Adaptive Optimal Control for A Class of Nonlinear Systems: The Online Policy Iteration Approach

DOI:
10.1109/TNNLS.2019.2905715

Citation for published version (APA):
He, S., Fang, H., Zhang, M., Liu, F., & Ding, Z. (2019). Adaptive Optimal Control for A Class of Nonlinear Systems: The Online Policy Iteration Approach. *IEEE Transactions on NEural Networks and Learning Systems*, *31*(2), 549-558. https://doi.org/10.1109/TNNLS.2019.2905715

Published in:
IEEE Transactions on NEural Networks and Learning Systems

To appear in IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS

Adaptive Optimal Control for A Class of Nonlinear Systems: The Online Policy Iteration Approach

Shuping He, *Member, IEEE*, Haiyang Fang, Maoguang Zhang, Fei Liu, *Member, IEEE*, Zhengtao Ding, *Senior Member, IEEE*

***Abstract*—This paper studies the online adaptive optimal controller design for a class of nonlinear systems through a novel policy iteration (PI) algorithm. By using the technique of neural network linear differential inclusion (LDI) to linearize the nonlinear terms in each iteration, the optimal law for controller design can be solved through the relevant algebraic Riccati equation (ARE) without using the system internal parameters. Based on PI approach, the adaptive optimal control algorithm is developed with the online linearization and the two-step iteration, i.e., policy evaluation and policy improvement. The convergence of the proposed PI algorithm is also proved. Finally, two numerical examples are given to illustrate the effectiveness and applicability of the proposed method.**

***Index Terms*—Nonlinear systems, policy iteration (PI), algebraic Riccati equation (ARE), adaptive optimal control, linear differential inclusion (LDI).**

I. Introduction

NONLINEAR control design has been an active research area for a long time. In the development of nonlinear control theory, optimality and veracity have become the fundamental principles of controller design. It is well known that designing an optimal controller for nonlinear systems depends on the unique positive-definite solution of the associated Hamilton-Jacobi-Bellman (HJB) equation [1], which reduces to the algebraic Riccati equation (ARE) when involved in linear systems. In general, the HJB equation is a nonlinear partial differential equation which is difficult even impossible to solve by analytical methods. As a favorable tool, dynamic programming (DP) [2] can be used to solve the HJB equation and a series of significant results centered on DP are published in [3]–[5]. Owing to the backward-in-time implementation of DP, these results are off-line and will encounter a dimensionality problem when they are applied in high-dimensional and more complex systems. In response to these problems, reinforcement learning (RL) or adaptive dynamic programming (ADP) have been introduced to solve the HJB equation [6]–[9]. One kind of RL algorithms, namely policy iteration (PI), can be used to deal with the adaptive optimal control problem of both linear and nonlinear systems, this algorithm contains two-step iteration, i.e., policy evaluation and policy improvement. The former means that the cost performance index relevant to the current control policy is evaluated, and the latter updates the policy to seek for a lower associated cost. In [10], Vrabie et al. proposed a novel online PI algorithm of continuous-time linear systems that can solve the ARE without knowing the internal dynamics; then the relevant method was introduced to similar linear systems with completely unknown dynamics [11]; He et al. [12] studied the adaptive optimal controller design for continuous-time Markov jump linear systems via an online PI algorithm; Liu and Wei [13] extended the PI algorithm to learn the infinite horizon optimal control solution for discrete-time nonlinear systems; and the authors in [14] discussed an online PI algorithm to solve the continuous-time optimal control problem of a class of unknown constrained-input systems. For the application of PI schemes in other respects, the readers can refer to [15]–[19].

On the other hand, neural network plays a very important role in solving nonlinear control problems because of its arbitrary approximation ability. The advantages of neural network over other linearization methods have been explained in [20]–[22]. In [23]–[27], neural network and PI algorithm are used to approximately solve the HJB equation. In these literatures, neural network was employed as critic network to represent the optimal cost function or actor network which represents the control policy. It is obvious that the studied results generated by these methods are effective. However, some computational complexity still exists and it might lead to some inevitable errors, because the approximation only stays at the level of computation and it is still difficult to solve the relevant HJB equation. Viewed from another angle, if we directly linearize the original system model by using neural network, it can fundamentally simplify the computation from the start, thereby avoiding solving the intractable nonlinear HJB equation. We always use this method to linearize the nonlinear model. It is known as neural network linear differential inclusion (LDI), proposed by Tanaka in [28]. As a specific neural network structure, the neural network LDI technique can convert the nonlinear terms into linear terms. Having been explicitly illuminated its principle in [28], the LDI technique was used in control design for nonlinear systems in discrete-time [29]

This work was supported in part by the National Natural Science Foundation of China under Grant 61673001, 61722306, the Foundation for Distinguished Young Scholars of Anhui Province under Grant 1608085J05, the Key Support Program of University Outstanding Youth Talent of Anhui Province under Grant gxydZD2017001, the State Key Program of National Natural Science Foundation of China under Grant 61833007 and the 111 Project under Grant B12018.

S. He, H. Fang and M. Zhang are with School of Electrical Engineering and Automation, Anhui University, Hefei 230601, China, S. He is also with Institute of Physical Science and Information Technology, Anhui University, Hefei, 230601, China (e-mail: shuping.he@ahu.edu.cn, fangocean1996@gmail.com).

F. Liu is with the Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), Institute of Automation, Jiangnan University, Wuxi 214122, China (email: fliu@jiangnan.edu.cn).

Z. Ding is with School of Electrical and Electronic Engineering, The University of Manchester, Manchester, M13 9PL, UK(email:zhengtao.ding@manchester.ac.uk).

and in continuous-time [30]. For other results related to the LDI method, the readers can refer to [31]–[33]. Notice that the previous approximation and the relevant design methods utilizing LDI are all off-line. There are few research results about the online LDI technique, and nothing of its application is reported in adaptive optimal control for nonlinear systems.

To realize the online adaptive algorithm, however, we need a synchronous linearization technique to accompany with the PI solution process. As a result, we put forward a new online LDI policy iteration (OLDIPI) algorithm, which can address the adaptive optimal control problem for a class of nonlinear systems. First, the original nonlinear system is approximated to a linear plant model based on the neural network LDI. Then inspired by the linear PI algorithm, the proposed OLDIPI algorithm can converge to the optimal solution; furthermore, the algorithm can be implemented online in least-square sense under a persistent excitation condition. The convergence of the proposed algorithm is also proved, and the corresponding simulation results are given to illustrate the feasibility and applicability.

The main contributions of this paper are as follows:

1) The online LDI representation is first realized and then used to solve the adaptive optimal control problems for nonlinear systems;
2) Compared with conventional off-line linearization methods (i.e., simply linearize the system around the origin once), our algorithm retains the nonlinear peculiarity of the system and guarantees the computational accuracy to the greatest extent. In addition, the combination of the online linearization and the online PI police provides a new perspective to address such problems;
3) By the designed OLDIPI algorithm, we do not have to solve the HJB equation. It simplifies the computation and improves the reality of the intractability of the optimal control problems for nonlinear systems. Based on this, the method has potential to solve some more complicated nonlinear control problems.

In our work, the designed algorithm is performed in an RL framework related to the LDI-represented neural network. Therefore, our designed nonlinear dynamical system can be treated as a type of learning systems [34]–[36]. This learning system design consists of two parts, i.e., the LDI-represented neural network approximation and the PI-based adaptive optimal control realization.

The remainder of this paper is organized as follows. In Section II, we give the LDI representation of a class of continuous-time nonlinear systems and the online PI algorithm of continuous-time linear systems. In Section III, we propose the OLDIPI algorithm and show its convergence, as well as the online implementation. In Section IV, two simulation examples are provided to demonstrate the effectiveness of the proposed algorithm. Finally, a brief conclusion and extension are contained in Section V.

II. Backgrounds

A. Neural-Network-Based System Description

The general nonlinear dynamical systems can be described by:

$$\dot{x}(t) = f(x(t), u(t)). \tag{1}$$

To some extent, many nonlinear systems can be regarded as the coupling of linear and nonlinear parts. To study the adaptive optimal control algorithm, in this paper, we consider a class of nonlinear system in the form of:

$$\dot{x}(t) = Ax(t) + Bu(t) + f(x(t)) + g(u(t)) \tag{2}$$

where $x(t) \in \Re^n$ is the state vector, $u(t) \in \Re^m$ is the control input; $f(\cdot)$ and $g(\cdot)$ are continuous and bounded nonlinear mapping of the state and control input; $A \in \Re^{n\times n}$ is the unknown constant matrix, $B \in \Re^{n\times m}$ is the known constant matrix, A and B are with the appropriate dimensions; moreover, (A, B) is assumed to be stabilizable.

The optimal control problem for nonlinear system (2) in the infinite horizon is equivalent to finding an optimal control policy, which can be given as follows:

$$u(t) = u^*(x(t)) = -K^* x(t) \tag{3}$$

which minimizes the following infinite horizon performance index:

$$V(x(t), u(t)) = \int_0^\infty [x^{\mathrm{T}}(\tau)Qx(\tau) + u^{\mathrm{T}}(\tau)Ru(\tau)]\mathrm{d}\tau \tag{4}$$

where $Q = Q^T \geq 0, R = R^T > 0$ and we assume that $(A, B, Q^{1/2})$ is stabilizable and detectable.

Remark 1: The nonlinear representation in system (2) can be considered as a special class of nonlinear systems (1). For system (2), although the nonlinear term $f(x(t))$ and $g(u(t))$ can be treated by some other methods, the expected accuracy is still a challenge. In our design, we will develop an LDI representation for a more effective adaptive optimal control.

For the purpose of designing the above control policy of nonlinear system (2), we need to deal with the nonlinear terms $f(x(t))$ and $g(u(t))$ firstly. By means of the arbitrary approximation ability of the neural network, we give a class of multi-layered perceptions (MLP's) that admit an LDI state-space representation.

Lemma 1 [30]: For the nonlinear terms $f(x(t))$ and $g(u(t))$, we give the following approximation representations by two L-layered MLP's:

$$\begin{cases} NN_x(x(t), W_1, W_2, ..., W_L) \\ \quad = \mathbf{\Psi}_L[W_L ... \mathbf{\Psi}_2[W_2 \mathbf{\Psi}_1[W_1 x(t)]]...], \\ NN_u(u(t), V_1, V_2, ..., V_L) \\ \quad = \mathbf{\Psi}_L[V_L ... \mathbf{\Psi}_2[V_2 \mathbf{\Psi}_1[V_1 u(t)]]...]. \end{cases} \tag{5}$$

where $W_i (i = 1, 2, ...L) \in \Re^{{n_i}^x \times n_{i-1}^x}$ and $V_i (i = 1, 2, ...L) \in \Re^{{n_i}^u \times n_{i-1}^u}$ respectively denote the relevant weight matrices from the $(i-1)$th layer to the ith layer; $n_i^s (i = 1, 2, ...L; s = x$ or $u)$ denotes the number of neurons in the ith layer, the superscript s denotes that the MLP's are associated with x or u; the activation function vector is defined as $\mathbf{\Psi}(\xi) \triangleq [\psi_1(\xi_1), \psi_2(\xi_2), ..., \psi_n(\xi_n)]^{\mathrm{T}}$.

Assume that the activation function is a class of sigmoid functions:

$$\psi_i(\xi_i) = \lambda_i(\frac{2}{1+e^{-\xi_i/q_i}}); q_i, \lambda_i > 0. \tag{6}$$

Furthermore, we denote the minimum and maximum values of $'(\xi)$ as:

$$s(k,\psi) = \begin{cases} \text{min}_\xi \psi'(\xi), & k=0, \\ \text{max}_\xi \psi'(\xi), & k=1. \end{cases} \tag{7}$$

Recalling to [37],$\psi_i(\xi_i)$ can be represented as:

$$\psi_i(\xi_i) = h(0)s(0,\psi_i) + h(1)s(1,\psi_i) \tag{8}$$

where $h(i) \geq 0 (i = 0, 1)$, and $\sum_{i=0}^{1} h(i) = 1$. This indicates that $\psi_i(\xi_i)$ can be implemented by interpolating the two straight lines whose slopes are $s(0,\psi_i)$ and $s(1,\psi_i)$.

In order to adopt the compact representation in [29], we define a set of n_i^s-dimensional index vectors tied to the ith layer of the approximating MLP's as:

$$\mathbf{\Upsilon}_{n_i^s} = \mathbf{\Upsilon}_{n_i^s}(\delta) \triangleq \{\delta \in \Re^{n_i^s} | \delta_j \in \{0,1\}\} \tag{9}$$

where δ is used as a binary indicator, $j(j = 1, 2, ..., n_i^s)$ denotes the jth neuron in ith layer.

By adopting the compact representation in [29], the MLP's $NN_x(x(t), W_1, W_2, ..., W_L)$ can be given as:

$$\begin{aligned}
&NN_x(x(t), W_1, W_2, ..., W_L) \\
&\quad = \mathbf{\Psi}_L[W_L...\mathbf{\Psi}_2[W_2\mathbf{\Psi}_1[W_1x(t)]]...] \\
&\quad = \mathbf{\Psi}_L[W_L...\mathbf{\Psi}_2[W_2[\sum_{i=0}^{1} h_{11}(i)s(i,\psi_{11})(W_1x(t))_1, \\
&\qquad ..., \sum_{i=0}^{1} h_{1n_1^x}(i)s(i,\psi_{1n_1^x})(W_1x(t))_{n_1^x}]^{\mathrm{T}}]...] \\
&\quad = \mathbf{\Psi}_L[W_L...\mathbf{\Psi}_2[W_2 \sum_{i_{11}=0}^{1} ... \sum_{i_{1n_1^x}=0}^{1} h_{11}(i_{11})...h_{1n_1^x}(i_{1n_2^x}) \\
&\qquad \cdot\text{diag}[s_{1i}(i_{1i},\psi_{1i})]W_1x(t)]...] \\
&\quad = \mathbf{\Psi}_L[W_L... \sum_{i_{21}=0}^{1} ... \sum_{i_{2n_2^x}=0}^{1} h_{21}(i_{21})...h_{2n_2^x}(i_{2n_2^x}) \\
&\qquad \cdot\text{diag}[s_{2i}(i_{2i},\psi_{2i})][W_2 \textstyle\sum_{i_{11}=0}^{1} ... \sum_{i_{1n_1^x}=0}^{1} h_{11}(i_{11})... \\
&\qquad h_{1n_1^x}(i_{1n_1^x})\text{diag}[s_{1i}(i_{1i},\psi_{1i})]W_1x(t)]...] \\
&\quad = \mathbf{\Psi}_L[W_L... \sum_{i_{21}=0}^{1} ... \sum_{i_{2n_2^x}=0}^{1} \sum_{i_{11}=0}^{1} ... \sum_{i_{1n_1^x}=0}^{1} h_{11}(i_{11})... \\
&\qquad h_{1n_1^x}(i_{1n_1^x})h_{21}(i_{21})...h_{2n_2^x}(i_{2n_2^x})\text{diag}[s_{2i}(i_{2i},\psi_{2i})] \\
&\qquad \cdot[W_2\text{diag}[s_{1i}(i_{1i},\psi_{1i})]W_1x(t)]...].
\end{aligned} \tag{10}$$

Similarly, $NN_u(u(t), V_1, V_2, ..., V_L)$ can be expressed as:

$$\begin{aligned}
&NN_x(u(t), V_1, V_2, ..., V_L) \\
&\quad = \mathbf{\Psi}_L[V_L...\mathbf{\Psi}_2[V_2\mathbf{\Psi}_1[V_1u(t)]]...] \\
&\quad = \mathbf{\Psi}_L[V_L...\mathbf{\Psi}_2[V_2[\sum_{i=0}^{1} h_{11}(i)s(i,\psi_{11})(V_1u(t))_1, \\
&\qquad ..., \sum_{i=0}^{1} h_{1n_1^u}(i)s(i,\psi_{1n_1^u})(V_1u(t))_{n_1^u}]^{\mathrm{T}}]...] \\
&\quad = \mathbf{\Psi}_L[V_L...\mathbf{\Psi}_2[V_2 \sum_{i_{11}=0}^{1} ... \sum_{i_{1n_1^u}=0}^{1} h_{11}(i_{11})...h_{1n_1^u}(i_{1n_2^u}) \\
&\qquad \cdot\text{diag}[s_{1i}(i_{1i},\psi_{1i})]V_1u(t)]...] \\
&\quad = \mathbf{\Psi}_L[V_L... \sum_{i_{21}=0}^{1} ... \sum_{i_{2n_2^u}=0}^{1} h_{21}(i_{21})...h_{2n_2^u}(i_{2n_2^u}) \\
&\qquad \cdot\text{diag}[s_{2i}(i_{2i},\psi_{2i})][V_2 \textstyle\sum_{i_{11}=0}^{1} ... \sum_{i_{1n_1^u}=0}^{1} h_{11}(i_{11})... \\
&\qquad h_{1n_1^u}(i_{1n_1^u})\text{diag}[s_{1i}(i_{1i},\psi_{1i})]V_1u(t)]...] \\
&\quad = \mathbf{\Psi}_L[V_L... \sum_{i_{21}=0}^{1} ... \sum_{i_{2n_2^u}=0}^{1} \sum_{i_{11}=0}^{1} ... \sum_{i_{1n_1^u}=0}^{1} h_{11}(i_{11})... \\
&\qquad h_{1n_1^u}(i_{1n_1^u})h_{21}(i_{21})...h_{2n_2^u}(i_{2n_2^u})\text{diag}[s_{2i}(i_{2i},\psi_{2i})] \\
&\qquad \cdot[V_2\text{diag}[s_{1i}(i_{1i},\psi_{1i})]V_1u(t)]...].
\end{aligned} \tag{11}$$

Suppose that there exist optimal approximation weights $\{W_L^*, ..., W_1^*\}$ and $\{V_L^*, ..., V_1^*\}$, which satisfy

$$\begin{cases} \|f(x(t)) - NN_x(x(t), W^*)\| \leq \varepsilon_1 \|x(t)\|, \\ \|g(u(t)) - NN_u(u(t), V^*)\| \leq \varepsilon_2 \|u(t)\|. \end{cases} \tag{12}$$

where $\varepsilon_1 > 0, \varepsilon_2 > 0$.

According to [30], the nonlinear system (2) can be written as:

$$\begin{aligned}
\dot{x}(t) = [A + &\sum_{\sigma \in \Upsilon_{n_1^x} \oplus ... \Upsilon_{n_L^x}} \mu_\sigma A_\sigma(\sigma, \Psi, W^*)]x(t) \\
&+[B + \sum_{\eta \in \Upsilon_{n_1^u} \oplus ... \Upsilon_{n_L^u}} \mu_\eta B_\eta(\eta, \Psi, V^*)]u(t)
\end{aligned} \tag{13}$$

where
$\sum_{\sigma \in \Upsilon_{n_1^x} \oplus ... \Upsilon_{n_L^x}} \mu_\sigma = \sum_{i_{L1}=0}^{1} \cdots \sum_{i_{Ln_L^x}=0}^{1} \cdots \sum_{i_{21}=0}^{1} \cdots$
$\sum_{i_{2n_2^x}=0}^{1} \sum_{i_{11}=0}^{1} \cdots \sum_{i_{1n_1^x}=0}^{1} h_{11}(i_{11})...h_{1n_1^x}(i_{1n_1^x})h_{21}(i_{21})$
$...h_{2n_2^x}(i_{2n_2^x})...h_{L1}(i_{L1})...h_{Ln_L^x}(i_{Ln_L^x}) = 1,$
$\sum_{\eta \in \Upsilon_{n_1^u} \oplus ... \Upsilon_{n_L^u}} \mu_\eta = \sum_{i_{L1}=0}^{1} \cdots \sum_{i_{Ln_L^u}=0}^{1} \cdots \sum_{i_{21}=0}^{1} \cdots$
$\sum_{i_{2n_2^u}=0}^{1} \sum_{i_{11}=0}^{1} \cdots \sum_{i_{1n_1^u}=0}^{1} h_{11}(i_{11})...h_{1n_1^u}(i_{1n_1^u})h_{21}(i_{21})$
$...h_{2n_2^u}(i_{2n_2^u})...h_{L1}(i_{L1})...h_{Ln_L^u}(i_{Ln_L^u}) = 1,$
$A_\sigma(\sigma, \Psi, W^*) =$diag$[s_{Li}(\sigma_{Li}, \psi_{Li})]W_L^*...W_2^*$
$\cdot$diag$[s_{1i}(\sigma_{1i}, \psi_{1i})]W_1^*$,
$B_\eta(\eta, \Psi, V^*) =$diag$[s_{Li}(\eta_{Li}, \psi_{Li})]V_L^*...V_2^*$
$\cdot$diag$[s_{1i}(\eta_{1i}, \psi_{1i})V_1^*$.

This kind of representation can be considered as a kind of LDI [38]. After using the LDI approximation to model the nonlinear terms, the PI algorithm and the relevant adaptive optimal control scheme will be studied in the following section for nonlinear system (2).

Remark 2: Eqs. (10)-(13) have illustrated how to use the LDI method to model the $f(x(t))$ and $g(u(t))$ function. In each iteration step of the following Algorithm 2, it involves a modeling process on piecewise $f(x(t))$ and $g(u(t))$.

Remark 3: There are many different LDI methods to address different issues. In [33], the authors give a common LDI description, i.e., $\dot{x} = \Omega x$, where Ω is a subset of $\Re^{n\times n}$. It treats the LDI as a tool to study the uncertainty of a family of linear time-varying systems. In [26], the LDI method is used to linearize and discuss the stability of a class of discrete-time nonlinear systems. While in our design, we use the neural network LDI method to convert the nonlinear terms $f(x(t))$ and $g(u(t))$ to the form as presented in (13), and subsequently in Algorithm 2, we will develop it to an online fashion.

B. PI Algorithm

Before designing the controll policy for nonlinear system (2), we give the following lemmas and algorithm for the linear case.

Lemma 2 [1]: For a class of linear systems $\dot{x}(t) = Ax(t) + Bu(t)$, if the controller is chosen as $u(t) = -Kx(t)$, the solution of the quadratic optimal control in the infinite horizon is:

$$K^* = R^{-1}B^{\mathrm{T}}P^* \tag{14}$$

where P^* is the positive definite solution of the following algebraic Riccati equation (ARE):

$$A^{\mathrm{T}}P + PA - PBR^{-1}B^{\mathrm{T}}P + Q = 0. \tag{15}$$

Lemma 3 [39]: Let $K(1) = 0$ be an initial stabilizing feedback gain matrix and $P(k)$ be the unique positive definite solutions of the Lyapunov equation:

$$\begin{aligned}[A - BK^{\mathrm{T}}(k)]P(k) + &P(k)[A - BK(k)] \\ &+ Q + K^{\mathrm{T}}(k)RK(k) = 0\end{aligned} \tag{16}$$

where, recursively,

$$K(k+1) = R^{-1}B^{\mathrm{T}}P(k). \tag{17}$$

Then, the following properties hold:

1) $A - BK(k)$ is Hurwitz;
2) $\lim_{k\to\infty} P(k) = P^*, \lim_{k\to\infty} K(k) = K^*$.

In Lemma 3, the numerical approximate solution of the ARE (15) can be attained by iteratively solving Lyapunov Eq. (16) and updating $K(k)$ by (17). Based on the work in Lemma 3, Vrabie et al. [10] proposed an online PI algorithm that can solve (16) without knowing the internal plant matrix A. The algorithm is shown as Algorithm 1, which can be effectively converged to the ARE (15) to solve matrix P.

III. Main Results

(17) OLDIPI Algorithm

In Section II, we have given the linear representation of nonlinear system (2) based on LDI approximation, then we apply it to the following OLDIPI algorithm shown as Algorithm 2.

Algorithm 1 PI Algorithm

• *Step 1:* Let $k = 1$. Give an initial stabilizing control policy $K(1) = 0$ and a cost matrix $P(0) = 0$, plus a small constant $\epsilon(\epsilon > 0)$ as the accuracy. Start the iteration.

• *Step 2:* Solve the following PI equation.

1) **Policy Evaluation**:

$$\begin{aligned}x_t^{\mathrm{T}}P(k)x_t = &\int_t^{t+T} x_\tau[Q + K^{\mathrm{T}}(k)RK(k)]x_\tau^{\mathrm{T}}\mathrm{d}\tau \\ &+ x_{t+T}^{\mathrm{T}}P(k)x_{t+T}.\end{aligned} \tag{18}$$

2) **Policy Improvement**:

$$K(k+1) = R^{-1}B^{\mathrm{T}}P(k). \tag{19}$$

• *Step 3:* If $\|P(k) - P(k-1)\| \leq \epsilon$, we stop and output $P(k)$ as the solution matrix P. Otherwise, set $k = k+1$, go to Step 2 and continue the iteration process.

Considering that $\Delta A(k)$ and $\Delta B(k)$ are derived by the continuous and bounded nonlinear function $f(x(t))$ and $g(u(t))$ in (2), the values of them are thereby restricted in a very small scale in the single iteration. By Theorem 3.1 in [28], the neural network LDI-described system is asymptotically stable as long as the corresponding conditions are satisfied. Thus, we can assume that $(\overline{A}(k), \overline{B}(k), Q^{1/2})$ is stabilizable and detectable in each iteration.

Based on the above assumption, the convergence of Algorithm 2 can be guaranteed by the following theorems.

Theorem 1: Assuming that $\overline{A}(k) - \overline{B}(k)\overline{K}(k)$ is always stable, the policy evaluation (23) in Algorithm 2 is equivalent to finding the solution of Lyapunov Eq. (22).

Proof: Considering the control gain as $\overline{K}(k)$, the kth iteration of the closed-loop form of nonlinear system (2) can be depicted:

$$\dot{x}_t = [\overline{A}(k) - \overline{B}(k)\overline{K}(k)]x_t. \tag{25}$$

Choosing the Lyapunov candidate function as $V_k(x_t) = x_t^{\mathrm{T}}\overline{P}(k)x_t$, we have

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}x_t^{\mathrm{T}}\overline{P}(k)x_t = &x_t^{\mathrm{T}}[(\overline{A}(k) - \overline{B}(k)\overline{K}(k))^{\mathrm{T}}\overline{P}(k) \\ &+ \overline{P}(k)(\overline{A}(k) - \overline{B}(k)\overline{K}(k))]x_t.\end{aligned} \tag{26}$$

Integrating (26) from t to $t+T$, it has

$$\begin{aligned}x_{t+T}^{\mathrm{T}}&\overline{P}(k)x_{t+T} - x_t^{\mathrm{T}}\overline{P}(k)x_t \\ &= \int_t^{t+T} x_\tau^{\mathrm{T}}[(\overline{A}(k) - \overline{B}(k)\overline{K}(k))^{\mathrm{T}}\overline{P}(k) \\ &\quad + \overline{P}(k)(\overline{A}(k) - \overline{B}(k)\overline{K}(k))]x_\tau\mathrm{d}\tau.\end{aligned} \tag{27}$$

Then it follows from (23) that

$$\begin{aligned}x_{t+T}^{\mathrm{T}}&\overline{P}(k)x_{t+T} - x_t^{\mathrm{T}}\overline{P}(k)x_t \\ &= -\int_t^{t+T} x_\tau^{\mathrm{T}}[Q + \overline{K}^{\mathrm{T}}(k)R\overline{K}(k)]x_\tau\mathrm{d}\tau.\end{aligned} \tag{28}$$

Comparing (27) and (28), we can get (22). Therefore, the asymptotic stability of $\overline{A}(k) - \overline{B}(k)\overline{K}(k)$ can guarantee that the solution of (23) equals to the unique solution of (22). ■

Algorithm 2 OLDIPI Algorithm

- *Step 1:* Let $k = 1$. Give an initial stabilizing control policy $\overline{K}(1) = 0$ and a cost matrix $\overline{P}(0) = 0$, plus a small constant $\epsilon(\epsilon > 0)$ as the accuracy. Start the iteration.
- *Step 2:* In each iteration, the nonlinear terms $f(x(t))$ and $g(u(t))$ are sampled at a shorter interval δ, for the aim of obtaining a series of sampling points:

$$\begin{cases} x_\delta = [x(t), x(t+\delta), x(t+2\delta), ..., x(t+T)], \\ u_\delta = [u(t), u(t+\delta), u(t+2\delta), ..., u(t+T)]. \end{cases} \tag{20}$$

Then according to Lemma 1, the nonlinear system states in kth iteration can be linerized by LDI:

$$\begin{cases} f(x(t)) = \sum\limits_{\sigma\in\Upsilon_{n_1^x}\oplus\ldots\Upsilon_{n_L^x}} \mu_\sigma A_\sigma(k)(\sigma,\Psi,W^*)x_\delta(t) \\ \qquad = \Delta A(k)x_\delta(t), \\ g(u(t)) = \sum\limits_{\eta\in\Upsilon_{n_1^u}\oplus\ldots\Upsilon_{n_L^u}} \mu_\eta B_\eta(k)(\eta,\Psi,V^*)u_\delta(t) \\ \qquad = \Delta B(k)u_\delta(t). \end{cases} \tag{21}$$

The new Lyapunov matrix equation in the kth step will be

$$[\overline{A}(k) - \overline{B}(k)\overline{K}(k)]^{\mathrm{T}}\overline{P}(k) + \overline{P}(k)[\overline{A}(k) - \overline{B}(k)\overline{K}(k)] = -[\overline{K}^{\mathrm{T}}(k)R\overline{K}(k) + Q]. \tag{22}$$

where $\overline{A}(k) = A + \Delta A(k)$, $\overline{B}(k) = B + \Delta B(k)$.

- *Step 3:*Substitute the above results into (18) and (19), the final problem becomes the solution of the following iterative equation:

1) **Policy Evaluation**:

$$x_t^{\mathrm{T}}\overline{P}(k)x_t = \int_t^{t+T} x_\tau[Q + \overline{K}^{\mathrm{T}}(k)R\overline{K}(k)]x_\tau^{\mathrm{T}}\mathrm{d}\tau + x_{t+T}^{\mathrm{T}}\overline{P}(k)x_{t+T}. \tag{23}$$

2) **Policy Improvement**:

$$\overline{K}(k+1) = R^{-1}\overline{B}^{\mathrm{T}}(k)\overline{P}(k). \tag{24}$$

- *Step 3:* If $\|\overline{P}(k) - \overline{P}(k-1)\| \le \epsilon$, we stop and output $\overline{P}(k)$ as the solution matrix $\overline{P}$. Otherwise, set $k = k+1$, go to Step 2 and continue the iteration process.

Theorem 2: Assuming that the control policy $\overline{K}(k)$ is stabilizing with the associated cost $V_k(x_t)$. Then the next step control policy $\overline{K}(k+1)$ attained by (24) will be stabilizing as well.

Proof: Considering $V_k(x_t)$ as the Lyapunov function associated with $\overline{K}(k+1)$ and taking the derivative of $V_k(x_t)$, we have:

$$\begin{aligned} \dot{V}_k(x_t) &= x_t^{\mathrm{T}}[\overline{P}(k)(\overline{A}(k) - \overline{B}(k)\overline{K}(k+1)) \\ &\quad + (\overline{A}(k) - \overline{B}(k)\overline{K}(k+1))^{\mathrm{T}}\overline{P}(k)]x_t \\ &= x_t^{\mathrm{T}}[\overline{P}(k)(\overline{A}(k) - \overline{B}(k)\overline{K}(k)) \\ &\quad + (\overline{A}(k) - \overline{B}(k)\overline{K}(k))^{\mathrm{T}}\overline{P}(k)]x_t \\ &\quad + x_t^{\mathrm{T}}[\overline{P}(k)\overline{B}(k)(\overline{K}(k) - \overline{K}(k+1)) \\ &\quad + (\overline{K}(k) - \overline{K}(k+1))\overline{B}^{\mathrm{T}}(k)\overline{P}(k)]x_t. \end{aligned} \tag{29}$$

By means of (22) and (24), the first term equals to $-[\overline{K}^{\mathrm{T}}(k)R\overline{K}(k) + Q]$, and the second can be rewritten as:

$$\begin{aligned} & x_t^{\mathrm{T}}[\overline{P}(k)\overline{B}(k)(\overline{K}(k) - \overline{K}(k+1)) \\ & +(\overline{K}(k) - \overline{K}(k+1))\overline{B}^{\mathrm{T}}(k)\overline{P}(k)]x_t \\ =\ & x_t^{\mathrm{T}}[\overline{K}^{\mathrm{T}}(k+1)R(\overline{K}(k) - \overline{K}(k+1)) \\ & +(\overline{K}(k) - \overline{K}(k+1))^{\mathrm{T}}R\overline{K}(k+1)]x_t \\ =\ & x_t^{\mathrm{T}}[-(\overline{K}(k) - \overline{K}(k+1))^{\mathrm{T}}R(\overline{K}(k) - \overline{K}(k+1)) \\ & -\overline{K}^{\mathrm{T}}(k+1)R\overline{K}(k+1) + \overline{K}^{\mathrm{T}}(k)R\overline{K}(k)]x_t. \end{aligned}$$

In the end, Eq. (29) is turned into:

$$\begin{aligned} \dot{V}_k(x_t) &= -x_t^{\mathrm{T}}[\overline{K}^{\mathrm{T}}(k+1)R\overline{K}(k+1) + Q]x_t \\ &\quad - x_t^{\mathrm{T}}[(\overline{K}(k) - \overline{K}(k+1))^{\mathrm{T}}R(\overline{K}(k) - \overline{K}(k+1))]x_t. \end{aligned} \tag{30}$$

Considering $Q \ge 0, R > 0$ in (4), it is clear that $\dot{V}_k(x_t) < 0$. Thus, the new control policy $\overline{K}(k+1)$ updated by (24) is stabilizing. It indicates that if the initial control policy is stabilizing, $\overline{A}(k) - \overline{B}(k)\overline{K}(k)$ is always stable in the whole iteration process. ■

Theorem 3 (Convergence): Under the assumptions of the stabilizability of $(\overline{A}(k), \overline{B}(k))$ and the detectability of $(Q^{1/2}, \overline{A}(k))$ at each iteration, with $Q \ge 0, R > 0$ in (4), the OLDIPI algorithm in Algorithm 2 converges to the optimal solution of the corresponding ARE:

$$\overline{A}^{\mathrm{T}}(k)\overline{P}(k) + \overline{P}(k)\overline{A}(k) - \overline{P}(k)\overline{B}(k)R^{-1}\overline{B}^{\mathrm{T}}(k)\overline{P}(k) + Q = 0. \tag{31}$$

Proof: According to Eq. (12), we can see that if $f(x(t))$, $g(u(t))$ are bounded, $\|\overline{A}(k)\|$, $\|\overline{B}(k)\|$ will be bounded as well. Let $\|\overline{A}_M\|$, $\|\overline{B}_M\|$ be the upper bounds of $\|\overline{A}(k)\|$ and $\|\overline{B}(k)\|$, respectively, i.e.,

$$\|\overline{A}(k)\| \le \|\overline{A}_M\|, \|\overline{B}(k)\| \le \|\overline{B}_M\| \tag{32}$$

where $\|\cdot\|$ denotes the induced matrix norm.

Assuming that $\overline{A}(k)$ and $\overline{B}(k)$ remain in their upper bounds at each iteration. By Theorem 1, we know that in this case the solution of the policy evaluation (23) is equivalent to solving the following Lyapunov equation:

$$[\overline{A}_M - \overline{B}_M\overline{K}(k)]^{\mathrm{T}}\overline{P}(k) + \overline{P}(k)[\overline{A}_M - \overline{B}_M\overline{K}(k)] = -[\overline{K}^{\mathrm{T}}(k)R\overline{K}(k) + Q]. \tag{33}$$

By Lemma 3, we know that by solving the Lyapunov equation iteratively in each iteration, the solution to the ARE is numerically approximated. The iteration equivalence between (22), (24) and (23), (24) implies that in this case the solutions of Algorithm 2 converge to the optimal solution of the following ARE:

$$\overline{A}_M^{\mathrm{T}}\overline{P}_M + \overline{P}_M\overline{A}_M - \overline{P}_M\overline{B}_M R^{-1}\overline{B}_M^{\mathrm{T}}\overline{P}_M + Q = 0. \tag{34}$$

Note that $\lim_{k\to\infty}\overline{P}(k) = \overline{P}_M^*$, where $\overline{P}_M^*$ denotes the unique positive definite solution of (34).

Although $\overline{A}(k)$ and $\overline{B}(k)$ are always changeable at each iteration, the optimal control conditions are still satisfied. Considering the upper bounds $\overline{A_M}$ and $\overline{B_M}$, it is concluded that the solutions of the OLDIPI algorithm in Algorithm

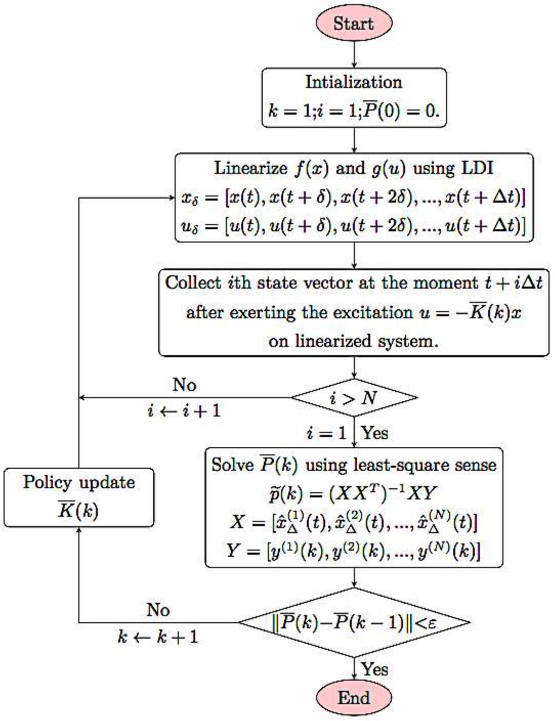

Start
Intialization
$k=1; i=1; \overline{P}(0)=0.$
Linearize $f(x)$ and $g(u)$ using LDI
$x_\delta = [x(t), x(t+\delta), x(t+2\delta), ..., x(t+\Delta t)]$
$u_\delta = [u(t), u(t+\delta), u(t+2\delta), ..., u(t+\Delta t)]$
Collect ith state vector at the moment $t+i\Delta t$ after exerting the excitation $u=-\overline{K}(k)x$ on linearized system.
No
$i \leftarrow i+1$
$i>N$
$i=1$
Yes
Solve $\overline{P}(k)$ using least-square sense
$\widetilde{p}(k) = (XX^T)^{-1}XY$
$X = [\hat{x}_\Delta^{(1)}(t), \hat{x}_\Delta^{(2)}(t), ..., \hat{x}_\Delta^{(N)}(t)]$
$Y = [y^{(1)}(k), y^{(2)}(k), ..., y^{(N)}(k)]$
Policy update
$\overline{K}(k)$
No
$k \leftarrow k+1$
$\|\overline{P}(k) - \overline{P}(k-1)\| < \varepsilon$
Yes
End

Fig. 1. The online implementation of Algorithm 2.

2 will converge to a certain value eventually. Note that $\lim_{k\to\infty} \overline{P}(k) = \overline{P}^*$, where $\overline{P}^*$ denotes the unique positive definite solution of (31) and $\overline{P}^* \in \mathbf{S}$, $\mathbf{S} = \mathbf{S}(\overline{P}) \triangleq \{\overline{P} \in \Re^{n\times n} | \ \|\overline{P}\| < \overline{P}_M^*\}$. This completes the proof. ■

B. Online Implementation

In this subsection, we implement the OLDIPI algorithm in Algorithm 2 without using the knowledge of the system internal dynamics. The important point is that the states of the nonlinear system are required to be observed and updated under a persistent excitation at each iteration. Considering $\overline{A}(k)$ is implicit in states x_t and x_{t+T}, so the system internal dynamic will not be required in PI computation scheme. To facilitate this solution process, we rewrite the left side of (23) as:

$$x_t^{\mathrm{T}} \overline{P}(k) x_t = [\widetilde{p}(k)]^{\mathrm{T}} \hat{x}(t) \tag{35}$$

where $\widetilde{p}(k)$ denotes the column vector made by stacking the elements of the upper triangular part of $\overline{P}(k)$ in order, and the off-diagonal elements of $\overline{P}(k)$ are merged as $2\overline{p}_{ij}$, i.e.,

$$\begin{aligned}\widetilde{p}(k) = [&\overline{p}_{11}(k), 2\overline{p}_{12}(k), ..., 2\overline{p}_{1n}(k),\\ &\overline{p}_{22}(k), 2\overline{p}_{23}(k), ..., \overline{p}_{nn}(k)]^{\mathrm{T}};\end{aligned} \tag{36}$$

and $\hat{x}(t)$ denotes the Kronecker product quadratic polynomial basis vector with the elements $\{x_i(t)x_j(t)\}_{i=1,2,...n; j=i,i+1,...n}$, i.e.,

$$\begin{aligned}\hat{x}(t) = [&x_1^2(t), x_1(t)x_2(t), ..., x_1(t)x_n(t),\\ &x_2^2(t), x_2(t)x_3(t), ..., x_n^2(t)].\end{aligned} \tag{37}$$

By (36) and (37), the kth policy evaluation in (23) can be replaced as:

$$\begin{aligned}&[\widetilde{p}(k)]^{\mathrm{T}}[\hat{x}(t) - \hat{x}(t+T)]\\ &\quad = \int_t^{t+T} x_\tau^{\mathrm{T}}[Q + \overline{K}^{\mathrm{T}}(k) R \overline{K}(k)] x_\tau \mathrm{d}\tau.\end{aligned} \tag{38}$$

It is necessary to point out that, there are $n(n+1)/2$ elements in the symmetric matrix $\overline{P}(k)$ to be solved. As the consequence, we at least need $N(N \geq n(n+1)/2)$ independent equations for solving $\widetilde{p}(k)$. Hence, N state vectors should be sampled in each iteration interval T, then the sampling interval will be $\Delta t = T/N$; ultimately the matrix sequences $\widetilde{p}(k)$ can be obtained by solving the following least-square equation, and it will generate $\overline{P}(k)$ subsequently:

$$\widetilde{p}(k) = (XX^{\mathrm{T}})^{-1} XY \tag{39}$$

where

$$X = [\hat{x}_\Delta^{(1)}(t), \hat{x}_\Delta^{(2)}(t), ..., \hat{x}_\Delta^{(N)}(t)],$$

$$\hat{x}_\Delta^{(i)}(t) = \hat{x}[t + (i-1)\Delta t] - \hat{x}[t + i\Delta t],$$

$$Y = [y^{(1)}(k), y^{(2)}(k), ..., y^{(N)}(k)],$$

$$y^{(i)}(k) = \int_{t+(i-1)\Delta t}^{t+i\Delta t} x_\tau^{\mathrm{T}}[Q + \overline{K}^{\mathrm{T}}(k) R \overline{K}(k)] x_\tau \mathrm{d}\tau.$$

With an adequate collection of status points, the calculation of least-square problem (39) can be online in real-time. Besides, the sampling points (20) for nonlinear terms should be sufficient enough to ensure an admissible accuracy for the implementation. It is remarkable that when the Algorithm 2 is implemented online, the interval T in (20) should be replaced by Δt. The online implementation of Algorithm 2 is shown in Fig. 1.

The online features of OLDIPI algorithm are mainly reflected in the following three aspects:

1) The linear adaptive optimal control is based on the online algorithm. It calculates the control policy after acquiring the sufficient data along the system state trajectory between the time interval T;
2) The LDI approximation is performed online, which linearizes the nonlinear terms by utilizing the data set collected in a short time interval δ, in contrast to the previous off-line LDI approximation, which simply linearizes the system around the origin once before the PI control scheme;
3) By the OLDIPI algorithm, we can online collect and observe the variation of the state trajectory and the control input among the whole solution procedure.

IV. Simulation Result

In this section, two simulation examples are presented to demonstrate the validity and applicability of the proposed adaptive optimal control algorithm.

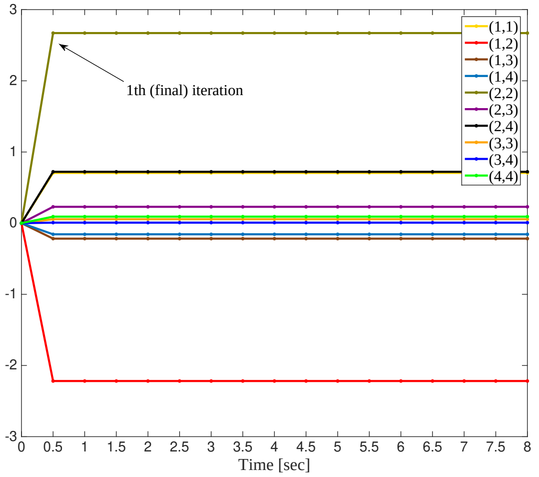


Fig. 2. $\overline{P}$ parameter updated at each iteration step.

A. Example 1

Consider a fourth-order nonlinear system as follows:

$$A=\begin{bmatrix}-2.0000 & 0.8233 & -7.8001 & -2.3333\\ 0.4988 & -6.4123 & -5.2302 & -1.5024\\ 1.0000 & 0.1000 & -10.2007 & -10.2007\\ 0.0000 & -5.2032 & 8.2000 & -4.0000\end{bmatrix}, B=\begin{bmatrix}0\\0\\1\\0\end{bmatrix},$$

$$f(x(t))=\begin{bmatrix}0\\0\\0\\ \sin(0.1x_1(t))\end{bmatrix},$$

$$g(u(t))=\begin{bmatrix}0\\0\\0\\ 0.5\sin^2(u(t))\cos^2(u(t))\end{bmatrix},$$

$$Q=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}, R=1.$$

For the above nonlinear terms, we construct two three-layered neural networks to perform LDI expression simultaneously on $f(x(t))$ and $g(u(t))$. The network approximating $f(x(t))$ contains four neurons in input layer, four in hidden layer and one in output layer, and the other network used to approximate $g(u(t))$ has one neuron in input layer, four in hidden layer and one in output layer. The parameters of the activation function are set as $q=0.5$, $\lambda=1$. The online simulation is designed by using $\Delta t=T/N=0.05$, $\delta=0.001$, with the initial state of the system is $x_0=[0.01 \quad 0 \quad 0.01 \quad 0.01]^{\mathrm{T}}$.

After the online sampling and policy iteration, we obtain the approximate solution by Algorithm 2:

$$\overline{P}=\begin{bmatrix}0.7060 & -1.1090 & -0.1087 & -0.0782\\ -1.1090 & 2.6710 & 0.1149 & 0.3609\\ -0.1087 & 0.1149 & 0.0554 & 0.0036\\ -0.0782 & 0.3609 & 0.0036 & 0.0918\end{bmatrix}. \tag{40}$$

The simulation results are shown in Figs. 2-3. Fig. 2 shows the matrix parameters in $\overline{P}(k)$ after each iteration and verifies the convergence of the proposed algorithm. It can be obviously seen from Fig. 2 that $\overline{P}(k)$ can converge to the optimal solution very quickly after one iteration step. Fig. 3 show the state trajectory and control input curve of the closed loop system respectively. From the online control curve in Fig. 3, we can see the OLDIPI control algorithm is efficient for both two nonlinear terms.

To illustrate the convergence effectiveness of $\Delta A(k)$ and $\Delta B(k)$, we apply our online LDI method and PI algorithm to the system with the same A and B, but let the nonlinear terms as $f(x(t))=0$ and $g(u(t))=0$ (i.e., a linear case). The state trajectory and the control input curve are shown in Fig. 4. Although $\Delta A(k)$ and $\Delta B(k)$ have little effect on the properties of the system in single iteration, after finishing the whole iteration process, they do have an accumulation effect on the convergence of the PI algorithm in prolonging the iteration time within a very acceptable limitation and cause an obvious fluctuation on the system dynamics. However, it is seen from Fig. 4 that the state trajectories of the closed-loop system are finally stabilizable, which also proves the convergence and stability of our designed PI algorithm.

Remark 4: The relevant weight matrices W_i and V_i are obtained after the training on the two three-layered neural networks, and they are updated via back-propagation algorithm by means of the input data (20). Then in the next iteration, W_i and V_i will have different values due to the different input data. Therefore, the values of W_i and V_i are consistently changing in each iteration, making it impractical to display all of them.

B. Example 2

In this subsection, the proposed adaptive optimal control algorithm is used with a nonlinear system in [40], [41] given by

$$\begin{cases}\dot{x}_1=-x_1^3-x_2,\\ \dot{x}_2=x_1+x_2+u.\end{cases} \tag{41}$$

Consider the nonlinear form in system (2), we rewrite nonlinear system (41) as:

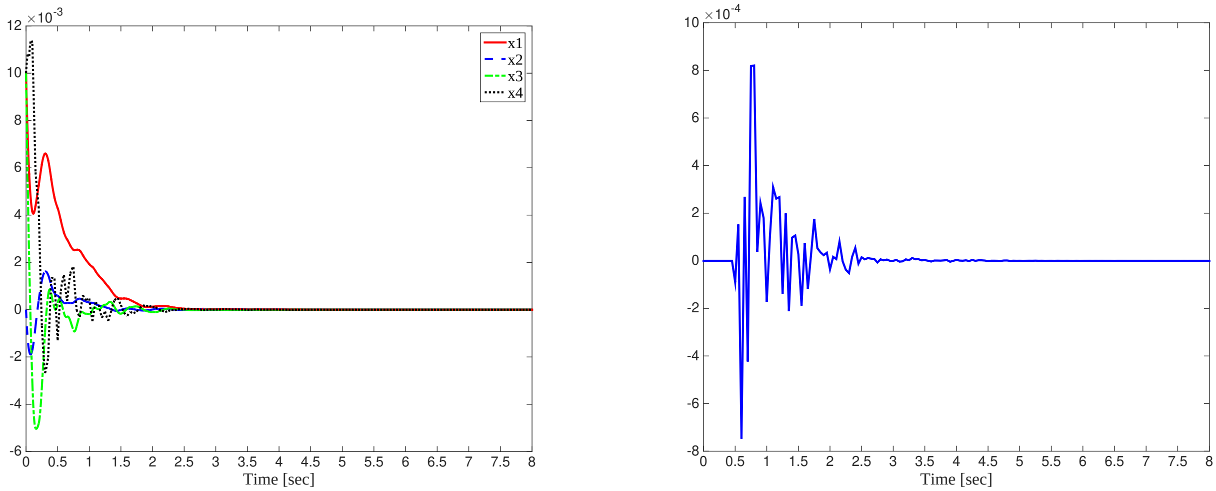
$$\dot{x}(t)=\begin{bmatrix}0&-1\\1&1\end{bmatrix}x(t)+\begin{bmatrix}0\\1\end{bmatrix}u(t)+\begin{bmatrix}-x_1^3(t)\\0\end{bmatrix}. \tag{42}$$

To handle the nonlinear term described in Eq. (42), we suppose a three-layered neural network which has two neurons in input layer, two in hidden layer and one in output layer. The parameters of activation function and online simulation in this part are the same as Example 1. And the initial state is set as $x_0=[1 \ -1]^{\mathrm{T}}$.

Then, we can obtain the optimal controller as:

$$\overline{P}=\begin{bmatrix}0.1296 & 0.0596\\ 0.0596 & 2.0361\end{bmatrix}. \tag{43}$$

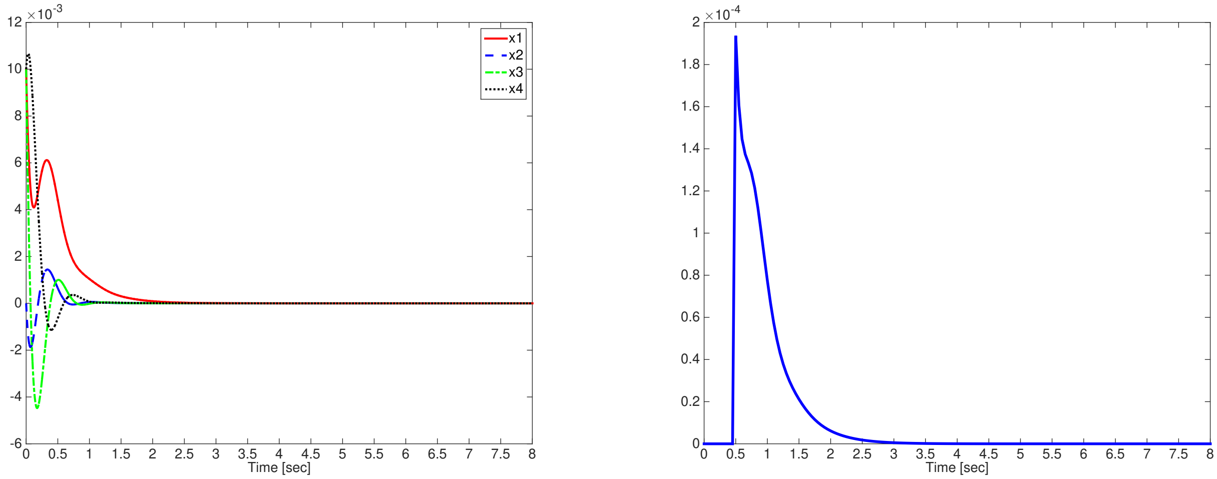
The simulation results by applying the OLDIPI control algorithm are presented in Figs. 5-7. From Fig. 5, it is clear that $\overline{P}(k)$ converges to the optimal solution after two iteration steps within 1 s. In contrast to the method proposed in [41], it employs the neural network without using LDI technique but requires 9 s to reach the final convergence. It demonstrates that our method is obviously more effective. In addition, we obtain a more explicit solution in the research result.



(a) The state trajectory of the closed-loop system by online iteration

(b) The control input curve by online iteration.

Fig. 3. The online control curve for Example 1 ($f(x(t)) = [0, 0, 0, \sin(0.1x_1(t)]^{\mathrm{T}}, g(u(t)) = [0, 0, 0, 0.5\sin^2(u(t))\cos^2(u(t))]^{\mathrm{T}}$).



(a) The state trajectory of the closed-loop system by online iteration

(b) The control input curve by online iteration.

Fig. 4. The online control curve for Example 1 ($f(x(t)) = 0, g(u(t)) = 0$).

V. Conclusion

A novel online PI algorithm to design has been proposed for the adaptive optimal controller of a class of continuous-time nonlinear systems with partially unknown system dynamics. By applying neural network LDI technique, the original nonlinear system can be approximated to a linear plant model accurately. Then, PI algorithm and the optimal solution of the corresponding ARE can be learned.

Inspired by the results in this paper, the new type algorithm can be applied to an akin system that has completely unknown nonlinear terms. The influence of sampling time and the neural network approximation error on research results will be addressed in the future work.

References

[1] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[2] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2012.

[3] P.-L. Lions, *Optimal Control of Diffusion Processes and Hamilton-Jacobi-Bellman Equations: The Dynamic Programming Principles and Applications*. M. Dekker, 1983.

[4] S. Peng, "A generalized dynamic programming principle and hamilton-jacobi-bellman equation," *Stochastics: An International Journal of Probability and Stochastic Processes*, vol. 38, no. 2, pp. 119–134, 1992.

[5] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, no. 10, pp. 1709–1721, 2005.

[6] X. Yang, H. He, D. Liu, and Y. Zhu, "Adaptive dynamic programming for robust neural control of unknown continuous-time non-linear systems," *IET Control Theory and Applications*, vol. 11, no. 14, pp. 2307–2316, 2017.

[7] X. Zhong, H. He, H. Zhang, and Z. Wang, "Optimal control for unknown discrete-time nonlinear Markov jump systems using adaptive dynamic

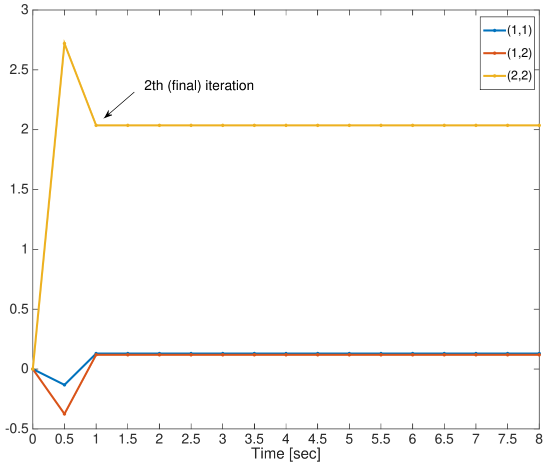


Fig. 5. $\overline{P}$ parameter updated at each iteration step.

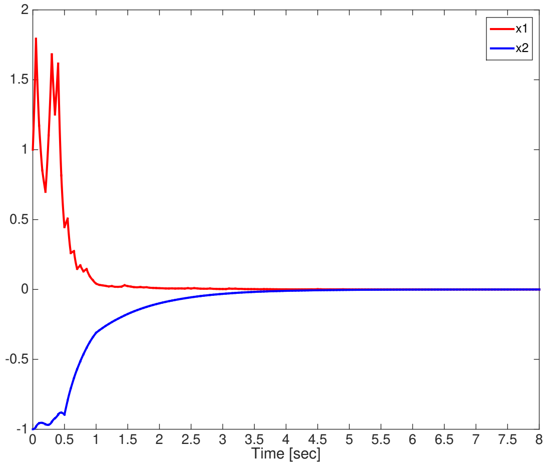


Fig. 6. The state trajectory of the closed-loop system by online iteration.

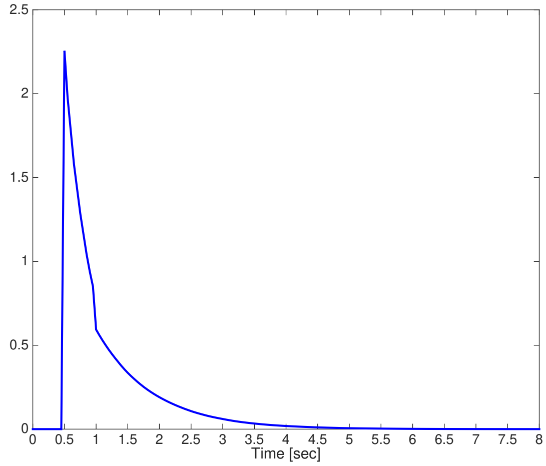


Fig. 7. The control input curve by online iteration.

programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2141–2155, 2014.

[8] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, 2009.

[9] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[10] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.

[11] Y. Jiang and Z.-P. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics," *Automatica*, vol. 48, no. 10, pp. 2699–2704, 2012.

[12] S. He, J. Song, Z. Ding, and F. Liu, "Online adaptive optimal control for continuous-time Markov jump linear systems using a novel policy iteration algorithm," *IET Control Theory and Applications*, vol. 9, no. 10, pp. 1536–1543, 2015.

[13] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 621–634, 2014.

[14] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 10, pp. 1513–1525, 2013.

[15] H. Zhang, H. Liang, Z. Wang, and T. Feng, "Optimal output regulation for heterogeneous multiagent systems via adaptive dynamic programming." *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 1, p. 18, 2017.

[16] H. Zhang, H. Jiang, C. Luo, and G. Xiao, "Discrete-time nonzero-sum games for multiplayer using policy-iteration-based adaptive dynamic programming algorithms," *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3331–3340, 2017.

[17] B. Luo, H.-N. Wu, T. Huang, and D. Liu, "Data-based approximate policy iteration for affine nonlinear continuous-time optimal control design," *Automatica*, vol. 50, no. 12, pp. 3281–3290, 2014.

[18] J. Song, S. He, F. Liu, Y. Niu, and Z. Ding, "Data-driven policy iteration algorithm for optimal control of continuous-time itô stochastic systems with Markovian jumps," *IET Control Theory and Applications*, vol. 10, no. 12, pp. 1431–1439, 2016.

[19] W. Guo, J. Si, F. Liu, and S. Mei, "Policy approximation in policy iteration approximate dynamic programming for discrete-time nonlinear systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 2794–2807, 2018.

[20] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[21] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991.

[22] X. Xu, Z. Huang, L. Zuo, and H. He, "Manifold-based reinforcement learning via locally linear reconstruction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 4, pp. 934–947, 2017.

[23] D. Wang, D. Liu, and H. Li, "Policy iteration algorithm for online design of robust control for a class of continuous-time nonlinear systems," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 627–632, 2014.

[24] D. Vrabie and F. Lewis, "Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems," *Neural Networks*, vol. 22, no. 3, pp. 237–246, 2009.

[25] D. Liu, X. Yang, and H. Li, "Adaptive optimal control for a class of continuous-time affine nonlinear systems with unknown internal dynamics," *Neural Computing and Applications*, vol. 23, no. 7-8, pp. 1843–1850, 2013.

[26] D. Wang, D. Liu, H. Li, and H. Ma, "Neural-network-based robust optimal control design for a class of uncertain nonlinear systems via adaptive dynamic programming," *Information Sciences*, vol. 282, pp. 167–179, 2014.

[27] D. Lu, X. Zhong, C. Sun, and H. He, "Event-triggered adaptive dynamic programming for continuous-time systems with control constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–12, 2016.

[28] K. Tanaka, "An approach to stability criteria of neural-network control systems," *IEEE Transactions on Neural Networks*, vol. 7, no. 3, pp. 629–642, 1996.

[29] S. Limanond and J. Si, "Neural network-based control design: an lmi approach," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1422–1429, 1998.

[30] C.-L. Lin and T.-Y. Lin, "An H_∞ approach for neural net-based control schemes," *IEEE Transactions on Automatic Control*, vol. 46, no. 10, pp. 1599–1605, 2001.

[31] X. Liao, G. Chen, and E. N. Sanchez, "LMI-based approach for asymptotically stability analysis of delayed neural networks," *IEEE Transactions on circuits and systems I: Fundamental Theory and Applications*, vol. 49, no. 7, pp. 1033–1039, 2002.

[32] R. Zhu, M. Sun, and C. Shi, "Absolute stability analysis and design of fuzzy control systems," in *Proc. IEEE International Conference on Networking, Sensing and Control.* IEEE, 2008, pp. 1721–1725.

[33] Y. Wu and T. Shen, "Reach control problem for linear differential inclusion systems on simplices," *IEEE Transactions on Automatic Control*, vol. 61, no. 5, pp. 1403–1408, 2016.

[34] D. Zhai, L. An, D. Ye, and Q. Zhang, "Adaptive reliable H_∞ static output feedback control against Markovian jumping sensor failures," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–14, 2018.

[35] D. Zhai, L. An, X. Li, and Q. Zhang, "Adaptive fault-tolerant control for nonlinear systems with multiple sensor faults and unknown control directions," *IEEE transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 4436–4446, 2018.

[36] Y. J. Liu, S. Li, S. Tong, and C. Chen, "Adaptive reinforcement learning control based on neural approximation for nonlinear discrete-time systems with unknown nonaffine dead-zone input," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–11, 2018.

[37] J. A. Suykens, B. De Moor, and J. Vandewalle, "Robust local stability of multilayer recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 1, pp. 222–229, 2000.

[38] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[39] D. Kleinman, "On an iterative technique for riccati equation computations," *IEEE Transactions on Automatic Control*, vol. 13, no. 1, pp. 114–115, 1968.

[40] R. W. Beard, G. N. Saridis, and J. T. Wen, "Galerkin approximations of the generalized hamilton-jacobi-bellman equation," *Automatica*, vol. 33, no. 12, pp. 2159–2177, 1997.

[41] D. Vrabie and F. L. Lewis, "Adaptive optimal control algorithm for continuous-time nonlinear systems based on policy iteration," in *Proc. 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 73–79.