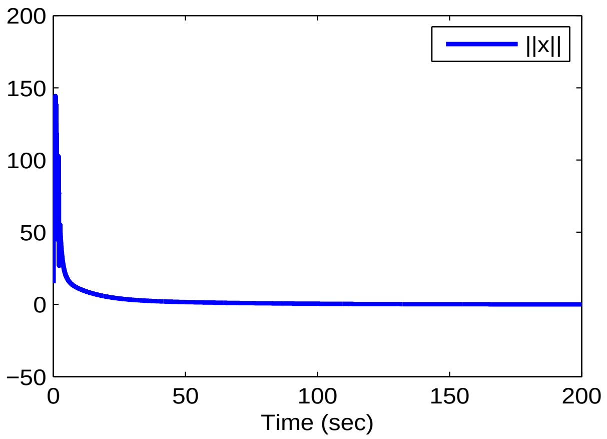

Fig. 2. Trajectory of the Euclidean norm of the state variables during the simulation.

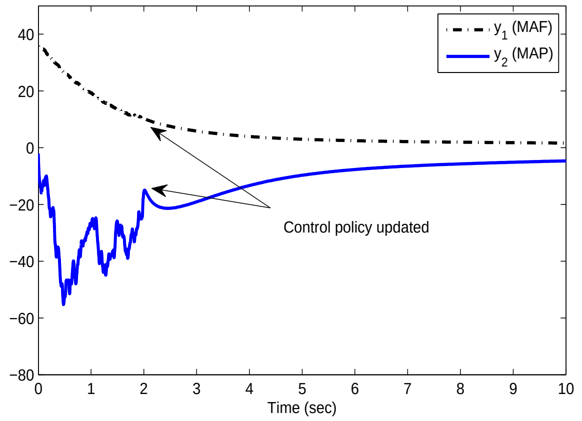

Fig. 3. Trajectories of the output variables from $t = 0s$ to $t = 10s$.

Some related work has appeared in Vrabie and Lewis (2009), Vamvoudakis and Lewis (2011), Vrabie and Lewis (2010), and also our recent work Jiang and Jiang (2011) proposing a framework of robust-ADP using nonlinear small-gain theorem (Jiang, Teel, & Praly, 1994).

Acknowledgements

The authors wish to thank Dr. Paul Werbos, Dr. Iven Mareels, Dr. Lei Guo, the Associate Editor and the reviewers for their constructive comments.

Appendix

A Proof of Lemma 6

It amounts to show that the following linear equation

$$\Theta_k X = 0 \tag{A.1}$$

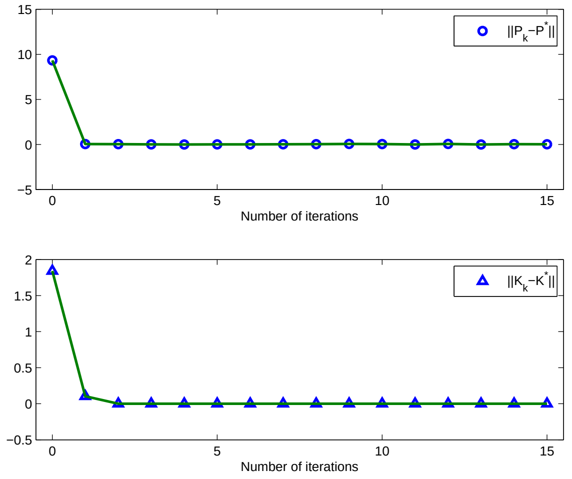

Fig. 4. Convergence of P_k and K_k to their optimal values P^* and K^* during the learning process.

has only the trivial solution $X = 0$.

To this end, we prove by contradiction. Assume $X = \begin{bmatrix} Y_v^T & Z_v^T \end{bmatrix}^T \in \mathbb{R}^{\frac{1}{2}n(n+1)+mn}$ is a nonzero solution of (A.1), where $Y_v \in \mathbb{R}^{\frac{1}{2}n(n+1)}$ and $Z_v \in \mathbb{R}^{mn}$. Then, a symmetric matrix $Y \in \mathbb{R}^{n\times n}$ and a matrix $Z \in \mathbb{R}^{m\times n}$ can be uniquely determined, such that $\hat{Y} = Y_v$ and $\text{vec}(Z) = Z_v$.

By (10), we have

$$\Theta_k X = I_{xx}\text{vec}(M) + 2I_{xu}\text{vec}(N) \tag{A.2}$$

where

$$M = A_k^T Y + Y A_k + K_k^T(B^T Y - RZ) + (YB - Z^T R)K_k, \tag{A.3}$$
$$N = B^T Y - RZ. \tag{A.4}$$

Notice that since M is symmetric, we have

$$I_{xx}\text{vec}(M) = I_{\bar{x}}\hat{M} \tag{A.5}$$

where $I_{\bar{x}} \in \mathbb{R}^{l\times\frac{1}{2}n(n+1)}$ is defined as:

$$I_{\bar{x}} = \left[\int_{t_0}^{t_1} \bar{x}d\tau,\ \int_{t_1}^{t_2} \bar{x}d\tau,\ \cdots,\ \int_{t_{l-1}}^{t_l} \bar{x}d\tau \right]^T. \tag{A.6}$$

Then, (A.1) and (A.2) imply the following matrix form of linear equations

$$\begin{bmatrix} I_{\bar{x}}, & 2I_{xu} \end{bmatrix} \begin{bmatrix} \hat{M} \\ \text{vec}(N) \end{bmatrix} = 0. \tag{A.7}$$

Under the rank condition in (13), we know $\begin{bmatrix} I_{\bar{x}}, & 2I_{xu} \end{bmatrix}$ has full column rank. Therefore, the only solution to (A.7) is

$\hat{M} = 0$ and $\text{vec}(N) = 0$. As a result, we have $M = 0$ and $N = 0$.

Now, by (A.4) we know $Z = R^{-1}B^TY$, and (A.3) is reduced to the following Lyapunov equation

$$A_k^TY + YA_k = 0. \tag{A.8}$$

Since A_k is Hurwitz for all $k \in \mathbb{Z}_+$, the only solution to (A.8) is $Y = 0$. Finally, by (A.4) we have $Z = 0$.

In summary, we have $X = 0$. But it contradicts with the assumption that $X \neq 0$. Therefore, Θ_k must have full column rank for all $k \in \mathbb{Z}_+$. The proof is complete. ■

References

[1] Al-Tamimi, A., Lewis, F. L., & Abu-Khalaf, M. (2007). Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. *Automatica*, 43(3), 473-481.

[2] Baird, L. C. III. (1994). Reinforcement learning in continuous time: Advantage updating. In: *Proc. of IEEE International Conference on Neural Networks* (pp. 2448-2453).

[3] Bhasin, S., Sharma, N., Patre, P., & Dixon, W. E. (2011). Asymptotic tracking by a reinforcement learning-based adaptive critic controller. *Journal of Control Theory and Applications*, 9(3), 400-409.

[4] Bradtke, S. J., Ydstie, B. E., & Barto, A. G. (1994). Adaptive linear quadratic control using policy iteration. In: *Proc. of American Control Conference*, (pp. 3475-3479).

[5] Dierks, T. & Jagannathan, S. (2011). Online optimal control of nonlinear discrete-time systems using approximate dynamic programming. *Journal of Control Theory and Applications*, 9(3), 361–369.

[6] K. DOYA, Reinforcement learning in continuous time and space, *Neural computation*, 12(2000), pp. 219–245.

[7] Ferrari, S., Steck, J. E., & Chandramohan, R. (2008). Adaptive feedback control by constrained approximate dynamic programming. *IEEE Transcations on System, Man, and Cybernetics, Part B: Cybernetics*, 38(4), 982-987.

[8] Ge, S. S., Lee, T. H., & Harris, C. J. (1998). *Adaptive neural network control of robotic manipulators*. World Scientific Pub. Co. Inc.

[9] Ioannou, P. A., & Sun, J. (1996). *Robust Adaptive Control*. Upper Saddle River, NJ: Prentice-Hall.

[10] Jiang, Y., & Jiang, Z. P. (2011). Robust approximate dynamic programming and global stabilization with nonlinear dynamic uncertainties. In: *Proc. of the Joint IEEE Conference on Decision and Control and European Control Conference* (pp. 115-120).

[11] Jiang, Z. P., Teel, A. R., & Praly, L. (1994). Small-gain theorem for ISS systems and applications. *Mathematics of Control, Signals, and Systems*, 7(2), 95-120.

[12] Jung, M., Glover, K., & Christen, U. (2005). Comparison of uncertainty parameterisations for H_∞ robust control of turbocharged diesel engines. *Control Engineering Practice*, 13, 15-25.

[13] Kleinman, D. (1968). On an iterative technique for Riccati equation computations. *IEEE Transactions on Automatic Control*, 13(1), 114-115.

[14] Krstić, M., Kanellakopoulos, I., & Kokotović, P. V. (1995). *Nonlinear and Adaptive Control Design*. John Wiley.

[15] Lewis, F. L., & Syrmos, V. L. (1995). *Optimal Control*. Wiley.

[16] Lewis, F. L., & Vrabie, D. (2009). Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Transactions on Circuits and Systems Magazine*, 9(3), 32-50.

[17] Lewis, F. L., & Vamvoudakis, K. G. (2011). Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(1), 14-23.

[18] Mareels, I., & Polderman, J. W. (1996). *Adaptive Systems: An Introduction*. Birkhäuser.

[19] Murray, J. J., Cox, C. J. Lendaris, G. G., & Saeks, R. (2002). Adaptive dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 32(2), 140-153.

[20] Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

[21] Tao, G. (2003). *Adaptive Control Design and Analysis*. Wiley.

[22] Vamvoudakis, K. G., & Lewis, F. L. (2010). Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, 46(5), 878–888.

[23] Vamvoudakis, K. G., & Lewis. F. L. (2011). Multi-player non-zero-sum games: Online adaptive learning solution of coupled Hamilton-Jacobi equations. *Automatica*, 47(8), 1556-1569.

[24] Vrabie, D., Pastravanu, O., Abu-Khalaf, M., & Lewis, F. L. (2009). Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica*, 45(2), 477-484.

[25] Vrabie, D., & Lewis, F. L. (2009). Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems, *Neural Networks*, 22(3), 237-246.

[26] Vrabie, D., & Lewis, F. (2010). Adaptive dynamic programming algorithm for finding online the equilibrium solution of the two-player zero-sum differential game. In: *Proc. of IEEE Joint Conference on Neural Networks* (pp. 1-8).

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

[28] Watkins, C. (1989). *Learning from delayed rewards. Ph.D. thesis.* King's College of Cambridge.

[29] Wei, Q., Zhang, H., & Dai, J. (2009). Model-free multiobjective approximate dynamic programming for discrete-time nonlinear systems with general performance index functions. *Neurocomputing*, 72(7-9), 1839-1848.

[30] Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioural sciences. Ph.D. thesis.* Harvard University.

[31] Werbos, P. J. (1989). Neural networks for control and system identification. In: *Proc. of IEEE Conference on Decision and Control* (pp. 260-265).

[32] Werbos, P. J. (1998). Stable adaptive control using new critic designs. *Arxiv preprint adap-org/9810001.*

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

[34] Xu, H., Jagannathan, S., & Lewis, F. L. (2011). Stochastic optimal control of unknown linear networked control system in the presence of random delays and packet losses. *Automatica*, in press.

[35] Zhang, H., Wei, Q., & Liu, D. (2011). An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games. *Automatica*, 47(1), 207-214.