Journal of Mathematical Modeling
Vol. 14, No. 1, 2026, pp. 1-17. Research Article

JMM

Dynamic adaptation strategies for optimal control in unknown linear time-invariant system

Homa Pouyanfar, Sohrab Effati*, Amin Mansoori

Department of Applied Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, P. O. Box 1159, Mashhad 91775, Iran

Email(s): homa.pouyanfar@gmail.com, s-effati@um.ac.ir, am.ma7676@yahoo.com

Abstract. This paper presents a framework for online adaptive optimal control of continuous-time linear systems with unknown dynamics. The approach uses approximate and adaptive dynamic programming to learn the optimal control policy and value function in real-time, without prior knowledge of the system matrices. We introduce two algorithms based on policy iteration and value iteration, providing proofs the convergence and stability. Our value iteration method is robust against from exploration noise. The effectiveness of these control strategies is demonstrated through two examples, highlighting their ability to achieve near-optimal performance despite unknown dynamics.

Keywords: Optimal control, adaptive dynamic programming, policy iteration, value iteration, exploration noise.
AMS Subject Classification 2010: 34A34, 65L05.

1 Introduction

In modern control theory, optimal control is an important subfield that focuses on developing controllers to optimize system performance. This process usually depends on the complete knowledge of system parameters. There are two main methods for solving optimal control problems: The Pontryagin method establishes the necessary conditions for optimality, whereas the dynamic programming method provides sufficient conditions. In dynamic programming, problems are usually solved backwards. For continuous-time (CT) systems, this methodology results in the Hamiltonian Jacobi Bellman (HJB) equation. In general, the traditional optimal control methods require precise knowledge of system dynamics to solve the algebraic Riccati equation (ARE) [12].

The study of linear quadratic optimal control problems within the context of linear systems has been ongoing for over fifty years. The application of Pontryagin's maximum principle to these optimal control problems, as detailed in [16] and [17], leads to a system of coupled two-point boundary value

*Corresponding author
Received: 4 April 2025 / Revised: 23 May 2025 / Accepted: 9 June 2025
DOI: 10.22124/JMM.2025.30264.2716

problems. In the realm of dynamic programming, the sufficient conditions for an optimal controller and the functional with prescribed derivatives proposed by [8] result in a set of partial differential equations known as the RE for these systems.

Assuming known system dynamics, the authors of [14] introduced an iterative approach to address linear time-delay optimal control problems. Moreover, neural networks (NN) have become a promising approach for researchers. In these methods, the optimal control problem is reformulated into a system of equations that can be resolved using established NN models, such as Perceptron models. In the work of Effati and Pakdaman [2], the authors focused on approximating the state, co-state and control functions that are essential for optimal control problems. In another study, Effati et al. [3] utilized artificial neural networks (ANNs) to address a linear optimal control problem defined by a quadratic cost functional with fuzzy variables. Additionally, the study referenced in [24] examined adaptive synchronization of two non-identical bidirectional associative memory NNs, which included time-varying delays and unknown parameters.

The design of adaptive controllers for unknown linear systems has been extensively explored in the literature (e.g., [5,21]). A traditional approach involves identifying system parameters before solving the associated ARE. However, this method often results in slow responses to parameter changes. Recently, inspired by biological learning behaviors, reinforcement learning (RL) and adaptive dynamic programming (ADP) have gained popularity for addressing optimal control problems in uncertain systems [20]. For continuous time linear systems, the authors of [22] proposed a policy iteration-based ADP method to solve the optimal control for partially unknown continuous time linear systems.

In this study, we aim to utilize RL techniques, such as policy iteration (PI) and value iteration (VI) algorithms, to address the optimal control problem. We leverage approximate and ADP to learn the optimal control policy and value function in real-time, without requiring any prior knowledge of the system matrices. Unlike the work in [6], which focuses solely on PI and necessitates a stabilizing initial control policy (K_0), our work introduces two distinct algorithms. The VI algorithm overcomes the limitation of requiring a stabilizing initial control policy, making it more practical in scenarios where finding such a K_0 is challenging or impossible.

The remainder of this paper is structured as follows: Section 2 formally states the problem. Section 3 details the proposed PI and VI algorithms, accompanied by the aforementioned convergence and stability proofs, as well as the trajectory error analysis and immunity of exploration noise. In Section 4, the effectiveness of the proposed algorithms is demonstrated through two illustrative examples. These examples are implemented using both PI and VI, and the convergence behavior and computational time are compared using figures and tables. Finally, Section 5 concludes the paper and discusses potential directions for future research.

Notations: Throughout this paper, we denote the set of real numbers by $\mathbb{R}$ and the set of positive integers by $\mathbb{Z}^+$. For a square matrix A, the notation $A \succ 0$ indicates that A is positive definite, while $A \succeq 0$ signifies that A is positive semi-definite. The transpose of matrix A is represented as A^T. For any given $n \in \mathbb{Z}^+$, $I_n \in \mathbb{R}^{n\times n}$ denotes the identity matrix. If A is a symmetric matrix, $\lambda_{\min}$ and $\lambda_{\max}$ represent its minimum and maximum eigenvalues, respectively. Given a matrix $D \in \mathbb{R}^{n\times m}$, the vectorization of D is defined as $\text{vec}(D) = [d_1^T, d_2^T, \ldots, d_m^T]^T \in \mathbb{R}^{m\times n}$. For a symmetric matrix $E \in \mathbb{R}^{m\times m}$, the vectorization is expressed as $\text{vecs}(E) = [e_{11}, 2e_{12}, \ldots, 2e_{1m}, e_{22}, 2e_{23}, \ldots, 2e_{m-1,m}, e_{mm}]^T \in \mathbb{R}^{\frac{1}{2}m\times(m+1)}$. For an arbitrary column vector $C \in \mathbb{R}^n$, we define $\text{vecv}(C) = [c_1^2, c_1c_2, \ldots, c_1c_n, c_2^2, c_2c_3, \ldots, c_{n-1}c_n, c_n^2]^T \in \mathbb{R}^{\frac{1}{2}n\times(n+1)}$. The space of all $n \times n$ real symmetric matrices is denoted by $\mathscr{D}^n$, and $\mathscr{D}_+^n = \{D \in \mathscr{D}^n : D \geq 0\}$ represents the

subset of non-negative matrices. The Kronecker product of matrices X and Y is denoted by $X \otimes Y$, and the Euclidean norm for vectors is indicated by $\|.\|$.

2 Problem formulation

To begin with, consider a class of CT linear system described by

$$\dot{x}(t) = Ax(t) + Bu(t),\ x(0) = x_0, \tag{1}$$

where $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ are the system state vector and the control input. $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are constant matrices of system. In the systems mentioned above, we assume that there is no knowledge about the dynamics of the systems denoted as (A,B), but they are controllable. The design objective is to find a linear optimal control law in the form of

$$u(t) = -\bar{K}x(t), \tag{2}$$

which minimizes the following performance index

$$J = \int_0^\infty \left(x^T(t)Qx(t) + u^T(t)Ru(t)\right)dt, \tag{3}$$

where $Q = Q^T \succeq 0$ and $R = R^T \succ 0$ are matrices with appropriate dimensions.

3 Analysis of optimal control problem solution

In the classical optimal control when the system information (A,B) is completely known, the optimal feedback gain matrix $\bar{K}$ in (2), can thus be determined by $\bar{K} = R^{-1}B^T\bar{P}$, where $\bar{P}$ is the solution of following well-known ARE

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

Due to the nonlinearity of equation (4) with respect to P, finding the solution $\bar{P}$ directly is often challenging, especially for large-scale matrices. However, various efficient algorithms have been developed to approximate the numerical solution of equation (4). One of these algorithms is known as Kleinman's algorithm [9], which is outlined below.

Lemma 1. *([9]) Let $K_0 \in \mathbb{R}^{m\times n}$ be a stabilizing feedback gain matrix and repeat the following steps for $j = 0,1,\ldots$*

1. *Solve for the real symmetric positive definite solution P_j of the Lyapunov equation*

$$A_j^TP + PA_j + Q + K_j^TRK_j = 0, \tag{5}$$

where $A_j = (A - BK_j)$.

2. *Update the feedback gain matrix by*

$$K_{j+1} = R^{-1}B^TP_j.$$

Then, the following properties hold:

1. $(A-BK_j)$ *is Hurwitz;*
2. $\bar{P} \preceq P_{j+1} \preceq P_j$;
3. $\lim_{k\to\infty} K_j = \bar{K}$, $\lim_{k\to\infty} P_j = \bar{P}$.

3.1 Policy iteration

In this subsection, we introduce an ADP method inspired by RL to develop online adaptive optimal controllers for unknown linear systems. Hence, exploration noise is incorporated into the input during parameter adaptation:

$$u_j(t) = -K_j x(t) + \eta, \tag{6}$$

where the exploration noise η is composed of trigonometric function noises of different frequencies and is added for on-line learning [6]. By substituting the control input (6) into the system (1), we can get

$$\dot{x}(t) = A_j x(t) + B\eta. \tag{7}$$

To begin, by taking the time derivative of $x^T(t)P_j x(t)$ and considering system (7) with equation (5), we have

$$\begin{aligned}\frac{d}{dt}(x^T(t)P_j x(t)) &= \dot{x}(t)^T P_j x(t) + x^T(t)P_j\dot{x}(t)\\ &= x^T(t)\big(A_j^T P_j + P_j A_j\big)x(t) + 2\big(u(t)+K_j x(t)\big)^T B^T P_j x(t)\\ &= -x^T(t)\big(Q + K_j^T R K_j\big)x(t) + 2\big(u(t)+K_j x(t)\big)^T R K_{j+1} x(t).\end{aligned} \tag{8}$$

Now, by integrating both sides of (8) over the time interval $[t, t+\delta t]$ it can be observed that

$$\begin{aligned}&x(t+\delta t)^T P_j x(t+\delta t) - x^T(t)P_j x(t)\\ &= \int_t^{t+\delta t}\Big[x^T(\tau)\big(A_j^T P_j + P_j A_j\big)x(\tau) + 2\big(u(\tau)+K_j x(\tau)\big)^T B^T P_j x(\tau)\Big]d\tau\\ &= -\int_t^{t+\delta t} x^T(\tau)\big(Q + K_j^T R K_j\big)x(\tau)d\tau + 2\int_t^{t+\delta t}\big(u(\tau)+K_j x(\tau)\big)^T R K_{j+1} x(\tau)d\tau.\end{aligned} \tag{9}$$

To simplify our computations, we aim to convert the unknown matrices P_j and K_{j+1} into vectors. A highly effective approach to accomplish this conversion without losing any information is by using the Kronecker product representation [10]. We rely on an important identity for this purpose:

$$\mathrm{vec}(XYZ) = (Z^T \otimes X)\mathrm{vec}(Y).$$

By Kronecker product representation, we obtain

$$\begin{aligned}x^T(t)\big(Q + K_j^T R K_j\big)x(t) &= \big(x^T(t)\otimes x^T(t)\big)\mathrm{vec}\big(Q + K_j^T R K_j\big),\\ \big(u(t)+K_j x(t)\big)^T R K_{j+1} x(t) &= \Big[\big(x^T(t)\otimes x^T(t)\big)\big(I_n \otimes K_j^T R\big) + \big(x^T(t)\otimes u(t)^T\big)\big(I_n \otimes R\big)\Big]\mathrm{vec}\big(K_{j+1}\big).\end{aligned}$$

Moreover, for positive integer r, define

$$\begin{aligned}
\mathtt{D}_{xx} &= \Big[\operatorname{vecv}\big(x(t_1)\big) - \operatorname{vecv}\big(x(t_0)\big), \operatorname{vecv}\big(x(t_2)\big) - \operatorname{vecv}\big(x(t_1)\big), \ldots, \operatorname{vecv}\big(x(t_r)\big) - \operatorname{vecv}\big(x(t_{r-1})\big)\Big], \\
\mathtt{T}_{xx} &= \Big[\int_{t_0}^{t_1} x^T(\tau)\otimes x^T(\tau)d\tau, \int_{t_1}^{t_2} x^T(\tau)\otimes x^T(\tau)d\tau, \ldots, \int_{t_{r-1}}^{t_r} x^T(\tau)\otimes x^T(\tau)d\tau\Big], \\
\mathtt{T}_{xu} &= \Big[\int_{t_0}^{t_1} x(\tau)^T\otimes u^T(\tau)d\tau, \int_{t_1}^{t_2} x^T(\tau)\otimes u^T(\tau)d\tau, \ldots, \int_{t_{r-1}}^{t_r} x^T(\tau)\otimes u^T(\tau)d\tau\Big],
\end{aligned}$$

where $t_0 < t_1 < \cdots < t_r$ are positive integers. So, (9) implies the following linear equation

$$\Omega_j \mathtt{F}_j = \Phi_j, \tag{11}$$

where

$$\Omega_j = \big[\mathtt{D}_{xx}, -2\mathtt{T}_{xx}\big(I_n\otimes K_j^T R\big) - 2\mathtt{T}_{xu}(I_n\otimes R)\big], \tag{12a}$$

$$\begin{aligned}
\mathtt{F}_j &= \big[\operatorname{vecs}(P_j)^T, \operatorname{vec}(K_{j+1})^T\big]^T, \\
\Phi_j &= -\mathtt{T}_{xx}\big[\operatorname{vec}(Q + K_j^T R K_j)\big].
\end{aligned} \tag{12b}$$

It is important to note that, the calculation of K_{j+1} and P_j after each iteration of the algorithm can be performed directly if Ω_j has full column rank:

$$\mathtt{F}_j = \Big(\big(\Omega_j\big)^T\Omega_j\Big)^{-1}\big(\Omega_j\big)^T\Phi_j. \tag{13}$$

By applying above equations to calculate the unique pair (P_j, K_{j+1}), we can obtain the optimal numerical solution for the matrix K_{j+1}. The process is summarized as Algorithm 1.

Algorithm 1: PI for Optimal-Feedback Controller

1: Find matrices $\bar{Q}$ and R with appropriate dimensions.
2: Set $j \leftarrow 0$ and choose a suitable output-feedback control gain K_0, where $(A - BK_0)$ is a stability matrix.
3: Implement $u_j(t) = -K_j x(t) + \eta$ on the system (1), construct the matrices Ω_j from equation (12a) and Φ_j from equation (12b) based on collected sampling data.
4: Compute the unique pair P_j and K_{j+1} from equation (13).
5: Update to $j \leftarrow j+1$ and iterate Steps 3-5 until the convergence condition $||K_j - K_{j+1}|| < \varepsilon$ is satisfied.
6: Terminate the exploration noise η and apply $u_{j+1}(t) = -K_{j+1}x(t)$, as the optimal control input.

Remark 1. *Computing the matrices* $\mathtt{T}_{xx}$ *and* $\mathtt{T}_{xu}$ *in Algorithm 1 can be challenging. It requires a total of* $\frac{1}{2}n(n+1) + mn$ *integrations to gather data on both the state and input variables. However, when calculating these matrices, numerical errors can arise, potentially leading to the non-existence of a solution for equation* (11). *In such situations, the solution to equation* (13) *can be interpreted as the least squares solution for equation* (11).

It is important to note that Algorithm 1 does not require any knowledge of the matrices A and B.

3.1.1 Convergence and stability analysis of Algorithm 1

In this section, we analyze the convergence and stability of Algorithm 1. The uniqueness of the solution to (13) is ensured by the following lemma.

Lemma 2. *If there exists an integer $r_0 > 0$, such that for all $r \geq r_0$, the following rank conditions hold*

$$rank([\mathtt{T}_{xx}, \mathtt{T}_{xu}]) = \frac{n(n+1)}{2} + mn, \tag{14}$$

then, the matrix Ω_j has full column rank for all $k \in \mathbb{Z}^+$. Subsequently equation (13), can be uniquely solved.

Proof. See [6] for the proof. □

The convergence of Algorithm 1 is proved under the rank condition established in Lemma 2.

Theorem 1. *The sequences $\{\mathtt{F}_j\}_{j=0}^{\infty}$ and $\{K_j\}_{j=1}^{\infty}$ generated by Algorithm 1 converge to $\mathtt{F}^*$ and $\bar{K}$ respectively, under the rank condition outlined in Lemma 2.*

Proof. Based on Lemmas 1 and 2, the proof is the same as [6]. □

The following theorem gives the stability of system (1).

Theorem 2. *Assuming that the matrix Q in performance index (3) satisfies $Q > \mu I_n$, where μ is a positive real number and the control policy $u^*(t) = -K_{j^*}x(t) + \eta$ is obtained using Algorithm 1, system (1) will generally be asymptotically stable at the origin if the following condition is met:*

$$\|K_j x(t)\|^2 \leq \frac{\theta\mu\|x(t)\|^2 + 2\lambda_{\min}(R)\|\eta - u(t)\|^2 - 2\|\eta^T x(t)\|B^T P_j}{\lambda_{\max}(R)}, \tag{15}$$

where the parameter $\theta \in (0,1)$.

Proof. Define the Lyapunov function $V(x(t)) = x^T(t)P_j x(t)$. Along with presenting solutions for the closed-loop system, we have

$$\begin{aligned}\frac{d}{dt}(x^T(t)P_j x(t)) &= \dot{x}^T(t)P_j x(t) + x^T(t)P_j\dot{x}(t)\\ &= x^T(t)\left[A_j^T P_j + P_j A_j\right]x(t) + 2\eta^T B^T P_j x(t).\end{aligned}$$

According to equation (5), we get

$$\frac{d}{dt}(x^T(t)P_j x(t)) = -x^T(t)\left[Q + K_j^T R K_j\right]x(t) + 2\eta^T B^T P_j x(t). \tag{16}$$

By adding and subtracting the term $x^T(t)K_j^T R K_j x(t)$ in (16) and using $K_j^T x(t) = \eta - u(t)$, with several manipulations yields

$$\begin{aligned}\frac{d}{dt}(x^T(t)P_j x(t)) &= -x^T(t)Qx(t) + 2\eta^T B^T P_j x(t)\\ &\quad + x^T(t)K_j^T R K_j x(t) - 2\big(\eta - u(t)\big)^T R\big(\eta - u(t)\big)\\ &\leq -\theta\mu\|x(t)\|^2 - (1-\theta)\mu\|x(t)\|^2 + 2\|\eta^T x(t)\|B^T P_j\\ &\quad + \lambda_{\max}(R)\|K_j x(t)\|^2 - 2\lambda_{\min}(R)\|\eta - u(t)\|^2.\end{aligned}$$

Based on (15), it is clear that

$$\dot{V}(x(t)) = \frac{d}{dt}(x^T(t)P_j x(t)) \preceq -(1-\theta)\mu\|x(t)\|^2.$$

Hence, asymptotic stability can be achieved. □

In order to validate the effectiveness of the control presented in equation (6), and to prevent the wasting of valuable online computing resources while avoiding the endless execution of Algorithm 1, the following convergence condition is provided to stop the algorithm.

Lemma 3. *Under the control law (6), the trajectory error is bounded by*

$$\mathtt{T}_{ee} = \frac{2\big(\|\eta^T\|\|RK_{j+1}\|\big)}{\lambda_{\min}(Q+K_j^T RK_j)}. \tag{17}$$

Proof. Choose a Lyapunov function candidate as $V(x(t)) = x^T(t)P_j x(t)$. By taking the derivative of $V(x(t))$ with respect to $x(t)$, we have

$$\begin{aligned}
\frac{d}{dt}(x^T(t)P_j x(t)) =& x^T(t)\big[A_j^T P_j + P_j A_j\big]x(t) + 2\eta^T B^T P_j x(t)\\
=& -x^T(t)\big(Q+K_j^T RK_j\big)x(t) + 2\eta^T RK_{j+1}x(t)\\
\leq& -\lambda_{\min}(Q+K_j^T RK_j)\|x(t)\|^2 + 2\big(\|\eta^T\|\|RK_{j+1}\|\|x(t)\|\big)\\
=& -\lambda_{\min}(Q+K_j^T RK_j)\|x(t)\|\Big(\|x(t)\| - \frac{2\big(\|\eta^T\|\|RK_{j+1}\|\big)}{\lambda_{\min}(Q+K_j^T RK_j)}\Big).
\end{aligned}$$

It is seen that $\frac{d}{dt}(V(x(t))) \preceq 0$ when $\|x(t)\| \geq \mathtt{T}_{ee}$. Hence, the trajectory error will converge to the ball $\|x(t)\| = \mathtt{T}_{ee}$. □

3.2 Value iteration

In this section, a VI algorithm is proposed. The PI Algorithm 1, starts with an initial stabilizer control gain matrix K_0. If the A and B matrices in system (1) are unknown, it can be difficult to find a gain matrix satisfying this requirement. To address this challenge, algorithm VI for system (1) is proposed in this section. The VI approach does not require a known stabilizing control law for initialization. First, we give several definitions. Let $\varepsilon > 0$ be a small threshold and $\{\mathcal{B}_i\}_{i=0}^{\infty}$ as a collection of bounded sets with nonempty interiors, and satisfying

$$\mathcal{B}_i \subseteq \mathcal{B}_{i+1},\ i \in \mathbb{Z}^+,\ \lim_{i\to\infty}\mathcal{B}_i = \mathscr{D}_+^n.$$

Define $\mathcal{L}_j = \frac{d}{dt}\Big(x^T(t)P_j x(t)\Big) + x^T(t)Qx(t)$ for each iteration of P_j. It follows from system (1) that

$$\begin{aligned}
\mathcal{L}_j &= \dot{x}^T(t)P_j x(t) + x^T(t)P_j\dot{x}(t) + x^T(t)Qx(t)\\
&= \big(Ax(t)+Bu(t)\big)^T P_j x(t) + x^T(t)P_j\big(Ax(t)+Bu(t)\big) + x^T(t)Qx(t)\\
&= x^T(t)\big(A^T P_j + P_j A\big)x(t) + 2u^T(t)B^T P_j x(t)\\
&= x^T(t)E_j x(t) + 2u^T(t)RK_j x(t),
\end{aligned} \tag{18}$$

where $E_j = A^T P_j + P_j A + Q$ and $K_j = R^{-1} B^T P_j$. By using the kronecker product representation and the predefined operators in notations, we have

$$\dot{x}(t) P_j x(t) = \big(x(t) \otimes \dot{x}(t)\big)^T \mathrm{vec}(P_j),$$
$$u^T(t) R K_j x(t) = \big[\big(x \otimes u\big)\big(I_n \otimes R\big)\big] \mathrm{vec}(K_j).$$

For convenience of description, we further define a set of functions $\mathcal{U}_{ab}$ as

$$\mathcal{U}_{ab} = [a(t_1) \otimes b(t_1), \ldots, a(t_r) \otimes b(t_r)]^T, \tag{20}$$

where a and b are column vectors.

Combining (18)-(20), the iterative learning equation (ILE), is given in the following linear parameters form:

$$\mathcal{W}^T \bar{\mathcal{Z}}_j = \bar{\mathcal{H}}_j, \tag{21}$$

where $\mathcal{W} = [\mathcal{U}_{xx}, 2\mathcal{U}_{xu}(I_n \otimes R)]$, $\bar{\mathcal{Z}}_j = [\mathrm{vech}(E_j), \mathrm{vec}(K_j)]$ and $\bar{\mathcal{H}}_j = 2\mathcal{U}_{x\dot{x}} \mathrm{vec}(P_j) + \mathcal{U}_{xx} \mathrm{vech}(Q)$.

Note that $\mathcal{W}$ is not square matrix in most cases, thus (21) is solved by using pseudo-inverse method as given in (22). Finally, the direct reinforcement learning (DRL)-based learning algorithm with VI scheme for system (1) is given in Algorithm 2. Its convergence is proved in Theorem 3.

Remark 2. *It is important to highlight that including exploration noise is crucial in the learning process as it helps to explore new policies that may be better than the current policy. However, there is a tradeoff between exploring new options and exploiting established policies, as new policies may not always be as effective as existing ones.*

Remark 3. *Algorithm 2 exhibits superiority over the other presented methods due to the following reasons:*

- ***Efficiency:*** *The proposed Algorithm 2 significantly reduces computational load by eliminating the need for repeated finite integrals, as opposed to the previous methods that heavily relied on such calculations ([1, 4, 15, 18]). This improvement in efficiency makes Algorithm 2 more suitable for practical implementation.*
- ***Online, Off-Policy Approach:*** *Algorithm 2 operates as an online off-policy learning algorithm. It does not depend on an initial stabilizing control policy or prior knowledge of system matrices, making it more flexible and adaptable to various control scenarios.*
- ***Reduced Complexity:*** *In contrast to [6], the design matrix $\mathcal{W}$ in (21) is independent of K_j. This eliminates the need for recalculating $\mathcal{W}$ in future learning iterations, leading to a reduced computational complexity per iteration compared to [6] for certain systems.*

Remark 4. *Algorithm 1 relies on a stabilizing control law being known, which could restrict its applications. However, it has a faster convergence rate compared to Algorithm 2, with quadratic convergence in the vicinity of the steady state. On the other hand, Algorithm 2 does not require a known stabilizing control law but may have a slower convergence rate as a trade-off.*

The differences between PI and VI are summarized in Table 1.

Similar to the analysis of Theorem 1, the convergence of Algorithm 2 is shown as follows.

Algorithm 2: VI for Optimal-Feedback Controller

Initialization: Choose $i, j \leftarrow 0$ and $P_0 = P_0^T \succ 0$.
Give an arbitrary initial control gain K_0 for the dynamic output feedback

$$\begin{cases} u(t) = -K_0 x(t) + \eta \\ \dot{x}(t) = Ax(t) + Bu(t) \end{cases}$$

Online data collection and control input:
Collect the online data of input $u_0(t_r)$ at the instant t_r, until the rank condition (14) is satisfied.
loop
  Solve (E_j, K_j) from the equation

$$\bar{\mathcal{Z}}_j = (\mathcal{W}^T \mathcal{W})^{-1} \mathcal{W}^T \bar{\mathcal{H}}_j. \tag{22}$$

  Update $\bar{\mathcal{P}}_{j+1}$ as,

$$\bar{\mathcal{P}}_{j+1} \leftarrow P_j + \varepsilon_j (E_j - K_j^T R K_j). \tag{23}$$

  if $\bar{\mathcal{P}}_{j+1} \notin \mathcal{B}_i$ **then**
    $P_{j+1} = P_0$ & $i \leftarrow i+1$
  else if $\|\bar{\mathcal{P}}_{j+1} - P_j\| < \varepsilon$ **then**
    return K_j as the estimate of K^*
    break
  else
    $P_{j+1} = \bar{\mathcal{P}}_{j+1}$
  end if
  $j \leftarrow j+1$
end loop
Update Controller: Update the control input to $u_j(t) = -K_j^* x(t)$.

Table 1: Algorithms comparison

Algorithm	PI	VI
Initial control policy	Required	Not required
Convergence rate	Quadratic	Sub-linear
Computational complexity	High	Low

Theorem 3. *If there exists a sampling instant t_i, such that the rank condition of (14) holds, then the sequences $\{P_j\}_{j=0}^{\infty}$ and $\{K_j\}_{j=1}^{\infty}$ obtained from Algorithm 2 converge to P^* and K^*.*

Proof. Note that $\mathcal{W}$ is independent of P_j and only relies on a series of past data. If the rank condition (14) is satisfied, it implies that $\mathcal{W}$ has full row rank. This indicates that equation (22) has a unique solution at each iteration. As shown in [1, 23], for an arbitrary P_0, the calculated $\bar{\mathcal{P}}_{j+1}$ will converge to P^* as

$j \to +\infty$ from the following equation

$$\bar{\mathcal{P}}_{j+1} \leftarrow P_j + \varepsilon_j \left(A^T P_j + P_j A + Q - P_j B R^{-1} B^T P_j\right). \tag{24}$$

In Algorithm 2, by substituting P_j into equation (21) and considering the full rankness of $\mathcal{W}$, equation (22) has a unique solution E_j and K_j at each iteration j. Consequently, P_{j+1} can be uniquely recalculated from equation (23). It should be pointed out that equation (23) is transformed from equation (24). Therefore, P_{j+1} computed from equations (22) and (23) converges to P^* as $j \to +\infty$. This implies that K_j converges to K^* as $j \to +\infty$. □

Now we show that the proposed method in Algorithm 2 is not affected by the exploration noise.

Theorem 4. *The proposed VI scheme is immune to exploration noise in parameter estimation.*

Proof. To demonstrate the immunity of Algorithm 2 to exploration noise, we define the control policy as $\hat{u}(t) = u(t) + \eta$, where η is the exploration noise. We aim to show that the VI-based learning equation (21) is not affected by exploration noise. Consider two pairs of estimates, $(\hat{E}_j, \hat{K}_j, \hat{P}_j)$ and (E_j, K_j, P_j), obtained using $\hat{u}(t)$ and $u(t)$, respectively. By combining (18) with the control input $\hat{u}(t)$, we obtain

$$2\dot{x}^T(t)\hat{P}_j x(t) + x^T(t)Qx(t) = x^T(t)\hat{E}_j x(t) + 2\hat{u}(t)^T R\hat{K}_j x(t). \tag{25}$$

Substituting the description of the closed-loop system (1) with $\hat{u}(t) = u(t) + \eta$ into (25), we can derive

$$\begin{aligned} &2\left(Ax(t) + Bu(t) + B\eta\right)^T \hat{P}_j x(t) + x^T(t)Qx(t) \\ =&x^T(t)\hat{E}_j x(t) + 2\left(u(t) + \eta\right)^T R\hat{K}_j x(t). \end{aligned} \tag{26}$$

By $\hat{K}_j = R^{-1}B^T\hat{P}_j$, we have $\eta^T B^T \hat{P}_j x(t) = \eta^T R\hat{K}_j x(t)$. Therefore, equation (25) can be rewritten as

$$2\left(Ax(t) + Bu(t)\right)^T \hat{P}_j x(t) + x^T(t)Qx(t) = x^T(t)\hat{E}_j x(t) + 2u^T(t)R\hat{K}_j x(t).$$

This form is equivalent to the learning equation derived using the control policy $u(t)$, which can be expressed as:

$$2\dot{x}^T(t)\hat{P}_j x(t) + x^T(t)Qx(t) = x^T(t)\hat{E}_j x(t) + 2u^T(t)R\hat{K}_j x(t). \tag{27}$$

From equations (18) and (27), it follows that $\hat{E}_j = E_j$ and $\hat{K}_j = K_j$ for any given $\hat{P}_0 = P_0$. This shows that the iterative learning method with the dynamic state feedback controller is immune to exploration noise. □

4 Numerical results

In this section, two simulation examples are provided to demonstrate the effectiveness of the developed online ADP algorithm.

Example 1. We study the controller design for a turbocharged diesel engine with exhaust gas recirculation, which is modeled by a sixth-order linear system. The system matrices A and B are directly extracted from [7] and are provided below:

$$A = \begin{bmatrix} -0.4125 & -0.0248 & 0.0741 & 0.0089 & 0 & 0 \\ 101.5873 & -7.2651 & 2.7608 & 2.8068 & 0 & 0 \\ 0.0704 & 0.0085 & -0.0741 & -0.0089 & 0 & 0.0200 \\ 0.0878 & 0.2672 & 0 & -0.3674 & 0.0044 & 0.3962 \\ -1.8414 & 0.0990 & 0 & 0 & -0.0343 & -0.0330 \\ 0 & 0 & 0 & -359.00 & 187.5364 & -87.0316 \end{bmatrix},$$

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

To demonstrate the effectiveness of the proposed computational adaptive optimal control approach, exact knowledge of A and B is not utilized in designing optimal controllers. Given the stability of the physical system, the initial stabilizing feedback gain can be set to $K_0 = 0$. The weighting matrices are chosen as

$$Q = \text{diag}\begin{bmatrix} 1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

During the simulation, initial values for the state variables are randomly chosen around the origin. From $t = 0$ s to $t = 2$ s, the exploration noise $e = \sum_{j=1}^{100} \sin(\Omega_j t)$ is used as the system input, where Ω_j, with $j = 1, \ldots, 100$ takes random numbers on $[-50, 50]$. State and input data are recorded over each 0.01-second interval. The PI started at $t = 2$ s, and convergence is attained after 16 iterations, when the stopping criterion $\|P_j - P_{j-1}\| < 0.003$ is satisfied. The convergence of P_j, K_j and u_j to their optimal values is depicted in Figure 1. Notice that if B is accurately known, the problem can also be solved using the method in [11]. Due to the PI algorithm's dependence on a difficult-to-satisfy initial stabilizing control gain, we have also implemented this example using Algorithm 2 and presented the convergence results in Figure 2. The average CPU time and the number of iterations of each algorithm for convergence are illustrated in Table 2. Now, we will provide a detailed explanation of how the proposed algorithm

Table 2: Performance comparison of Algorithms 1-2 in Example 1

Algorithm	PI	VI
No. of Iterations	16	100
CPU Time (sec)	0.35527	0.56301

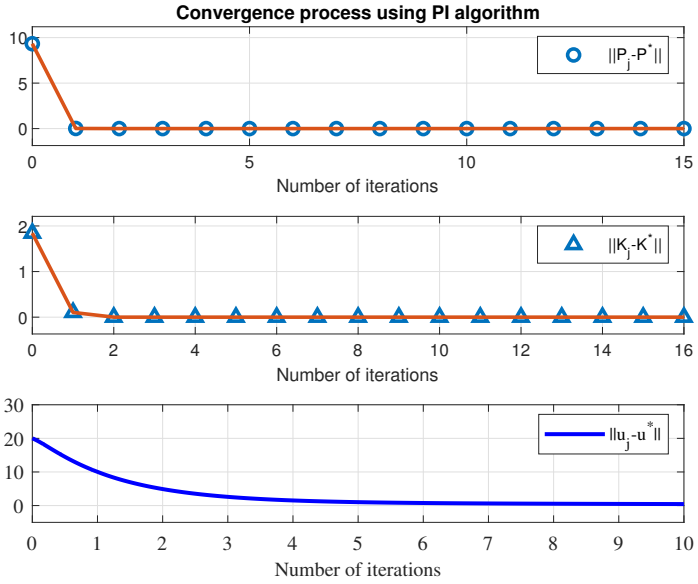

Figure 1: Convergence of P_j, K_j and u_j to their optimal values during the learning process using Algorithm 1 in Example 1

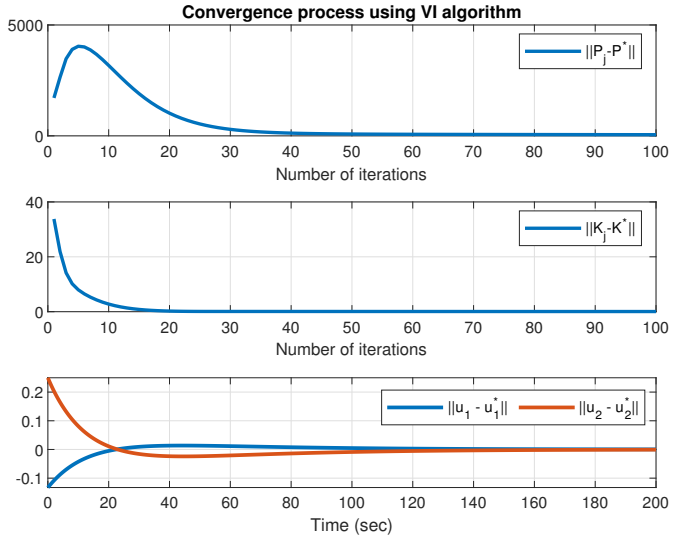

Figure 2: Convergence of P_j, K_j and u_j to their optimal values during the learning process using Algorithm 2 in Example 1

computes the cost and feedback gain matrices using the Algorithm 1 outlined below:

$$P_{\text{approx}} = \begin{bmatrix} 127.5320 & 0.5416 & 16.8310 & 1.8310 & 1.4033 & 0.0117 \\ 0.5416 & 0.0675 & 0.0375 & 0.0292 & 0.0433 & 0.0001 \\ 16.8310 & 0.0375 & 18.8042 & -0.3328 & 4.1500 & 0.0012 \\ 1.8310 & 0.0292 & -0.3328 & 0.5038 & -0.1223 & -0.0001 \\ 1.4033 & 0.0433 & 4.1500 & -0.1223 & 3.3589 & 0.0004 \\ 0.0117 & 0.0001 & 0.0012 & -0.0001 & 0.0004 & 0.0006 \end{bmatrix},$$

$$K_{\text{approx}} = \begin{bmatrix} -0.7952 & -0.0684 & -0.0725 & 0.0242 & -0.0487 & -0.0002 \\ 1.6511 & 0.1098 & 0.0974 & 0.0601 & 0.0211 & 0.0002 \end{bmatrix}.$$

For comparison purposes, we will also present the optimal solutions derived from directly solving the ARE, as indicated in equation (4), by utilizing the following MATLAB command: $[K_{\text{opt}}, P_{\text{opt}}] = \text{lqr}(A, B, Q, R)$:

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

$$K_{\text{opt}} = \begin{bmatrix} -0.7952 & -0.0684 & -0.0726 & 0.0242 & -0.0488 & -0.0002 \\ 1.6511 & 0.1098 & 0.0975 & 0.0601 & 0.0213 & 0.0002 \end{bmatrix}.$$

Additionally, we will introduce the matrices P_{approx} K_{approx}, P_{opt} and K_{opt}. These matrices are obtained by implementing Algorithm 2 with a specified interval length of $T = 0.1\,\text{s}$. This will provide a comprehensive comparison between the optimal and approximate solutions, highlighting the effectiveness of our proposed method:

$$P_{\text{approx}} = (1.0e+03) \begin{bmatrix} 1.2766 & 0.0045 & 0.1692 & 0.0174 & 0.0049 & 0.0001 \\ 0.0045 & 0.0013 & 0.0003 & 0.0004 & 0.0013 & 0.000 \\ 0.1692 & 0.0003 & 0.1913 & -0.0038 & 0.0382 & 0.000 \\ 0.0174 & 0.0004 & -0.0038 & 0.0050 & 0.0039 & 0.000 \\ 0.0049 & 0.0013 & 0.0382 & 0.0039 & 0.0831 & -0.000 \\ 0.0001 & 0.000 & 0.000 & 0.000 & -0.000 & 0.0001 \end{bmatrix},$$

$$K_{\text{approx}} = \begin{bmatrix} -1.0807 & -0.0433 & -0.0828 & 0.0115 & -0.0752 & 0.0001 \\ 2.1038 & 0.0704 & 0.1140 & 0.0730 & 0.0723 & 0.0003 \end{bmatrix}.$$

$$P_{\text{opt}} = (1.0e+03) \begin{bmatrix} 1.2750 & 0.0046 & 0.1683 & 0.0183 & 0.0140 & 0.0001 \\ 0.0046 & 0.0013 & 0.0004 & 0.0003 & 0.0004 & 0.000 \\ 0.1683 & 0.0004 & 0.1881 & -0.0033 & 0.0416 & 0.000 \\ 0.0183 & 0.0003 & -0.0033 & 0.0045 & -0.0009 & 0.000 \\ 0.0140 & 0.0004 & 0.0416 & -0.0009 & 0.0337 & -0.000 \\ 0.0001 & 0.000 & 0.000 & 0.000 & -0.000 & 0.0001 \end{bmatrix},$$

$$K_{\text{opt}} = \begin{bmatrix} -1.0856 & -0.0428 & -0.0842 & 0.0141 & -0.0482 & 0.0001 \\ 2.1127 & 0.0695 & 0.1176 & 0.0683 & 0.0237 & 0.0003 \end{bmatrix}.$$

Example 2. The F-16 aircraft control system is used to demonstrate the effectiveness of the theoretical results. The dynamics of the aircraft are defined as follows [13, 19]:

$$\dot{x}(t) = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -20.2 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 20.2 \end{bmatrix} u(t).$$

In this context, $x_1(t)$, $x_2(t)$ and $x_3(t)$ denote the angle of attack, pitch rate and elevator deflection angle, respectively. The variable $u(t)$ represents the elevator actuator angle.

In the simulation, the initial system state is chosen randomly and the sampling period is set to 0.01. Several parameters employed in the analysis include $Q = \text{diag}([0.1, 0, 0.1])$ and $R = \text{diag}([0.01])$. The learning error trajectories, specifically $\|P_j - P^*\|$, $\|K_j - K^*\|$ and $\|u - u^*\|$, are depicted in Figures 3 and 4. These figures demonstrate that P_j, K_j and u, obtained through Algorithms 1 and 2, converge towards the optimal values P^*, K^* and u^*. The comparison results presented in Table 3 indicate that the convergence speed of the PI algorithm is faster than that of the VI algorithm. It is worth pointing out that the PI algorithm requires an initial stabilizing matrix K_0, while the VI algorithm does not have this requirement. Consequently, the VI algorithm achieves rapid convergence without the necessity for an initial stabilizing K_0.

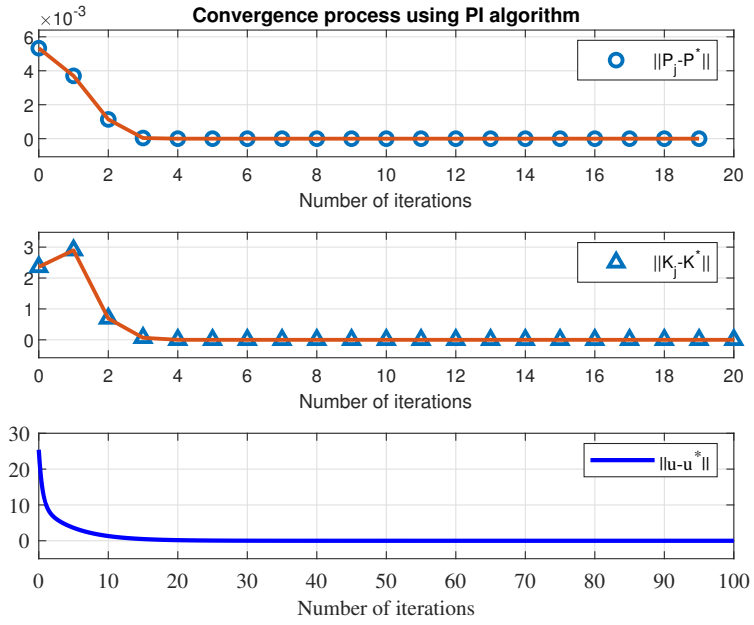

Figure 3: Convergence of P_j, K_j and u to their optimal values during the learning process using Algorithm 1 in Example 2

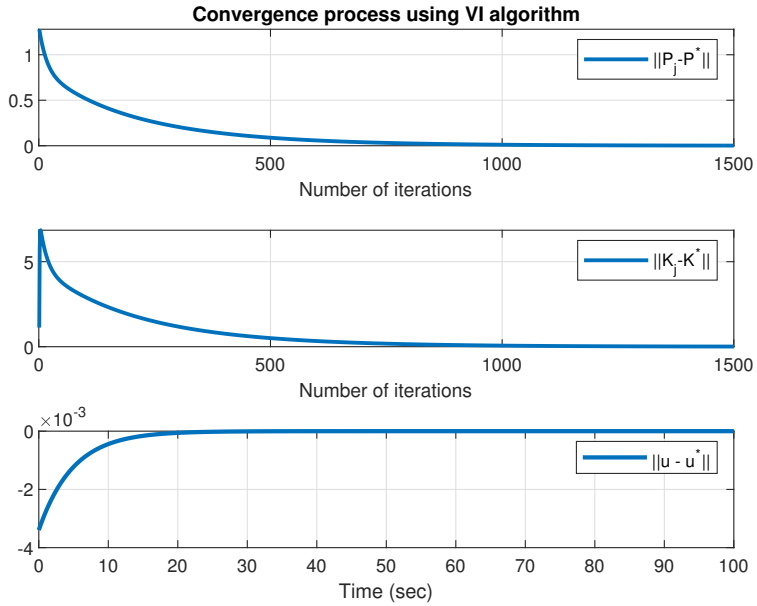

Figure 4: Convergence of P_j, K_j and u to their optimal values during the learning process using Algorithm 2 in Example 2

Table 3: Performance comparison of Algorithms 1-2 in Example 2

Algorithm	PI	VI
No. of Iterations	20	1000
CPU Time (sec)	0.40074	1.05073

5 Conclusion

This paper presents two computational approaches that utilize PI and VI algorithms to create online adaptive optimal controllers for CT linear systems with completely unknown dynamics. The PI algorithm iteratively solves the ARE by utilizing real-time state and input information, thereby eliminating the need for prior knowledge of the system matrices. Notably, the VI algorithm is flexible because it does not require K_0-stability, making it suitable for systems with uncertain dynamics. We demonstrated the effectiveness of methods by applying both algorithms to two practical examples: the control design of a turbocharged diesel engine and an F-16 aircraft. Each example illustrated the capabilities and performance of both algorithms in tackling the challenges posed by unknown dynamics.

Looking forward, our future research will focus on exploring time-varying delayed dynamical systems characterized by unknown dynamics.

Acknowledgements

We would like to express our sincere gratitude to Ferdowsi University of Mashhad, as well as to the anonymous reviewers for their invaluable comments that have led to the improvement of the paper.

Conflicts of interest

The authors declare that there are no conflicts of interest.

References

[1] T. Bian, Z.P. Jiang, *Value iteration and adaptive dynamic programming for data-driven adaptive optimal control design*, Automatica **71** (2016) 348–360.

[2] S. Effati, M. Pakdaman, *Optimal control problem via neural networks*, Neural Comput. Appl. **23** (2013) 2093–2100.

[3] S. Effati, A. Mansoori, M. Eshaghnezhad, *Linear quadratic optimal control problem with fuzzy variables via neural network*, J. Exp. Theor. Artif. Intell. **33** (2021) 283–296.

[4] W. Gao, M. Mynuddin, D. C. Wunsch, Z. P. Jiang, *Reinforcement learning-based cooperative optimal output regulation via distributed adaptive internal model*, IEEE Trans. Neural Netw. Learn. Syst. **33** (2021) 5229-–5240.

[5] P.A. Ioannou, J. Sun, *Robust Adaptive Control*, PTR Prentice-Hall, Upper Saddle River, NJ, 1996.

[6] Y. Jiang, Z.P. Jiang, *Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics*, Automatica. **48** (2012) 2699—2704.

[7] M. Jung, K. Glover, U. Christen, *Comparison of uncertainty parameterisations for H_∞ robust control of turbocharged diesel engines*, Control Eng. Pract. **13** (2005) 15–25.

[8] G. L. Kharatishvili, *The maximum principle in the theory of optimal processes involving delay*, Dokl. Akad. Nauk. **136** (1961) 39–42.

[9] D. Kleinman, *On an iterative technique for Riccati equation computations*, IEEE Trans. Automat. Control. **13** (1968) 114–115.

[10] A.J. Laub, *Matrix Analysis for Scientists and Engineers*, Society for Industrial and Applied Mathematics, Philadelphia, 2004.

[11] F. Lewis, D. Vrabie, *Reinforcement learning and adaptive dynamic programming for feedback control*, IEEE Circuits Syst. Mag. **9** (2009) 32–50.

[12] F. Lewis, D. Vrabie, V.L. Syrmos, *Optimal Control*, John Wiley & Sons, 2012.

[13] Y.S. Ma, J. Sun, Y. Xu, S.S. Cui, Z.G. Wu, *Adaptive dynamic programming for optimal control of unknown LTI system via interval excitation*, IEEE Trans. Automat. Control. **70(7)** (2025) 4896–4903. Online published, https://doi.org/10.1109/TAC.2025.3542328.

[14] A. Mansoori, S. Effati, M. Eshaghnezhad, *An iterative method to solve the time delay optimal control problem with bounded control variable*, Int. J. Comput. Math. **101** (2024) 653–667.

[15] H. Modares, F. Lewis, Z. P. Jiang, *Optimal output-feedback control of unknown continuous-time linear systems using off-policy reinforcement learning*, IEEE Trans. Cybern. **46** (2016) 2401–2410.

[16] D.S. Naidu, *Optimal Control Systems*, CRC press, 2018.

[17] L.S. Pontryagin, *Mathematical Theory of Optimal Processes*, Routledge, 2018.

[18] S.A. Rizvi, Z. Lin, *Reinforcement learning-based linear quadratic regulation of continuous-time systems using dynamic output feedback*, IEEE Trans. Cybern. **50** (2019) 4670–4679.

[19] B.L. Stevens, F. Lewis, E.N. Johnson, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*, John Wiley & Sons, 2015.

[20] R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction*, Bradford Book, Cambridge, MA, 2018.

[21] G. Tao, *Adaptive Control Design and Analysis*, John Wiley & Sons, 2003.

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

[23] K. Xie, Y. Zheng, W. Lan, X. Yu, *Adaptive optimal output regulation of unknown linear continuous-time systems by dynamic output feedback and value iteration*, Control Eng. Pract. **141** (2023) 105675.

[24] M. Zarefard, S. Effati, *Adaptive synchronization between two non-identical BAM neural networks with unknown parameters and time-varying delay*, Int. J. Control Autom. Syst. **15** (2017) 1877–1887.