

A novel stable value iteration-based approximate dynamic programming algorithm for discrete-time nonlinear systems

Yan-Hua Qu(曲延华), An-Na Wang(王安娜), Shen Lin(林盛)
Citation: Chin. Phys. B . 2018, 27(1): 010203. **doi:** 10.1088/1674-1056/27/1/010203

Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Improved control for distributed parameter systems with time-dependent spatial domains utilizing mobile sensor—actuator networks

Jian-Zhong Zhang(张建中), Bao-Tong Cui(崔宝同), Bo Zhuang(庄波)
Chin. Phys. B . 2017, 26(9): 090201. **doi:** 10.1088/1674-1056/26/9/090201

A local energy-preserving scheme for Klein–Gordon–Schrödinger equations

Cai Jia-Xiang, Wang Jia-Lin, Wang Yu-Shun
Chin. Phys. B . 2015, 24(5): 050205. **doi:** 10.1088/1674-1056/24/5/050205

A conservative Fourier pseudospectral algorithm for the nonlinear Schrödinger equation

Lv Zhong-Quan, Zhang Lu-Ming, Wang Yu-Shun
Chin. Phys. B . 2014, 23(12): 120203. **doi:** 10.1088/1674-1056/23/12/120203

A meshless algorithm with moving least square approximations for elliptic Signorini problems

Wang Yan-Chong, Li Xiao-Lin
Chin. Phys. B . 2014, 23(9): 090202. **doi:** 10.1088/1674-1056/23/9/090202

Consensus problems of first-order dynamic multi-agent systems with multiple time delays

Ji Liang-Hao, Liao Xiao-Feng
Chin. Phys. B . 2013, 22(4): 040203. **doi:** 10.1088/1674-1056/22/4/040203

A novel stable value iteration-based approximate dynamic programming algorithm for discrete-time nonlinear systems

Yan-Hua Qu(曲延华)†, An-Na Wang(王安娜), and Sheng Lin(林盛)

College of Information Science and Engineering, Northeastern University, Shenyang 110819, *China*

(Received 4 July 2017; revised manuscript received 11 October 2017; published online 17 November 2017)

The convergence and stability of a value-iteration-based adaptive dynamic programming (ADP) algorithm are considered for discrete-time nonlinear systems accompanied by a discounted quadric performance index. More importantly than sufficing to achieve a good approximate structure, the iterative feedback control law must guarantee the closed-loop stability. Specifically, it is firstly proved that the iterative value function sequence will precisely converge to the optimum. Secondly, the necessary and sufficient condition of the optimal value function serving as a Lyapunov function is investigated. We prove that for the case of infinite horizon, there exists a finite horizon length of which the iterative feedback control law will provide stability, and this increases the practicability of the proposed value iteration algorithm. Neural networks (NNs) are employed to approximate the value functions and the optimal feedback control laws, and the approach allows the implementation of the algorithm without knowing the internal dynamics of the system. Finally, a simulation example is employed to demonstrate the effectiveness of the developed optimal control method.

Keywords: adaptive dynamic programming (ADP), convergence, stability, discounted quadric performance index

PACS: 02.60.Gf, 02.30.Jr, 02.30.Yy **DOI:** 10.1088/1674-1056/27/1/010203

1. Introduction

Optimal control of the nonlinear system has been viewed as the key problem in the control field in the last few decades. As is known, dynamic programming (DP)[1] has emerged as a powerful technique to solve optimal control problems. However, due to the curse of dimensionality, it is computationally untenable to obtain the optimal solution. For nonlinear systems, we can obtain the optimal stabilizing feedback control law by solving the Hamilton–Jacobi–Bellman (HJB) equation. However, due to its inherently nonlinear nature, it is challenging to obtain the solution. For linear systems, the HJB equation reduces to the well-known Riccati equation, which results in the linear quadratic regulator (LQR) controller.[2]

To overcome this difficulty, adaptive dynamic programming (ADP) was firstly proposed by werbos[3] for the discrete-time (DT) dynamic system, which is a reinforcement learning (RL) approach and utilizes the function approximation for the value functions based on adaptive critics. Different from DP, ADP solves the optimal control problems forward-in-time. Bertsekas and Tsitsiklis proposed a neuro-dynamic programming (NDP)[4] algorithm, where neural networks are employed to approximate the value function. Werbos classified ADP algorithms into the following four main schemes: heuristic dynamic programming (HDP), dual HDP (DHP), action-dependent HDP (ADHDP), and action-dependent dual HDP (ADDHP). Prokhorov and Wunsch proposed a new ADP scheme which is known as the globalized DHP (GDHP)[5] and action-dependent GDHP. There are mainly two recursive ADP algorithms: the policy iteration (PI) algorithm and the value iteration (VI) algorithm.[6]

The PI algorithm contains two steps: the policy evaluation and the policy improvement. The iterative value functions and control laws are obtained iteratively, and the iterative control laws must stabilize the system.[7–24] An initial stabilizing control law is required, however, it is often difficult to obtain. While in most applications, fewer iterations are required, and computationally demanding is more than that of the VI iteration algorithm. Abu-Khalaf *et al.*[25] utilized a neural network (NN) to obtain an approximate solution to the value function for continuous-time (CT) nonlinear systems with saturating actuators. Zhang *et al.*[26] proposed an iterative ADP method to solve a class of nonlinear zero-sum (ZS) differential games. Generalized policy iteration (GPI) for CT systems, which includes PI, VI, and optimistic policy iteration (OPI) algorithms, was proposed in Ref. [27]. Data-based optimal control problems using novel PI and VI algorithms were implemented online by Frank *et al.*[28] Song *et al.* developed a novel off-policy integral reinforcement learning method to solve the nonlinear continuous-time multiplayer nonzero-sum games with completely unknown dynamics.[29]

For the VI algorithm, the initial stabilizing control law is not needed. However, the closed-loop stability cannot be guaranteed using the iterative control laws. Luo *et al.*[30] proposed a near-optimal control approach for a class of DT affine nonlinear systems under control constraints by the iterative DHP method. Zhang *et al.*[31] proposed an infinite-time optimal

†Corresponding author. E-mail: quyanhuawang@sina.com

© 2018 Chinese Physical Society and IOP Publishing Ltd

tracking control method for the DT nonlinear system based on the greedy HDP algorithm. Al-Tamimi *et al.*[32] derived a VI-based HDP algorithm to solve the optimal control for the DT nonlinear system and provided the convergence proof. However, the stabilizing control only can be obtained until the value function sequence converges and the iterative control laws are not necessarily stabilizing, which greatly limits the applications of the VI algorithm. Li *et al.*[33] proposed a general value iteration (GVI) algorithm and proved that if the condition $V_0(\boldsymbol{x}_k) \geq V_1(\boldsymbol{x}_k)$ holds, all the iterative control laws are stabilizing; Wei *et al.*[34] proposed a stable θ-ADP scheme; but both of them need an initial value which is also difficult to obtain. All the works above focus on the case of the discount factor being 1. Wang *et al.*[35] developed a VI algorithm to solve the non-affine nonlinear DT systems with a discount factor in the cost function via the GDHP algorithm, nevertheless the stability of the closed-loop is not considered. For discrete LTI systems without discount factor, Primbs *et al.*[36] demonstrated that there always exists a finite i^*, and the closed-loop system is asymptotically stable for all $i \geq i^*$. It should be pointed out that proving convergence of an ADP algorithm is different than showing the closed-loop stability. Ensuring optimality must guarantee the stability of the nonlinear systems.

It is noted that the HJB equation with the discount factor will not always serve as a Lyapunov equation when the discount factor is less than 1, which implies that even the optimal value function may not serve as a Lyapunov function and the optimal control cannot guarantee the closed-loop stability. The primary objective of this paper is to investigate the necessary and sufficient condition for the HJB equation with the discount factor serving as a Lyapunov function, therefore the optimal control is stabilizing. In the following, the stability for a finite horizon formulation of value iteration of DT nonlinear systems with a discount factor in the cost function is considered. We prove that given any initial value function for the infinite horizon problem, there must exist a finite horizon length such that the obtained iterative control laws will ensure the stability, which will greatly increase the practicability of the VI algorithm.

The rest of the paper is organized as follows. Section 2 shows the problem formulation as well as the DT HJB equation with discount factor for the nonlinear systems. Section 3 proposes a novel convergence proof and the necessary and sufficient condition for the HJB equation with the discount factor serving as a Lyapunov function, then the finite numbers of the iteration stability proof of the VI algorithm are introduced. In Section 4, neural networks are utilized to implement the VI algorithm that provides stabilizing control laws. In the last section, simulation examples are employed to demonstrate the effectiveness of the proposed algorithms.

2. Preliminaries

Let us consider the affine DT nonlinear system of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{g}(\boldsymbol{x}_k)\boldsymbol{u}(\boldsymbol{x}_k),\ k = 1,2,\ldots, \tag{1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ is the state vector, and $\boldsymbol{u}(\boldsymbol{x}_k) \in \mathbb{R}^m$ is the input vector. Suppose that the system is drift free and, without loss of generality, $\boldsymbol{x} = 0$ is an equilibrium state of the system under the control $\boldsymbol{u}_k = 0$.

Definition 1 A nonlinear dynamical system is defined to be stabilizable on a compact set $\boldsymbol{\Omega} \in \mathbb{R}^n$ if there exists a control input $\boldsymbol{u} \in \mathbb{R}^m$ such that, for all initial conditions $\boldsymbol{x}_0 \in \boldsymbol{\Omega}$, the state $\boldsymbol{x}_k \to 0$ as $k \to \infty$.

The main objective is to find the control law $\boldsymbol{u}(\boldsymbol{x}_k)$ that minimizes the following infinite-horizon cost function:

$$V(\boldsymbol{x}_k, \boldsymbol{u}) = \sum_{j=k}^{\infty} \alpha^{j-k} U(\boldsymbol{x}_j, \boldsymbol{u}(\boldsymbol{x}_j)), \tag{2}$$

in which U is the positively defined utility function, i.e., $U(0,0) = 0$ and $U(\boldsymbol{x}_k, \boldsymbol{u}(\boldsymbol{x}_k)) \geq 0, \forall \boldsymbol{x}_k,\ \boldsymbol{u}(\boldsymbol{x}_k)$, α is the discount factor with $0 < \alpha \leq 1$. Note that the control law needs not only to be stabilizing, but also to guarantee that equation (2) is finite, that is, the control law must be admissible.

Definition 2[17] A control $\boldsymbol{u}(\boldsymbol{x}_k)$ is defined to be admissible with respect to Eq. (2) on $\boldsymbol{\Omega}$ if $\boldsymbol{u}(\boldsymbol{x}_k)$ is continuous on a compact set $\boldsymbol{\Omega} \subseteq \mathbb{R}^n$, $\boldsymbol{u}(0) = 0$, $\boldsymbol{u}(\boldsymbol{x})$ stabilizes Eq. (1) on $\boldsymbol{\Omega}$, and for all $\boldsymbol{x}_0 \in \boldsymbol{\Omega}$, $J(\boldsymbol{x}_0, \boldsymbol{u})$ is finite.

It is readily seen that equation (2) can be written as

$$V(\boldsymbol{x}_k) = U(\boldsymbol{x}_k, \boldsymbol{u}_k) + \alpha \sum_{j=k+1}^{\infty} \alpha^{j-k-1} U(\boldsymbol{x}_j, \boldsymbol{u}_j). \tag{3}$$

Following the Bellman's optimal principle, the optimal cost function $V^*(\boldsymbol{x}_k)$ must satisfy the DT HJB equation

$$V^*(\boldsymbol{x}_k) = \min_{\boldsymbol{u}(\boldsymbol{x}_k)} \{U(\boldsymbol{x}_k, \boldsymbol{u}_k) + \alpha V^*(\boldsymbol{x}_{k+1})\}. \tag{4}$$

Remark 1 Note that equation (4) is a Lyapunov equation when the discount factor $\alpha = 1$. However, for some $\alpha < 1$, equation (4) may not serve as a Lyapunov equation, and the optimal control cannot guarantee the closed-loop stability, which will be illustrated by the numerical simulation example in Appendix A.

In this paper, the utility function U is chosen as the quadric form, i.e., $U(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{x}_k^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{R} \boldsymbol{u}_k$. By noting the fact that $\boldsymbol{u}^*$ satisfies the first-order necessary condition, taking gradient of the right-hand side of Eq. (4) with respect to $\boldsymbol{u}$ yields

$$\frac{\partial (\boldsymbol{x}_k^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{R} \boldsymbol{u}_k)}{\partial \boldsymbol{u}_k} + \alpha \left(\frac{\partial \boldsymbol{x}_{k+1}}{\partial \boldsymbol{u}_k} \right)^{\mathrm{T}} \frac{\partial V^*(\boldsymbol{x}_{k+1})}{\partial \boldsymbol{x}_{k+1}} = 0, \tag{5}$$

then we obtain

$$\boldsymbol{u}^*(\boldsymbol{x}_k) = -\frac{\alpha}{2} \boldsymbol{R}^{-1} \boldsymbol{g}(\boldsymbol{x}_k)^{\mathrm{T}} \frac{\partial V^*(\boldsymbol{x}_{k+1})}{\partial \boldsymbol{x}_{k+1}}. \tag{6}$$

Substituting Eq. (6) in Eq. (4), we have

$$V^*(\boldsymbol{x}_k) = \boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \frac{\alpha^2}{4}\left(\frac{\partial V^*(\boldsymbol{x}_{k+1})}{\partial \boldsymbol{x}_{k+1}}\right)^{\mathrm{T}} \boldsymbol{g}(\boldsymbol{x}_k)\boldsymbol{R}^{-1}\boldsymbol{g}(\boldsymbol{x}_k)^{\mathrm{T}} \times \frac{\partial V^*(\boldsymbol{x}_{k+1})}{\partial \boldsymbol{x}_{k+1}} + \alpha V^*(\boldsymbol{x}_{k+1}). \tag{7}$$

The above equation reduces to the algebraic Riccati equation (ARE) for the linear systems accompanied by a quadratic cost function (LQR). However, for general nonlinear systems, equation (7) is difficult to solve directly because it involves solving a nonlinear partial difference equation. To overcome the difficulty in solving the HJB equation, recursive methods will be proposed to solve the HJB equation indirectly.

2.1. Discrete-time value iteration algorithm

Here we show the value iteration algorithm. We can start with any initial value $V_0(\boldsymbol{x}_k) \geq 0$, then solve for $\boldsymbol{u}_0$ from

$$\boldsymbol{u}_0(\boldsymbol{x}_k) = \arg\min_{\boldsymbol{u}(\boldsymbol{x}_k)} \left\{\boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}(\boldsymbol{x}_k) + \alpha V_0(\boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{g}(\boldsymbol{x}_k)\boldsymbol{u}(\boldsymbol{x}_k))\right\}.$$

Then, the cost function is updated as

$$V_1(\boldsymbol{x}_k) = \boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}_0(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}_0(\boldsymbol{x}_k) + \alpha V_0(\boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{g}(\boldsymbol{x}_k)\boldsymbol{u}_0(\boldsymbol{x}_k)).$$

The value iteration algorithm runs between the following two iterative steps:

Value update: update the value function by

$$V_{i+1}(\boldsymbol{x}_k) = \boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}_i(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}_i(\boldsymbol{x}_k) + \alpha V_i(\boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{g}(\boldsymbol{x}_k)\boldsymbol{u}_i(\boldsymbol{x}_k)); \tag{8}$$

Policy improvement: determine the improved policy by

$$\boldsymbol{u}_{i+1}(\boldsymbol{x}_k) = \arg\min_{\boldsymbol{u}(\boldsymbol{x}_k)} \left\{\boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}(\boldsymbol{x}_k) + \alpha V_{i+1}(\boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{g}(\boldsymbol{x}_k)\boldsymbol{u}(\boldsymbol{x}_k))\right\}. \tag{9}$$

In the above recurrent iteration, the cost functions and control laws are updated until they convert to the optimal values. In Ref. [32], the convergence has been proved for $\alpha = 1$, as i goes to the infinite, it follows

$$V_\infty(\boldsymbol{x}_k) = \min_{\boldsymbol{u}_k}\left\{\boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}_k^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}_k + V_\infty(\boldsymbol{x}_{k+1})\right\} = \boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}_\infty^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}_\infty + V_\infty(\boldsymbol{x}_{k+1}), \tag{10}$$

and the optimal control law is given by

$$\boldsymbol{u}_\infty(\boldsymbol{x}_k) = \arg\min_{\boldsymbol{u}_k}\left\{\boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}_k^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}_k + V_\infty(\boldsymbol{x}_{k+1})\right\}.$$

3. The main results

In the above section, one can see that equation (10) naturally serves as a Lyapunov equation, hence the control law $\boldsymbol{u}_\infty(\boldsymbol{x}_k)$ is stabilizing. However, it is not acceptable to execute infinite iterative steps in practical applications. Thus the termination condition of the algorithm is proposed, i.e., if $|V_{i+1}((\boldsymbol{x}_k)) - V_i((\boldsymbol{x}_k))| < \epsilon$ holds, then $\boldsymbol{u}_i(\boldsymbol{x}_k)$ will be considered as the approximate optimal controller, but whether $\boldsymbol{u}_i(\boldsymbol{x}_k)$ is stabilizing still remains unsolved. Therefore, finding a finite horizon length of which the iterative control law will provide the stability is the crucial issue. Moreover, from the discussion in Remark 1, there must be some restrictions on the selection of the discounted factor. Otherwise, by choosing an inappropriate discounted factor, the optimal control law will make the resulting closed system unstable. To overcome these two difficulties, we propose the following theorem, which is inspired by the innovative works.[33,38,39]

Theorem 1 Suppose that the condition

$$0 \leq \alpha V^*(\boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{g}(\boldsymbol{x}_k)\boldsymbol{u}(\boldsymbol{x}_k)) \leq \theta U(\boldsymbol{x}_k, \boldsymbol{u}(\boldsymbol{x}_k)) \tag{11}$$

holds uniformly for some positive scalar $0 < \theta < \infty$, and there are $0 \leq \delta \leq 1$ and $1 \leq \omega \leq \infty$ such that $0 \leq \delta V^* \leq V_0 \leq \omega V^*$, $0 < \alpha \leq 1$. The value function sequence $\{V_i\}$ and the control law sequence $\{\boldsymbol{u}_i\}$ are respectively iteratively updated by Eqs. (2) and (1). Then the iterative value function V_i approaches V^* according to the following inequality:

$$\left[1 + \frac{\delta - 1}{(1+\theta^{-1})^i}\right] V^*(\boldsymbol{x}_k) \leq V_i(\boldsymbol{x}_k) \leq \left[1 + \frac{\omega - 1}{(1+\theta^{-1})^i}\right] V^*(\boldsymbol{x}_k). \tag{12}$$

Define $V_\infty(\boldsymbol{x}_k) = \lim_{i\to\infty} V_i(\boldsymbol{x}_k)$ and $\boldsymbol{u}_\infty(\boldsymbol{x}_k) = \lim_{i\to\infty} \boldsymbol{u}_i(\boldsymbol{x}_k)$, then $V_\infty(\boldsymbol{x}_k) = V^*(\boldsymbol{x}_k)$, $\boldsymbol{u}_\infty(\boldsymbol{x}_k) = \boldsymbol{u}^*(\boldsymbol{x}_k)$. Moreover, if the discount factor satisfies $\theta(1+\theta)^{-1} < \alpha \leq 1$, then there exists a finite index i^* such that for $i \geq i^*$, the iterative control laws are asymptotically stabilizing for any given initial value function.

Proof The assumption $0 \leq \alpha V^*(f(\boldsymbol{x}, \boldsymbol{u})) \leq \theta U(\boldsymbol{x}, \boldsymbol{u})$ implies that

$$\frac{\delta - 1}{1+\theta}(\theta U(\boldsymbol{x}_k, \boldsymbol{u}(\boldsymbol{x}_k)) - \alpha V^*(\boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{g}(\boldsymbol{x}_k)\boldsymbol{u}(\boldsymbol{x}_k)) \leq 0. \tag{13}$$

The left hand side of inequality (9) will be demonstrated by mathematical induction. For $i = 1$,

$$\begin{aligned} V_1(\boldsymbol{x}) &= \min_u[\alpha V_0(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) + U(\boldsymbol{x}, \boldsymbol{u})] \\ &\geq \min_u[\alpha\delta V^*(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) + U(\boldsymbol{x}, \boldsymbol{u})] \\ &\geq \min_u\left[\left(\delta - \frac{\delta - 1}{\theta + 1}\right)\alpha V^*(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) + \left(1 + \theta\frac{\delta - 1}{\theta + 1}\right)U(\boldsymbol{x}, \boldsymbol{u})\right] \\ &= \frac{1 + \delta\theta}{\theta + 1}\min_u[\alpha V^*(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) + U(\boldsymbol{x}, \boldsymbol{u})] \\ &= \left[1 + \frac{\delta - 1}{1 + \theta^{-1}}\right] V^*(\boldsymbol{x}). \end{aligned}$$

Assume that for $k-1$, it holds that

$$\left[1+\frac{\delta-1}{(1+\theta^{-1})^{k-1}}\right]V^* \le V_{k-1},$$

then we have

$$\begin{aligned}
V_k(\boldsymbol{x}) &= \min_{u}[\alpha V_{k-1}(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}))+U(\boldsymbol{x},\boldsymbol{u})]\\
&\ge \min_{u}\left\{\left[1+\frac{\delta-1}{(1+\theta^{-1})^{k-1}}\right]\alpha V^*(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}))+U(\boldsymbol{x},\boldsymbol{u})\right\}\\
&\ge \min_{u}\left\{\left[1+\frac{(\delta-1)\theta^k}{(\theta+1)^k}\right]U(\boldsymbol{x},\boldsymbol{u})\right.\\
&\quad\left.+\left[1+\frac{\delta-1}{(\theta^{-1}+1)^{k-1}}-\frac{(\delta-1)\theta^{k-1}}{(\theta+1)^k}\right]\alpha V^*(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}))\right\}\\
&= \left[1+\frac{(\delta-1)\theta^k}{(\theta+1)^k}\right]\min_{u}[\alpha V^*(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}))+U(\boldsymbol{x},\boldsymbol{u})]\\
&= \left[1+\frac{\delta-1}{(1+\theta^{-1})^k}\right]V^*(\boldsymbol{x}).
\end{aligned}$$

The left hand side of inequality (9) has been proved and by the same way, the right-hand side can easily be derived. As k goes to infinity, it is obvious that

$$\lim_{i\to\infty}\left[1+\frac{\delta-1}{(1+\theta^{-1})^i}\right]V^*(\boldsymbol{x}) = V^*(\boldsymbol{x}),$$
$$\lim_{i\to\infty}\left[1+\frac{\omega-1}{(1+\theta^{-1})^i}\right]V^*(\boldsymbol{x}) = V^*(\boldsymbol{x}).$$

Therefore, $V_\infty(\boldsymbol{x}_k)=V^*(\boldsymbol{x}_k)$, $\boldsymbol{u}_\infty(\boldsymbol{x}_k)=\boldsymbol{u}^*(\boldsymbol{x}_k)$.

Now we prove the second part of the theorem. If $\theta(1+\theta)^{-1}<\alpha\le 1$, then according to inequality (11), we have

$$(1-\alpha)V^*(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}))<U(\boldsymbol{x},\boldsymbol{u}),\ \ \boldsymbol{x}\ne\boldsymbol{0},\ \boldsymbol{u}\ne\boldsymbol{0}. \tag{14}$$

By reviewing Eq. (4), we have

$$V^*(\boldsymbol{x}_{k+1})-V^*(\boldsymbol{x}_k)=(1-\alpha)V^*(\boldsymbol{x}_{k+1})-U(\boldsymbol{x},\boldsymbol{u})<0,$$

hence the HJB equation (4) serves as a Lyapunov equation.

Since $V^*(\boldsymbol{x}_k)<(1-\alpha)^{-1}U(\boldsymbol{x}_k,\boldsymbol{u}^*(\boldsymbol{x}_k))$, which is equivalent to $\exists\lambda\in(0,1)$ such that

$$V^*(\boldsymbol{x}_k)\le\frac{\lambda}{1-\alpha}U(\boldsymbol{x}_k,\boldsymbol{u}^*(\boldsymbol{x}_k)). \tag{15}$$

If $0\le V_0(\boldsymbol{x}_k)\le V^*(\boldsymbol{x}_k)$, then we have

$$\left[1-\left(\frac{\alpha\lambda}{1-\alpha+\alpha\lambda}\right)^i\right]V^*(\boldsymbol{x}_k)\le V_i(\boldsymbol{x}_k)\le V^*(\boldsymbol{x}_k), \tag{16}$$

thus it follows

$$\begin{aligned}
&\alpha(V_i(\boldsymbol{x}_{k+1})-V_i(\boldsymbol{x}_k))\\
&=V_{i+1}(\boldsymbol{x}_k)-\alpha V_i(\boldsymbol{x}_k)-U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k))\\
&\le J^*(\boldsymbol{x}_k)-\alpha\left[1-\left(\frac{\alpha\lambda}{1-\alpha+\alpha\lambda}\right)^i\right]V^*(\boldsymbol{x}_k))-U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k)\\
&=\left[1-\alpha+\left(\frac{\alpha\lambda}{1-\alpha+\alpha\lambda}\right)^i\right]V^*(\boldsymbol{x}_k)-U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k))\\
&\le\frac{\lambda}{1-\alpha}\left[1-\alpha+\left(\frac{\alpha\lambda}{1-\alpha+\alpha\lambda}\right)^i\right]U(\boldsymbol{x}_k,\boldsymbol{u}^*(\boldsymbol{x}_k))\\
&\quad-U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k)).
\end{aligned} \tag{17}$$

As $i\to\infty$, $\boldsymbol{u}_i(\boldsymbol{x}_k)\to\boldsymbol{u}^*(\boldsymbol{x}_k)$, so there exists $i_1\in\mathbb{N}_+$, s.t. $i>i_1$, it holds that

$$U(\boldsymbol{x}_k,\boldsymbol{u}^*(\boldsymbol{x}_k))\le(1+\epsilon)U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k)), \tag{18}$$

where $\lambda(1+\epsilon)<1$. Then we obtain

$$\begin{aligned}
\alpha(V_i(\boldsymbol{x}_{k+1})-V_i(\boldsymbol{x}_k))\le&\left\{\frac{\lambda}{1-\alpha}\left[1-\alpha+\left(\frac{\alpha\lambda}{1-\alpha+\alpha\lambda}\right)^i\right]\right.\\
&\left.\times(1+\epsilon)-1\right\}U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k)).
\end{aligned} \tag{19}$$

For $\boldsymbol{x}_k\ne 0$, then $U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k))>0$, we only need to prove $\exists\, i_2\in\mathbb{N}_+$, $i>i_2$,

$$\frac{\lambda}{1-\alpha}\left[1-\alpha+\left(\frac{\alpha\lambda}{1-\alpha+\alpha\lambda}\right)^i\right](1+\epsilon)-1<0, \tag{20}$$

which is equivalent to

$$\left(\frac{\alpha\lambda}{1-\alpha+\alpha\lambda}\right)^i<\{[\lambda(1+\epsilon)]^{-1}-1\}(1-\alpha). \tag{21}$$

We obtain i_2 as

$$i_2>\frac{\ln\{[\lambda(1+\epsilon)]^{-1}-1\}(1-\alpha)}{\ln\alpha\lambda-\ln(1-\alpha+\alpha\lambda)}. \tag{22}$$

Therefore, if the iterative index $i\ge i^*=\max\{i_1,i_2\}$, then $V_i(\boldsymbol{x}_{k+1})-V_i(\boldsymbol{x}_k)<0$. According to the Lyapunov stability criteria, $V_i(\boldsymbol{x}_k)$ is a Lyapunov function, and $\boldsymbol{u}_i(\boldsymbol{x}_k)$ is asymptotically stable.

For the case of $V_0(\boldsymbol{x}_k)\ge V^*(\boldsymbol{x}_k)$, the proof can be given by the same way and is omitted here. This completes the proof.

Theorem 2 The i-th iterative control $\boldsymbol{u}_i(\boldsymbol{x}_k)$ is stabilizing if

$$V_{i+1}(\boldsymbol{x}_k)-\alpha V_i(\boldsymbol{x}_k)<U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k)), \tag{23}$$

where α is the discount factor.

Proof From theorem 1, $\lim_{i\to\infty}V_i(\boldsymbol{x}_k)=V^*(\boldsymbol{x}_k)$, which implies

$$\lim_{i\to\infty}[V_{i+1}(\boldsymbol{x}_k)-\alpha V_i(\boldsymbol{x}_k)]=(1-\alpha)V^*(\boldsymbol{x}_k). \tag{24}$$

For $\lim_{i\to\infty}U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k))=U(\boldsymbol{x}_k,\boldsymbol{u}^*(\boldsymbol{x}_k))$ holds and according to Remark 2, we have $\exists i^*$, $i\ge i^*$, $V_{i+1}(\boldsymbol{x}_k)-\alpha V_i(\boldsymbol{x}_k)<U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k))$, thus

$$\begin{aligned}
&\alpha[V_i(\boldsymbol{x}_{k+1})-V_i(\boldsymbol{x}_k)]\\
&=V_{i+1}(\boldsymbol{x}_k)-\alpha V_i(\boldsymbol{x}_k)-U(\boldsymbol{x}_k,\boldsymbol{u}_i(\boldsymbol{x}_k))<0.
\end{aligned} \tag{25}$$

For $i\ge i^*$, V_i is a Lyapunov function. According to the Lyapunov stability criteria, the $\boldsymbol{u}_i(\boldsymbol{x}_k)$ law is asymptotically stabilizing. The proof is completed.

Remark 3 Theorem 2 illustrates that there exists a finite index i^*, such that $\boldsymbol{u}_i(\boldsymbol{x}_k)$ is stabilizing when $i > i^*$. However, i^* is difficult to accurately obtain. Theorem 2 proposes a sufficient condition for the i-th iterative control law to be stabilizing. It can be easily used to verify whether $\boldsymbol{u}_i(\boldsymbol{x}_k)$ is stabilizing instead of obtaining i^*. Therefore, inequality (23) can be conveniently used in the algorithm design, which gives the termination condition to determine whether $\boldsymbol{u}_i(\boldsymbol{x}_k)$ is stabilizing. If the termination condition $|V_{i+1}(\boldsymbol{x}) - V_i(\boldsymbol{x})| < \epsilon$ is satisfied, we cannot guarantee that $\boldsymbol{u}_i$ is stabilizing, because we do not know whether $i > i^*$ holds. Then, we use the sufficient condition (23) to determine whether $\boldsymbol{u}_i$ is stabilizing. Only if these two termination conditions hold simultaneously, then we obtain the stabilizing approximate optimal controller.

Remark 4 Compared with the existing work[25,26,29] without the discounted factor, the closed-loop stability is proved by introducing the PI algorithm and the algorithm starts with an admissible control. Song *et al.* developed a novel off-policy integral reinforcement learning method to solve the nonlinear continuous-time multiplayer nonzero-sum games with completely unknown dynamics,[29] the asymptotic stability of the closed-loop system and the existence of Nash equilibrium were proved but the stabilizing initial policies must be given. This is very difficult for the nonlinear systems. However, our work firstly proposes the closed-loop stability proof by using the VI algorithm, a definite advantage is that the admissible control is not required. Moreover, by introducing an appropriate discount factor, the convergence rate of the algorithm will be significantly improved, which will be shown in the simulation part.

4. Implementation of value iteration algorithm via neural networks

It is well known that neural networks are usually used to approximate smooth functions on a prescribed compact set.[4,5] At each step, the cost function $V_i(\boldsymbol{x})$ is approximated by

$$\hat{V}_i(\boldsymbol{x}) = (\boldsymbol{W}_i^L)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{j=1}^{L} \omega_i^j \phi_j(\boldsymbol{x}), \tag{26}$$

and $\boldsymbol{u}_i(\boldsymbol{x}_k)$ is approximated by

$$\hat{\boldsymbol{u}}_i(\boldsymbol{x}) = (\boldsymbol{\Theta}_i^L)^{\mathrm{T}}\boldsymbol{\sigma}(\boldsymbol{x}) = \sum_{j=1}^{L} \theta_i^j \sigma_j(\boldsymbol{x}), \tag{27}$$

where $\phi_j(\boldsymbol{x}) : \Omega \to \mathbb{R}$, $\phi_j(\boldsymbol{x})|_{\boldsymbol{x}=0} = 0$, the neural network weights of the i-th step are ω_i^j, and L is the number of hidden layer neurons, $\boldsymbol{\phi}(\boldsymbol{x}) \equiv [\phi_1(\boldsymbol{x})\phi_2(\boldsymbol{x})\cdots\phi_L(\boldsymbol{x})]^{\mathrm{T}}$, $\boldsymbol{W}_i^L \equiv [\omega_i^1(\boldsymbol{x})\omega_i^2(\boldsymbol{x})\cdots\omega_i^L(\boldsymbol{x})]^{\mathrm{T}}$, $\boldsymbol{\sigma}(\boldsymbol{x}) \equiv [\boldsymbol{\sigma}_1(\boldsymbol{x})\boldsymbol{\sigma}_2(\boldsymbol{x})\cdots\boldsymbol{\sigma}_L(\boldsymbol{x})]^{\mathrm{T}}$, $\boldsymbol{\Theta}_i^L \equiv [\boldsymbol{\theta}_i^1(\boldsymbol{x})\boldsymbol{\theta}_i^2(\boldsymbol{x})\cdots\boldsymbol{\theta}_i^L(\boldsymbol{x})]^{\mathrm{T}}$.

Note that

$$\begin{aligned} &E(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, \boldsymbol{W}_i^L, \hat{\boldsymbol{u}}_i(\boldsymbol{x}_k)) \\ &= \boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \hat{\boldsymbol{u}}_i^{\mathrm{T}}(\boldsymbol{x}_k)\boldsymbol{R}\hat{\boldsymbol{u}}_i(\boldsymbol{x}_k) + \alpha\hat{V}_i(\boldsymbol{x}_{k+1}) \\ &= \boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \hat{\boldsymbol{u}}_i^{\mathrm{T}}(\boldsymbol{x}_k)\boldsymbol{R}\hat{\boldsymbol{u}}_i(\boldsymbol{x}_k) + \alpha(\boldsymbol{W}_i^L)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_{k+1}). \end{aligned} \tag{28}$$

Then we obtain the residual error

$$e_L = (\boldsymbol{W}_{i+1}^L)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_k) - E(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, \boldsymbol{W}_i^L, \hat{\boldsymbol{u}}_k). \tag{29}$$

The weighted residuals method is used to find the least-squares solution, as a result, one has

$$\left\langle \frac{\partial e_L(\boldsymbol{x})}{\partial \boldsymbol{W}_i^L}, e_L(\boldsymbol{x}) \right\rangle = 0, \tag{30}$$

where $\langle \boldsymbol{f}, \boldsymbol{g} \rangle = \int_\Omega \boldsymbol{f}\boldsymbol{g}^{\mathrm{T}}\mathrm{d}\boldsymbol{x}$ is a Lebesgue integral. Equation (30) can be expanded as

$$\begin{aligned} 0 = \int_\Omega \boldsymbol{\phi}(\boldsymbol{x}_k)\Big(&\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{x}_k)\boldsymbol{W}_{i+1}^L \\ &- \boldsymbol{E}^{\mathrm{T}}(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, \boldsymbol{W}_i^L, \hat{\boldsymbol{u}}_i(\boldsymbol{x}_k))\Big)\mathrm{d}\boldsymbol{x}_k. \end{aligned} \tag{31}$$

Selecting $\{\phi_j(\boldsymbol{x})\}^L$ that are linearly independent of the compact set $\Omega \in \mathbb{R}^n$, then $\int_\Omega \boldsymbol{\phi}(\boldsymbol{x}_k)\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_k$ is invertible, so the unique solution $\boldsymbol{W}_{i+1}^L$ exists and is given by

$$\begin{aligned} \boldsymbol{W}_{i+1}^L = &\left(\int_\Omega \boldsymbol{\phi}(\boldsymbol{x}_k)\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_k\right)^{-1} \\ &\times \int_\Omega \boldsymbol{\phi}(\boldsymbol{x}_k)\boldsymbol{E}^{\mathrm{T}}\left(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, \boldsymbol{W}_i^L, \hat{\boldsymbol{u}}_i(\boldsymbol{x}_k)\right)\mathrm{d}\boldsymbol{x}_k. \end{aligned} \tag{32}$$

The action NN weights are tuned according to Eq. (9), which is equivalent to

$$\begin{aligned} \boldsymbol{\Theta}_i^L = \arg\min_{\boldsymbol{\Theta}^L}\big\{&\boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \hat{\boldsymbol{u}}_i^{\mathrm{T}}(\boldsymbol{x}_k, \boldsymbol{\Theta}^L)\boldsymbol{R}\hat{\boldsymbol{u}}_i(\boldsymbol{x}_k, \boldsymbol{\Theta}^L) \\ &+ \alpha\hat{V}_i(\boldsymbol{x}_{k+1}^i)\big\}, \end{aligned} \tag{33}$$

where $\boldsymbol{x}_{k+1}^i = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{g}(\boldsymbol{x}_k)\hat{\boldsymbol{u}}_i(\boldsymbol{x}_k, \boldsymbol{\Theta}^L)$. The weights are updated using the gradient descent training method

$$\begin{aligned} &\boldsymbol{\Theta}_{i|m+1}^L \\ &= \boldsymbol{\Theta}_{i|m}^L - \lambda\frac{\partial\left(\boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}_k + \hat{\boldsymbol{u}}_i^{\mathrm{T}}(\boldsymbol{x}_k, \boldsymbol{\Theta}_{i|m}^L)\boldsymbol{R}\hat{\boldsymbol{u}}_i(\boldsymbol{x}_k, \boldsymbol{\Theta}_{i|m}^L) + \alpha\hat{V}_i(\boldsymbol{x}_{k+1}^i)\right)}{\partial\boldsymbol{\Theta}_i^L} \\ &= \boldsymbol{\Theta}_{i|m}^L - \lambda\boldsymbol{\sigma}(\boldsymbol{x}_k)(2\boldsymbol{R}\hat{\boldsymbol{u}}_i^{\mathrm{T}}\left(\boldsymbol{x}_k, \boldsymbol{\Theta}_{i|m}^L\right) \\ &\quad + \alpha\boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x}_k)\frac{\partial\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{x}_{k+1})}{\partial\boldsymbol{x}_{k+1}}\boldsymbol{W}_{i+1}^L, \end{aligned} \tag{34}$$

and $\boldsymbol{\Theta}_i^L = \lim_{m\to\infty}\boldsymbol{\Theta}_{i|m}^L$, where $\lambda > 0$ is the step size and m is the iteration number.

Note that equations (32) and (34) do not need the internal system dynamics, and $\boldsymbol{g}(\boldsymbol{x}_k)$ updates the control only using Eq. (34), so the algorithm runs with partially unknown dynamics.

The optimal control for nonlinear DT affine systems can be obtained offline by the following algorithm.

1) Define an NN as $\hat{V}_i(\boldsymbol{x}) = (\boldsymbol{W}_i^L)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$ to approximate the value function $V_i(\boldsymbol{x})$, and another NN as $\hat{\boldsymbol{u}}_i(\boldsymbol{x}) = (\boldsymbol{\Theta}_i^L)^{\mathrm{T}}\boldsymbol{\sigma}(\boldsymbol{x})$ to approximate the value function $\boldsymbol{u}_i(\boldsymbol{x})$.

2) Initialize the weights $\boldsymbol{W}_i^L$ and $\boldsymbol{\Theta}_i^L$. Set $i = 0$ and the specified error ϵ.

3) Compute $\boldsymbol{\Theta}_i^L$ by Eq. (34) to derive the value function $\hat{\boldsymbol{u}}_i(\boldsymbol{x}) = (\boldsymbol{\Theta}_i^L)^{\mathrm{T}}\boldsymbol{\sigma}(\boldsymbol{x})$; then compute $\boldsymbol{W}_i^L$ using Eq. (32) to derive the value function $\hat{V}_i(\boldsymbol{x}) = (\boldsymbol{W}_i^L)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$.

4) If $|\hat{V}_{i+1}(\boldsymbol{x}) - \hat{V}_i(\boldsymbol{x})| < \epsilon$ holds, go to 5); otherwise, set $i = i+1$, and go to 3).

5) If condition (23) holds, go to step 6); otherwise, the discount factor should be settled larger.

6) Obtain the final optimal value and control law, stop.

5. Numerical simulation

The following nonlinear discrete-time system example is used to demonstrate the effectiveness of the NN implementation for the aforementioned algorithm.

Example 1 Consider the following DT affine nonlinear system:

$$\boldsymbol{f}(\boldsymbol{x}_k) = \begin{bmatrix} \sin \boldsymbol{x}_{1k} \\ \boldsymbol{x}_{1k}\boldsymbol{x}_{2k} \end{bmatrix}, \quad \boldsymbol{g}(\boldsymbol{x}_k) = \begin{bmatrix} 0.4 \\ -0.2 \end{bmatrix}.$$

The initial state is set as $\boldsymbol{x}_0 = [1 \ \ -1]^{\mathrm{T}}$, the performance index follows the form in Eq. (2) and the weights are given by

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

The basis functions of the NNs are selected as follows. The control is approximated by $\hat{\boldsymbol{u}}_i(\boldsymbol{x}_k) = (\boldsymbol{\Theta}_i^L)^{\mathrm{T}}\boldsymbol{\sigma}(\boldsymbol{x}_k)$, where $\boldsymbol{\sigma}(\boldsymbol{x}_k)$ is the basis given by

$$\begin{aligned}\boldsymbol{\sigma}(\boldsymbol{x}_k) = [&\boldsymbol{x}_k(1) \ \ \boldsymbol{x}_k(2) \ \ \boldsymbol{x}_k^3(1) \ \ \boldsymbol{x}_k^2(1)\boldsymbol{x}_k(2) \\ &\boldsymbol{x}_k(1)\boldsymbol{x}_k^2(1) \ \ \boldsymbol{x}_k^3(2) \ \ \boldsymbol{x}_k^5(1) \ \ \boldsymbol{x}_k^4(1)\boldsymbol{x}_k(2) \\ &\boldsymbol{x}_k^3(1)\boldsymbol{x}_k^2(2) \ \ \boldsymbol{x}_k^2(1)\boldsymbol{x}_k^3(2) \ \ \boldsymbol{x}_k(1)\boldsymbol{x}_k^4(2) \ \ \boldsymbol{x}_k^5(2)]^{\mathrm{T}},\end{aligned}$$

and the weights are

$$\begin{aligned}\boldsymbol{\Theta}_i^L = [&\boldsymbol{\theta}_{i1} \ \ \boldsymbol{\theta}_{i2} \ \ \boldsymbol{\theta}_{i3} \ \ \boldsymbol{\theta}_{i4} \ \ \boldsymbol{\theta}_{i5} \ \ \boldsymbol{\theta}_{i6} \\ &\boldsymbol{\theta}_{i7} \ \ \boldsymbol{\theta}_{i8} \ \ \boldsymbol{\theta}_{i9} \ \ \boldsymbol{\theta}_{i10} \ \ \boldsymbol{\theta}_{i11} \ \ \boldsymbol{\theta}_{i12}]^{\mathrm{T}}.\end{aligned}$$

The value is approximated by $\hat{V}_i(\boldsymbol{x}_k) = (\boldsymbol{W}_i^L)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_k)$, where $\boldsymbol{\phi}(\boldsymbol{x}_k)$ is the basis given by

$$\begin{aligned}\boldsymbol{\phi}(\boldsymbol{x}_k) = [&\boldsymbol{x}_k^2(1) \ \ \boldsymbol{x}_k(1)\boldsymbol{x}_k(2) \ \ \boldsymbol{x}_k^2(2) \ \ \boldsymbol{x}_k^4(1) \\ &\boldsymbol{x}_k^3(1)\boldsymbol{x}_k^1(2) \ \ \boldsymbol{x}_k^2(1)\boldsymbol{x}_k^2(2) \ \ \boldsymbol{x}_k^3(1)\boldsymbol{x}_k^1(2) \\ &\boldsymbol{x}_k^4(2)\boldsymbol{x}_k^6(1) \ \ \boldsymbol{x}_k^5(1)\boldsymbol{x}_k(2) \ \ \boldsymbol{x}_k^4(1)\boldsymbol{x}_k^2(2) \\ &\boldsymbol{x}_k^3(1)\boldsymbol{x}_k^3(2) \ \ \boldsymbol{x}_k^2(1)\boldsymbol{x}_k^4(2) \ \ \boldsymbol{x}_k(1)\boldsymbol{x}_k^5(2) \ \ \boldsymbol{x}_k^6(2)]^{\mathrm{T}},\end{aligned}$$

and the weights are

$$\begin{aligned}\boldsymbol{W}_i^L = [&\boldsymbol{\omega}_{i1} \ \ \boldsymbol{\omega}_{i2} \ \ \boldsymbol{\omega}_{i3} \ \ \boldsymbol{\omega}_{i4} \ \ \boldsymbol{\omega}_{i5} \ \ \boldsymbol{\omega}_{i6} \ \ \boldsymbol{\omega}_{i7} \ \ \boldsymbol{\omega}_{i8} \\ &\boldsymbol{\omega}_{i9} \ \ \boldsymbol{\omega}_{i10} \ \ \boldsymbol{\omega}_{i11} \ \ \boldsymbol{\omega}_{i12} \ \ \boldsymbol{\omega}_{i13} \ \ \boldsymbol{\omega}_{i14} \ \ \boldsymbol{\omega}_{i15}]^{\mathrm{T}}.\end{aligned}$$

The training error of the NN is selected as 10^{-6} and the discount factor $\alpha = 1$. The algorithm runs 301 iterative steps under the learning rate $\lambda = 0.05$ to ensure that the accuracy 10^{-6} has been reached. The convergence processes of the state trajectories and the control inputs are given in Figs. 1 and 2, respectively. The iterative value function is shown in Fig. 3. Figure 4 shows that the condition (23) is satisfied from the plot of the error between $V_{301}(\boldsymbol{x}_k) - V_{300}(\boldsymbol{x}_k)$ and $U(\boldsymbol{x}_k, \boldsymbol{u}_{300}(\boldsymbol{x}_k))$, the error is less than zero globally in the square area $[-1;1] \times [-1;1]$, thus $\boldsymbol{u}_{300}(\boldsymbol{x}_k)$ is stabilizing.

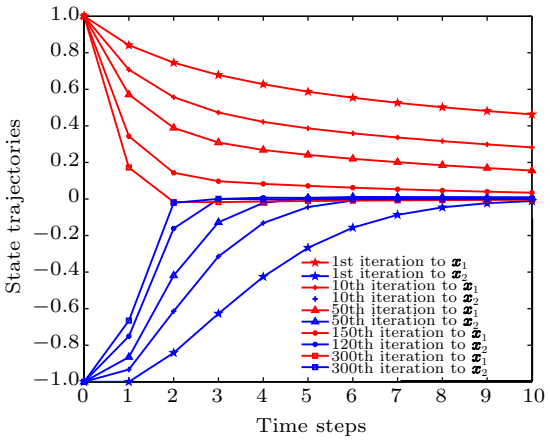


Fig. 1. (color online) The state trajectories.

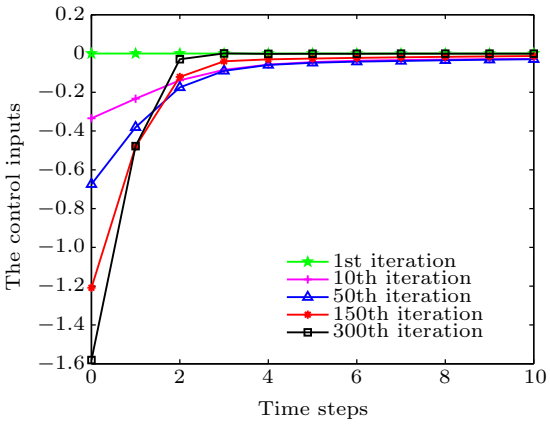


Fig. 2. (color online) The control inputs.

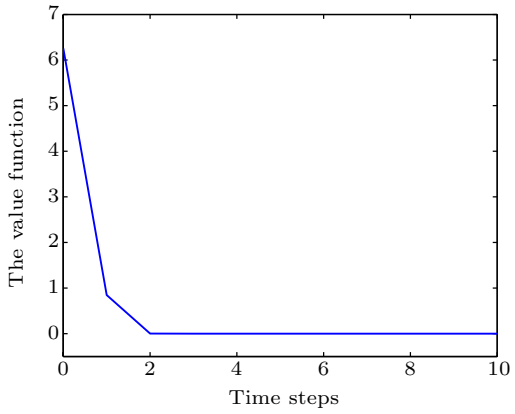


Fig. 3. (color online) The value function.

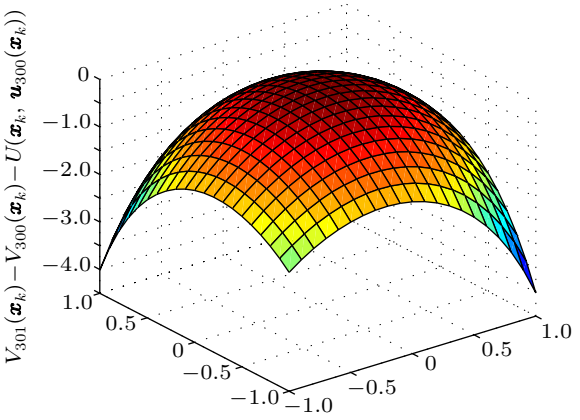


Fig. 4. (color online) The error between $V_{301}(\boldsymbol{x}_k) - V_{300}(\boldsymbol{x}_k)$ and $U(\boldsymbol{x}_k, \boldsymbol{u}_{300}(\boldsymbol{x}_k))$.

In order to show the discount factor impacts on the convergence speed of the algorithm, we set $\alpha = 0.6$. The algorithm runs 234 steps then the given accuracy 10^{-6} is reached, which means that the smaller discount factor ensures the faster convergence speed for the value function. To visually compare with the case of $\alpha = 1$, we show the convergence processes of the state trajectories and the control inputs in Figs. 5 and 6, respectively. Figure 7 shows that the condition (23) is satisfied from the plot of error between $V_{234}(\boldsymbol{x}_k) - 0.6V_{233}(\boldsymbol{x}_k)$ and $U(\boldsymbol{x}_k, \boldsymbol{u}_{233}(\boldsymbol{x}_k)$, the error is negative globally on $[-1; 1] \times [-1; 1]$, hence, the iterative control law $\boldsymbol{u}_{233}(\boldsymbol{x}_k)$ is stabilizing.

The numerical results show that there exists a finite horizon length of which the iterative control laws will guarantee the stability. By introducing a discount factor, the proposed value iteration algorithm not only can obtain the stabilizing iterative control law that ensures the closed-loop stability of the system, but also shows a faster convergence property, which greatly improves the practicability of the developed algorithm.

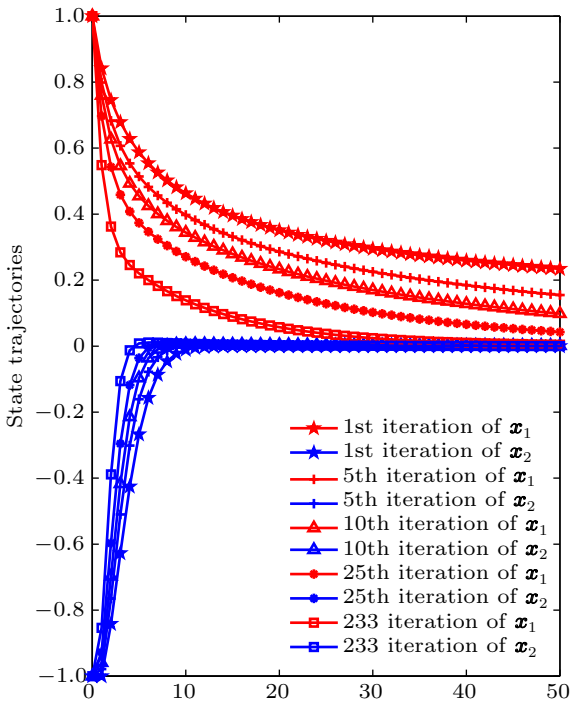


Fig. 5. (color online) The state trajectories.

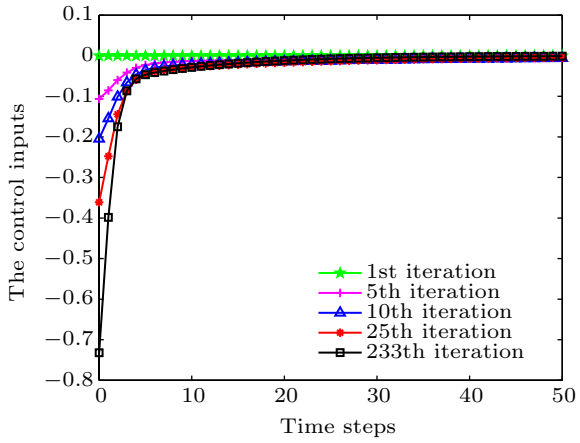


Fig. 6. (color online) The control inputs.

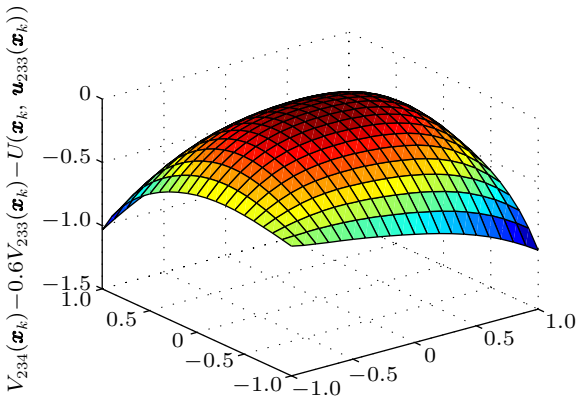


Fig. 7. (color online) The error between $V_{234}(\boldsymbol{x}_k) - 0.6V_{233}(\boldsymbol{x}_k)$ and $U(\boldsymbol{x}_k, \boldsymbol{u}_{233}(\boldsymbol{x}_k))$.

6. Conclusion

In this paper, we proved the convergence and stability of a finite horizon formulation of VI algorithm for nonlinear DT affine systems accompanied by a discounted quadric performance index. The necessary and sufficient condition for the optimal value function serving as a Lyapunov function was investigated. It was proven that for the infinite horizon problem, there exists a finite iterative such that the iterative control laws will guarantee the stability. Neural networks (NNs) were used to implement the approach without knowing the internal system dynamics. A simulation example was given to demonstrate the effectiveness of the developed optimal control approach.

Appendix A

In order to show that introducing a discount factor to the performance index may cause divergence of the considered systems using the optimal control by solving the HJB equation, a linear system quadric regulator problem is considered here for simplicity.

Consider the following linear system $\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k$, where

$$\boldsymbol{A} = \begin{bmatrix} 1 & -0.5 \\ 1.5 & 0.3 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ -0.1 \end{bmatrix}.$$

The weight matrices are chosen as

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

We can derive the optimal control by solving the Riccati equation

$$\boldsymbol{K} = \boldsymbol{A}^{\mathrm{T}}[\alpha\boldsymbol{K} - \alpha^2\boldsymbol{K}\boldsymbol{B}(\alpha\boldsymbol{B}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{B} + \boldsymbol{R})^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{K}]\boldsymbol{A} + \boldsymbol{Q},$$

the optimal feedback control is given by

$$\boldsymbol{u}^*(\boldsymbol{x}_k) = -\alpha(\alpha\boldsymbol{B}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{B} + \boldsymbol{R})^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{A}\boldsymbol{x}_k.$$

Let $\alpha = 0.9$, then we obtain the optimal control

$$\boldsymbol{u}^*(\boldsymbol{x}_k) = [0.3365 \;\; 0.2245]\boldsymbol{x}_k.$$

The closed-loop system gain matrix $\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}$ is

$$\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K} = \begin{bmatrix} 1.000 & -0.5000 \\ 1.4664 & 0.2776 \end{bmatrix},$$

the poles of the system are $0.6388 \pm 0.7763\mathrm{j}$, the spectral radius of $\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}$ is 1.0054, thus the closed-loop system diverges.

References

[1] Bellman R E 1957 *Dynamic Programming* (Princeton: Princeton University Press)

[2] Zhang H G, Liu D R, Luo Y H and Wang D 2013 *Adaptive Dynamic Programming for Control-algorithms and Stability* (London: Springer-Verlag)

[3] Werbos P J 1992 *Approximate Dynamic Programming for Real-time Control and Neural Modeling, Handbook of Intelligent Control* (New York: Van Nostrand Reinhold)

[4] Bertsekas D P and Tsitsiklis J N 1995 *Proceedings of the 34th IEEE Conference on IEEE*, 1995, pp. 560–564

[5] Prokhorov D V and Wunsch D C 1997 *IEEE Trans. Neural Netw.* **8** 997

[6] Wang F Y, Zhang H G and Liu D R 2009 *IEEE Computational Intelligence Magazine* **4** 39

[7] Doya K 2000 *Neural Computation* **12** 219

[8] Liang H J, Li H Y, Yu Z D, Li P and Wang W 2017 *IET Control Theory & Applications* **11** 1928

[9] Hanselmann T, Noakes L and Zaknich A 2007 *IEEE Trans on Neural Networks* **18** 631

[10] Wei Q L, Shi G, Song R Z and Liu Y 2017 *IEEE Trans. Indus. Elect.* **64** 5468

[11] Howard R 1960 *Dynamic Programming and Markov Processes* (Cambridge: MIT Press)

[12] Song R Z, Xiao W D, Sun C Y and Wei Q L 2015 *Chin. Phys. B* **22** 090502

[13] Wei Q L, Lewis F L, Shi G and Song R Z 2017 *IEEE Trans. Indus. Elect.* **64** 9527

[14] Leake R J and Liu R W 1967 *J. SIAM Control* **5** 54

[15] Song R Z and Wei Q L 2017 *Chin. Phys. B* **26** 030505

[16] Wei Q L, Song R Z, S Q Y and Xiao W D 2015 *Chin. Phys. B* **24** 090504

[17] Beard R, Saridis G and Wen J 1997 *Automatica* **33** 2158

[18] Wei Q L, Liu D R and Xu Y C 2015 *Chin. Phys. B* **24** 030502

[19] Song R Z, Xiao W D and Wei Q L 2014 *Chin. Phys. B* **23** 050504

[20] Wei Q L, Liu D R, Lewis F L, Liu Y and Zhang J 2017 *IEEE Trans. Indus. Elect.* **64** 4110

[21] Zhang Y, Zhang Z, Qian H and Hu G 2017 *Chin. Phys. B* **26** 100508

[22] Wei Q L, Liu D R, and Lin H Q 2016 *IEEE Trans. Cybernetics* **46** 840

[23] Hong Y Y, Qiang Z and Qi J Z 2017 *Chin. Phys. B* **26** 100506

[24] Wei Q L, Lewis F L, Sun Q Y, Yan P F and Song R Z 2017 *IEEE Trans. Cybernetics* **47** 1224

[25] Abu-Khalaf M and Lewis F L 2005 *Automatica* **41** 779

[26] Zhang H G, Wei Q L and Liu D R 2011 *Automatica* **47** 207

[27] Vrabie D and Lewis F L 2009 *Proceedings of International Joint Conference on Neural Networks*, Atlanta, pp. 3224–3231

[28] Lewis F L and Vamvoudakis K G 2011 *IEEE Trans. Syst., Man, Cybern. B* **41** 14

[29] Song R Z, Lewis F L and Wei Q L Vrabie D and Lewis F L 2017 *IEEE Trans. Neural Netw. Learn. Syst.* **28** 704

[30] Zhang H G, Luo Y H and Liu D R 2009 *IEEE Trans. Neural Netw.* **20** 1490

[31] Zhang H G, Wei Q L and Luo Y H 2008 *IEEE Trans. Syst., Man, Cybern. B* **38** 937

[32] Al-Tamimi A, Lewis F L and Abu-Khalaf M 2008 *IEEE Trans. Syst., Man, Cybern. B* **38** 943

[33] Li H L, Liu D R and Wang D 2012 *Proceedings of the* 31*st Chinese Control Conference*, July 25–27, 2012, Hefei, China, p. 2932

[34] Wei Q L and Liu D R Vrabie D and Lewis F L 2012 *WCCI* 2012 *IEEE World Congress on Computational Intelligence*, June, 10–15, 2012, Brisbane, Australia

[35] Wang D, Liu D R, Wei Q L, Zhao D B and Jin N 2012 *Automatica* **48** 1825

[36] Primbs J A and Nevistic V 2000 *Automatica* **36** 965

[37] Lewis F L and Vrabie D 2009 *Proceedings of the 7th Asian Control Conference*, Hong Kong, China, August 27–29, 2009

[38] Rantzer A *IEE Proc. Control Theory Appl.* **153** 567

[39] Lincoln B and Rantzer A Vrabie D and Lewis F L 2006 *IEEE Trans. Autom. Con.* **51** 1249