OPTIMAL CONTROL APPLICATIONS AND METHODS
Optim. Control Appl. Meth. 2010; **00**:1–14
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/oca

Optimal Control for Discrete-Time Systems with Actuator Saturation

Qiao Lin[1], Qinglai Wei[1,*], and Bo Zhao[1]

[1] *The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China*

SUMMARY

In this study, we use generalized policy iteration approximate dynamic programming (ADP) algorithm to design an optimal controller for a class of discrete-time systems with actuator saturation. A integral function is proposed to manage the saturation nonlinearity in actuators and then the generalized policy iteration ADP algorithm is developed to deal with the optimal control problem. Compared with other algorithm, the developed ADP algorithm includes two iteration procedures. In the present control scheme, two neural networks are introduced to approximate the control law and performance index function. Furthermore, numerical simulations illustrate the convergence and feasibility of the developed method. Copyright © 2010 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Approximate dynamic programming, discrete-time, generalized policy iteration, optimal control, saturating actuators

1. INTRODUCTION

In actuators, saturation nonlinearity is universal phenomenon. To deal with the control problem of systems with saturating actuators, researchers have done many works. In [1], a semi-global approach was used to slove the above problem. Other ways to solve saturation phenomena can be obtained in [2, 3]. However, these traditional ways didn't take optimal control laws into consideration. To avoid this shortcoming, Lewis et al. [4] used the framework of the Hamilton-Jacobi-Bellman (HJB) equation appearing in optimal control theory to solve the above problem. Actually, the solution of HJB equation is difficult to obtain. Thus, an effective tool, called artificial neutral networks (ANNs or NNs), is proposed. The ability of self-learning is the main advantage of NNs. So we can choose NNs to realize the function approximation in approximate dynamic programming (ADP) algorithm. The effective brain-like ADP algorithm [5, 6, 7] can solve the HJB equation forward-in-time and overcome the cure of dimensionality. With many advantages, the ADP algorithm has been developed as a powerful tool for solving optimal control problem. There are some synonyms of ADP including approximate dynamic programming [8, 9], adaptive dynamic programming [10, 11, 12], adaptive critic designs [13, 14], neural dynamic programming [15, 16], neurodynamic programming [17], and reinforcement learning [18, 19].

In recent years, Lewis [20, 21, 22], Jagannathan [23, 24], Murray [25], Powell [26] and Liu [27, 28, 29, 30] have made great contribution to the development of the ADP algorithms.

[*] Correspondence to: Qinglai Wei, The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. E-mail: qinglai.wei@ia.ac.cn

According to [5], ADP approaches were classified into several schemes including heuristic dynamic programming (HDP), action dependent HDP (ADHDP), dual heuristic programming (DHP), action dependent DHP (ADDHP), globalized DHP (GDHP), and ADGDHP. Al-Tamimi et al. [20] solved the discrete-time HJB equation of optimal control by a HDP iteration algorithm. Qiao et al. [31] used DHP to solve the wide-area coordinating control of a power system with a large wind farm and multiple FACTS devices. Mu et al. [32] studied GDHP to approximate optimal tracking control for a class of discrete-time nonlinear systems. In [33], the ADHDP method was used to coordinated multiple ramps metering. Value iteration and policy iteration are two classes of ADP algorithms to obtain the solution of the HJB. Value iteration algorithm iterates between policy improvement and cost function update. With an initial stabilizing control policy, the policy iteration algorithm iterates between policy improvement and policy evaluation. In [34], Li and Liu used value iteration to deal with optimal control for discrete-time nonlinear systems. Luo and Wu [35] proposed computationally efficient simultaneous policy update algorithm for nonlinear H∞ state feedback control with Galerkin's method.

Considering the superiority of the ADP algorithm, more and more researchers used the ADP algorithm to solve the optimal control probelm for the discrete-time nonlinear systems with actuator saturation. In [36], Zhang et al. studied the iterative DHP algorithm to deal with the constrained control problem. Song et al. [37] used HDP to overcome the saturation nonlinearity for time-delay systems. In [38], Liu et al. designed an optimal controller for unknown discrete-time nonlinear systems with control constraints by DHP. However, in [39], we got that almost all ADP and reinforcement learning algorithms could be represented by the generalized policy iteration algorithm. So in order to promote the development of ADP, it is significant to study the generalized policy iteration algorithm. Furthermore, there were a great deal of efforts to use the generalized policy iteration ADP algorithm to manage the optimal control problem. Wei and Liu [40, 41] used the generalized policy iteration algorithm to deal with the discrete-time systems. In [42, 43], the generalized policy iteration algorithm was developed to solve the continuous-time nonlinear optimal control problems. Lin et al. [44] studied the generalized policy iteration algorithm to deal with the optimal tracking control problem. Above all, we know that the generalized policy iteration algorithm has been an efficient tool in the optimal control field. However, to the best of our knowledge, there's no research on how to use the developed algorithm to solve the constrained optimal control problem.

In this paper, we focus on how to use the generalized policy iteration algorithm to obtain the optimal controller for the discrete-time nonlinear systems with actuator saturation. The present generalized policy iteration algorithm has i-iteration and j-iteration. By changing the value of i and j, the developed algorithm can be transformed into value iteration and policy iteration algorithms. When j is equal to zero, the generalized policy iteration ADP algorithm will become a value iteration algorithm [17]. On the other hand, when j approaches the infinity, the developed algorithm can be considered as a policy iteration algorithm [45]. Furthermore, the developed algorithm can accelerate the convergence rate without requiring to solve the HJB equation for i-iteration. First, a nonquadratic function is used to derive the HJB equation for discrete-time nonlinear systems with actuator saturation. Then the novel ADP algorithm is proposed to solve the HJB equation. Meanwhile, convergence criteria of the generalized policy iteration algorithm can be proved with an initial arbitrary admissible control law. By the proof process, we will get that the control law and the iterative cost function both converge monotonically to the optimum. The action network and critic network are used to compute the control law and approximate the performance index function.

The rest of the paper is organized as follows. In Section 2, the optimal control problem and the discrete-time HJB equation are recalled for discrete-time nonlinear systems. In Section 3, the generalized policy iteration algorithm is derived and the properties of the algorithm are analyzed. In Section 4, we propose the NN implementation of the developed approach. In Section 5, two numerical examples are given to show the effectiveness of the developed algorithm. In Section 6, the conclusions are given.

2. PRELIMINARIES

Let's consider the following discrete-time nonlinear systems:

$$x_{k+1} = f(x_k) + g(x_k)u_k \tag{1}$$

where $x_k \in \mathbb{R}^n$, $f(x_k) \in \mathbb{R}^n$, $g(x_k) \in \mathbb{R}^{n\times m}$ and the input $u_k \in \mathbb{R}^m$. Here assume that $f + gu$ is Lipschitz continuous on a set Ω in $\mathbb{R}^n$ containing the origin, and that the system is controllable on $\Omega \in \mathbb{R}^n$. We denote $\Omega_u = \{u_k | u_k = [u_{1k}, u_{2k}, ..., u_{mk}]^{\mathsf{T}}$
$\in \mathbb{R}^m, |u_{ik}| \le \overline{u}_i, i = 1, 2, ..., m\}$, where $\overline{u}_i$ is the saturating bound. Let $\overline{U} = diag[\overline{u}_1, \overline{u}_2, ..., \overline{u}_m]$.

Now let $\underline{u}_k = \{u_k, u_{k+1}, u_{k+2}, ...\}$ be a control sequence from k to ∞ with each $u_i \in \Omega_u$. So the goal of this paper is to find the optimal control law for the systems (1) so that the control sequence $\underline{u}_k$ minimizes the following performance index function

$$J(x_k, \underline{u}_k) = \sum_{i=k}^{\infty} \{x_i^{\mathsf{T}} Q x_i + W(u_i)\}, \tag{2}$$

where the weight matrix Q and $W(u_i) \in \mathbb{R}$ are positive definite.

In this paper, the constrained optimal control problem will be studied. Inspired by the study of [4] and [36], we can define

$$W(u_i) = 2\int_0^{u_i} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds, \tag{3}$$

$$\Lambda^{-1}(u_i) = \left[\Lambda^{-1}(u_{1i}), \Lambda^{-1}(u_{2i}), ..., \Lambda^{-1}(u_{mi})\right]^{\mathsf{T}}, \tag{4}$$

where R is positive definite, $s \in \mathbb{R}^m$, $\Lambda \in \mathbb{R}^m$, $\Lambda^{-\mathsf{T}}$ denotes $(\Lambda^{-1})^{\mathsf{T}}$, and $\Lambda(\cdot)$ is a monotonic odd function and a bounded single mapping function that satisfies $|\Lambda(\cdot)| \le 1$. The hyperbolic tangent function $\Lambda(\cdot) = \tanh(\cdot)$ is a good example that meets the aforementioned requirements. From the front, we can get that R is positive definite and $\Lambda^{-1}(\cdot)$ is a monotonic odd function, so $W(u_i)$ is positive definite.

In the sense, let $J^*(x_k) = \min\limits_{\underline{u}_k} J(x_k, \underline{u}_k)$ denote the optimal performance index function and u_k^* be the optimal control law. Moreover, according to discrete-time Bellman's optimality principle, the optimal performance index function can be written as

$$\begin{aligned} J^*(x_k) &= \min_{u_k} \sum_{i=k}^{\infty} \{x_i^{\mathsf{T}} Q x_i + W(u_i)\} \\ &= \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + J^*(x_{k+1}) \right\}. \end{aligned} \tag{5}$$

And the optimal control law can be expressed as

$$\begin{aligned} u_k^* &= \arg\min_{u_k} \sum_{i=k}^{\infty} \{x_i^{\mathsf{T}} Q x_i + W(u_i)\} \\ &= \arg\min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + J^*(x_{k+1}) \right\}. \end{aligned} \tag{6}$$

It's not difficult to find that if we can obtain the optimal performance index function $J^*(x_k)$, the optimal controller for discrete-time nonlinear systems with actuator saturation can be obtained. So in the following, a novel ADP algorithm will be used to solved equation (5).

DOI: 10.1002/oca

3. THE OPTIMAL CONTROL BASED ON GENERALIZED POLICY ITERATION ADP ALGORITHM

3.1. Derivation of the generalized policy iteration ADP algorithm

From [46], we can get that the traditional ADP algorithm, including value and policy iteration algorithm, just have one iteration procedure. However, the developed algorithm contains i-iteration and j-iteration. Additional, the convergence rate of the developed ADP algorithm can be sped up with no need for solving the HJB equation for i-iteration. Moreover, a control law, which not only stabilizes the system (1) but also make the performance index function finite, is said to be admissible [45].

For simplicity, the systems (1) can be represented as

$$x_{k+1} = F(x_k, u_k). \tag{7}$$

In the developed generalized policy iteration ADP algorithm, the control law and cost function are updated by iterations. First, the initial cost function $V_0(x_k)$ can be obtained with an initial admissible control law $v_0(x_k)$ as follows:

$$\begin{aligned} V_0(x_k) &= x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_0(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_0(x_{k+1}) \\ &= x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_0(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_0(F(x_k, v_0(x_k))). \end{aligned} \tag{8}$$

From [40], the control law $v_1(x_k)$ can be computed by:

$$v_1(x_k) = \arg\min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_0(F(x_k, u_k)) \right\}. \tag{9}$$

Then, the second iteration procedure will be introduced. We define an arbitrary non-negative integer sequence, that is $\{M_1, M_2, M_3, \ldots\}$. M_1 is the upper boundary of j_1. When j_1 increases from 0 to M_1, the iterative cost function is obtained by

$$V_{1,j_1+1}(x_k) = x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_1(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{1,j_1}(F(x_k, v_1(x_k))), \tag{10}$$

where

$$\begin{aligned} V_{1,0}(x_k) &= \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_0(x_{k+1}) \right\} \\ &= x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_1(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_0(F(x_k, v_1(x_k))). \end{aligned} \tag{11}$$

The iterative cost function can be defined as

$$V_1(x_k) = V_{1,M_1}(x_k). \tag{12}$$

Therefore, for $i = 2, 3, 4, \ldots$, the control law and cost function of the generalized policy iteration ADP algorithm can be updated as follows:

1) i-iteration

$$v_i(x_k) = \arg\min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{i-1}(F(x_k, u_k)) \right\}, \tag{13}$$

2) j-iteration

$$V_{i,j_i+1}(x_k) = x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_i(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{i,j_i}(F(x_k, v_i(x_k))), \tag{14}$$

DOI: 10.1002/oca

where $j_i = 0, 1, 2, \ldots, M_i$,

$$V_{i,0}(x_k) = \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{i-1}(x_{k+1}) \right\}$$
$$= x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_i(k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{i-1}(F(x_k, v_i(x_k))) \tag{15}$$

and the iterative cost function can be obtained by

$$V_i(x_k) = V_{i,M_i}(x_k). \tag{16}$$

In each j-iteration, the control law remains unchanged. What the step does is to solve the generalized HJB eqution:

$$V_{i,j_i}(x_k) = x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_i(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{i,j_i}(F(x_k, v_i(x_k))). \tag{17}$$

Remark 1
In fact, when j is equal to zero, the generalized policy iteration ADP algorithm can be regarded as a value iteration ADP algorithm [17]. And when j approaches the infinity, the developed algorithm becomes a policy iteration [45]. Above all, we can conclude that the developed novel ADP algorithm is a general idea that unifies almost all ADP and reinforcement learning methods.

From (8)–(16), the iterative cost function $V_{i,j_i}(x_k)$ and the iterative control law $v_i(x_k)$ are used to approximate $J^*(x_k)$ and u_k^*, respectively. Therefore, it's important to determine whether the algorithm is convergent. In the following, the convergence analysis will be studied.

3.2. *Convergence analysis of the generalized policy iteration ADP algorithm*

Theorem 1
Let the sequence $\{V_{i,j_i}(x_k)\}$ be defined as in (14). Let the control law sequence $\{v_i(x_k)\}$ be defined as in (13) with $v_0(x_k)$ satisfying (8). Let $\{M_1, M_2, M_3, \ldots\}$ be an arbitrary non-negative integer sequence. Then, we can conclude that $\{V_{i,j_i}(x_k)\}$ is a non-increasing sequence satisfying:

$$V_{i,j_i+1}(x_k) \le V_{i,j_i}(x_k) \tag{18}$$

and

$$V_{i+1,j_{i+1}}(x_k) \le V_{i,j_i}(x_k) \tag{19}$$

where $0 \le j_i \le M_i$ and $0 \le j_{i+1} \le M_{i+1}$.

Proof
In the following, we will use mathematical induction to prove $V_{i,j_i+1}(x_k) \le V_{i,j_i}(x_k)$.

First, we prove that (18) holds for $i = 1$. From (8) and (11), we have

$$V_{1,0}(x_k) = \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_0(F(x_k, u_k) \right\}$$
$$\le x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_0(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_0(F(x_k, v_0(x_k)))$$
$$= V_0(x_k). \tag{20}$$

Then, for $j_1 = 0$, using (10) and (20), we have

$$V_{1,1}(x_k) = x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_1(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{1,0}(F(x_k, v_1(x_k)))$$
$$\le x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_1(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_0(F(x_k, v_1(x_k)))$$
$$= V_{1,0}(x_k). \tag{21}$$

Assume (18) holds for $j_1 = l - 1$, where $1 < l \le M_1$ and l is positive integer. Then for $j_1 = l$, we have

$$\begin{aligned} V_{1,l+1}(x_k) &= x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_1(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{1,l}(F(x_k, v_1(x_k))) \\ &\le x(k)^{\mathsf{T}} Q x_k + 2\int_0^{v_1(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{1,l-1}(F(x_k, v_1(x_k))) \\ &= V_{1,l}(x_k). \end{aligned} \tag{22}$$

Therefore (18) holds for $i = 1$.

Second, we assume that (18) holds for $i = r$, where $1 < r \le \infty$ and r is positive integer, that is

$$V_{r,j_r+1}(x_k) \le V_{r,j_r}(x_k). \tag{23}$$

Then, for $i = r + 1$, using (15) and (16), we have

$$\begin{aligned} V_{r+1,0}(x_k) &= x(k)^{\mathsf{T}} Q x_k + 2\int_0^{v_{r+1}(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_r(F(x_k, v_{r+1}(x_k))) \\ &= \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_r(F(x_k, u_k)\right\} \\ &\le x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_r(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_r(F(x_k, v_r(x_k))) \\ &= V_{r,M_r+1}(x_k) \\ &\le V_{r,M_r}(x_k) \\ &= V_r(x_k). \end{aligned} \tag{24}$$

Next, for $j_{r+1} = 0$, using (14) and (24), we get

$$\begin{aligned} V_{r+1,1}(x_k) &= x(k)^{\mathsf{T}} Q x_k + 2\int_0^{v_{r+1}(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{r+1,0}(F(x_k, v_{r+1}(x_k))) \\ &\le x(k)^{\mathsf{T}} Q x_k + 2\int_0^{v_{r+1}(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_r(F(x_k, v_{r+1}(x_k))) \\ &= V_{r+1,0}(x_k). \end{aligned} \tag{25}$$

Assume (18) holds for $j_{r+1} = p - 1$, where $1 < p \le M_{r+1}$ and p is positive integer. Then for $j_{r+1} = p$, we get

$$\begin{aligned} V_{r+1,p+1}(x_k) &= U(x_k, v_{r+1}(x_k)) + V_{r+1,p}(F(x_k, v_{r+1}(x_k))) \\ &\le U(x_k, v_{r+1}(x_k)) + V_{r+1,p-1}(F(x_k, v_{r+1}(x_k))) \\ &= V_{r+1,p}(x_k). \end{aligned} \tag{26}$$

Therefore, (18) holds for $i = r + 1$. And (18) is proved by mathematical induction.

Next, when $0 \le j_{i+1} \le M_{i+1}$, using (16)–(18), we can obtain

$$V_{i+1}(x_k) = V_{i+1,M_{i+1}}(x_k) \le V_{i+1,j_{i+1}}(x_k) \le V_{i+1,0}(x_k) \le V_i(x_k) \le V_{i,j_i}(x_k). \tag{27}$$

It's not difficult to find that by means of (27), the inequality (19) is proved. □

The monotonicity of the developed ADP algorithm has been discussed in Theorem 1. Starting with an arbitrary initial admissible control law $v_0(x_k)$, $\{V_{i,j_i}(x_k)\}$ is proved to be a monotonically non-increasing sequence. In the following part, the convergence properties of the developed algorithm will be presented.

Lemma 1
If a sequence $\{a_n\}, n = 0, 1, \ldots,$ is convergent, then its subsequence is convergent. And sequence $\{a_n\}$ and its subsequence will have the same limit. [47]

Theorem 2
From (8)–(16), we can get the iterative control law $v_i(x_k)$ and the iterative cost function $V_{i,j_i}(x_k)$. Then when i approaches the infinity, the iterative cost function $V_{i,j_i}(x_k)$ converges to the optimal performance index function $J^*(x_k)$, i.e.,

$$\lim_{i\to\infty} V_{i,j_i}(x_k) = J^*(x_k). \tag{28}$$

Proof
First, let the iterative cost function sequence $\{V_{i,j_i}(x_k)\}$ be $\{V_0(x_k), V_{1,0}(x_k), V_{1,1}(x_k), \ldots, V_{1,M_1}(x_k), V_1(x_k), V_{2,0}(x_k), V_{2,1}(x_k), \ldots, V_{2,M_2}(x_k), \ldots\}$. Then, we choose a subsequence $\{V_i(x_k)\}$, that is $\{V_0(x_k), V_1(x_k), V_2(x_k), \ldots\}$. According to Lemma 1,

$$\lim_{i\to\infty} V_{i,j_i}(x_k) = \lim_{i\to\infty} V_i(x_k). \tag{29}$$

Thus, in order to prove the (28), we can choose to prove the following equation

$$\lim_{i\to\infty} V_i(x_k) = J^*(x_k). \tag{30}$$

Define $V_\infty(x_k) = \lim\limits_{i\to\infty} V_i(x_k)$. Then using Theorem 1 and (15), we have

$$\begin{aligned}
V_i(x_k) &\leq V_{i,0}(x_k) \\
&= x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_i(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{i-1}(F(x_k, v_i(x_k))) \\
&= \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{i-1}(F(x_k, u_k)) \right\}.
\end{aligned} \tag{31}$$

According to (31), we get

$$\begin{aligned}
V_\infty(x_k) &= \lim_{i\to\infty} V_i(x_k) \\
&\leq V_i(x_k) \\
&\leq \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{i-1}(F(x_k, u_k)) \right\}.
\end{aligned} \tag{32}$$

Let $i \to \infty$, we have

$$V_\infty(x_k) \leq \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_\infty(F(x_k, u_k)) \right\}. \tag{33}$$

On the other hand, according to Theorem 1, $\{V_i(x_k)\}$ is a monotonically non-increasing sequence, so we can find a positive integer ϕ that satisfies:

$$V_\phi(x_k) - s \leq V_\infty(x_k) \leq V_\phi(x_k), \tag{34}$$

where s is an arbitrary positive constant. Therefore, using (17) and (34), we have

$$\begin{aligned}
V_\infty(x_k) &\geq x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_\phi(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_\phi(F(x_k, v_\phi(x_k))) - s \\
&\geq x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_\phi(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_\infty(F(x_k, v_\phi(x_k))) - s \\
&= \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_\infty(F(x_k, u_k)) \right\} - s,
\end{aligned} \tag{35}$$

And considering the arbitrariness of s, we can obtain that

$$V_\infty(x_k) \geq \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_\infty(F(x_k, u_k)) \right\}. \tag{36}$$

According to (33) and (36), $V_\infty(x_k)$ can be obtained, that is

$$V_\infty(x_k) = \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_\infty(F(x_k, u_k)) \right\}. \tag{37}$$

Then, let η be an arbitrary positive constant. According to (5), we can find a sequence of admissible control laws $\underline{\pi}_k$ such that the optimal performance index function satisfies

$$J(x_k, \underline{\pi}_k) \leq J^*(x_k) + \eta, \tag{38}$$

where $\underline{\pi}_k = \{\pi_k, \pi_{k+1}, \pi_{k+2}, \ldots\}$. From the above mentioned, $\underline{\pi}_k$ is an control sequence, and the length of the sequence is ϱ, where ϱ is a positive constant. Combining (2) and Theorem 1, we have

$$\begin{aligned} V_\infty(x_k) &\leq V_\varrho(x_k) \\ &\leq \min_{u_k} \left\{ x_k^{\mathsf{T}} Q x_k + 2\int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{\varrho-1}(F(x_k, u_k)) \right\} \\ &\leq J(x_k, \underline{\pi}_k). \end{aligned} \tag{39}$$

According to (37) with (38), we can obtain

$$V_\infty(x_k) \leq J^*(x_k) + \eta. \tag{40}$$

Noting that η is chosen arbitrarily, we have

$$V_\infty(x_k) \leq J^*(x_k). \tag{41}$$

On the other hand, from the definition of $J^*(x_k)$ in (5), it's easy to find that $V_i(x_k)$ is not less than $J^*(x_k)$. So when i approaches to infinite, $V_\infty(x_k) \geq J^*(x_k)$ will be obtained. Above all, we have $J^*(x_k) \leq V_\infty(x_k) \leq J^*(x_k)$, and the equation (30) is proved. □

Combining Theorem 1 with Theorem 2, the iterative cost function $V_{i,j_i}(x_k)$, which is initialized by an arbitrary admissible control law, is a monotonically nonincreasing function and converges to the J^*. According to the definition of ${u_k}^*$ in (6), when $V_{i,j_i} \to J^*$, the v_i converges to the optimal control law u^*.

4. IMPLEMENTATION OF THE GENERALIZED POLICY ITERATION ADP ALGORITHM

In this section, we choose two NNs to implement the generalized policy iteration ADP algorithm. Figure 1 shows the whole structural diagram of the developed algorithm.

4.1. *The critic network*

The role of the critic network is to approximate the cost function $V_{i,j_i}(x_k)$. The critic network has three layers and the output is given as

$$\hat{V}_{i,j_i}(x_k) = \xi^{\mathsf{T}}_{c(i,j_i)} \sigma(\chi^{\mathsf{T}}_{c(i,j_i)} x_k), \tag{42}$$

where $\delta(\cdot)$ is a sigmoid function, $\xi_{c(i,j_i)}$ and $\chi_{c(i,j_i)}$ are the weight matrices of the critic function. The target cost function can be obtained by

$$V_{i,j_i}(x_k) = x_k^{\mathsf{T}} Q x_k + 2\int_0^{v_i(x_k)} \Lambda^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}Rds + V_{i,j_i-1}(F(x_k, v_i(x_k))). \tag{43}$$

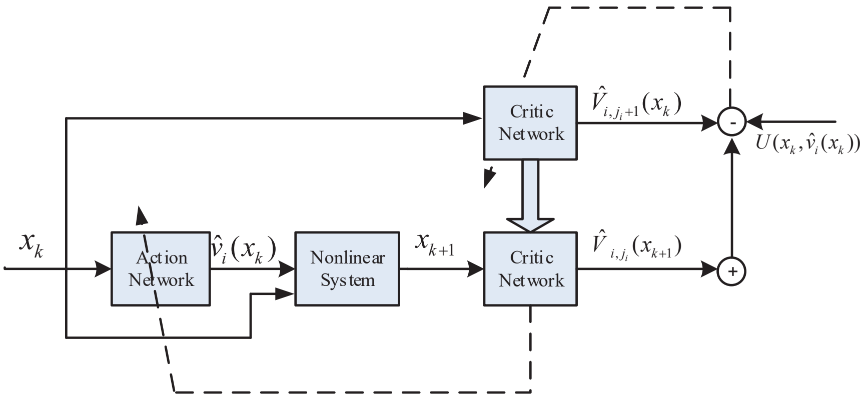


Figure 1. Structure diagram of the algorithm

The error function of the critic network can be written as

$$e_{c(i,j_i)k} = \hat{V}_{i,j_i}(x_k) - V_{i,j_i}(x_k). \tag{44}$$

Then we need to minimize the following objective function

$$E_{c(i,j_i)k} = \frac{1}{2} e^{\mathsf{T}}_{c(i,j_i)k} e_{c(i,j_i)k}. \tag{45}$$

The gradient-based weight update rule for the critic network can be given by

$$\xi_{c(i,j_i)}(\varpi + 1) = \xi_{c(i,j_i)}(\varpi) - \alpha_c \left[\frac{\partial E_{c(i,j_i)k}}{\partial \xi_{c(i,j_i)}(\varpi)} \right], \tag{46}$$

where ϖ is the iterative step and $\alpha_c > 0$ is the learning rate of the critic network. Then, in order to compute the other weight $\chi_{c(i,j_i)}$, we just need to replace ξ with χ.

4.2. *The action network*

The action network has the input layer, the hidden layer and the output layer. Moreover, the input is the state x_k and the output is

$$\hat{v}_i(k) = \xi^{\mathsf{T}}_{ai} \sigma(\chi^{\mathsf{T}}_{ai}(x_k)). \tag{47}$$

The error function of the action network is defined as

$$e_{aik} = \hat{v}_i(x_k) - v_i(x_k), \tag{48}$$

where the $v_i(x_k)$ is the target function of the action network, which is obtained by

$$v_i(x_k) = \arg\min_{u_k} \{x^{\mathsf{T}}_k Q x_k + 2 \int_0^{u_k} \Lambda^{-\mathsf{T}}(\overline{U}^{-1} s)\overline{U} R ds + V_{i-1}(F(x_k, u_k))\}. \tag{49}$$

Then, in order to obtain the weights, we need to minimize the following performance error measure:

$$E_{aik} = \frac{1}{2} e^{\mathsf{T}}_{aik} e_{aik}, \tag{50}$$

The weight is updated by gradient descent method:

$$\xi_{ai}(\gamma + 1) = \xi_{ai}(\gamma) - \beta_a \left[\frac{\partial E_{aik}}{\partial \xi_{ai}(\gamma)} \right], \tag{51}$$

where γ is the iterative step and $\beta_a > 0$ is the learning rate of the action network. The other weight χ_{ai} updating algorithm is similar to the one for ξ_{ai}.

DOI: 10.1002/oca

5. NUMERICAL EXAMPLES

In this section, the power of the generalized policy iteration ADP algorithm in discrete-time nonlinear systems with actuator saturation will be shown.

5.1. *Example 1*

Consider the following discrete-time nonlinear system [48]:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) + 0.1x_2(k) \\ -0.1x_1(k) + 1.1x_2(k) - 0.1x_2(k)x_1^2(k) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}, \tag{52}$$

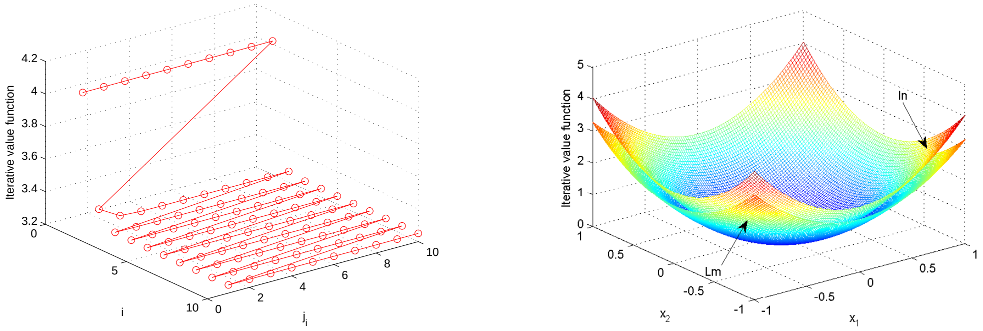
and assume that the control constraint is set to $|u_1| \leq 0.3$ and $|u_2| \leq 0.3$. The cost function is defined as

$$J(x_k) = \sum_{i=k}^{\infty} \left\{ x_i^{\mathsf{T}} Q x_i + 2\int_0^{u_i} \tanh^{-\mathsf{T}}(\overline{U}^{-1} s)\overline{U} R ds \right\}.$$

where $Q = R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\bar{U} = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix}$.

First, we perform the simulation of the generalized policy iteration ADP algorithm. The initial state is given as $x(0) = [1, -1]^{\mathsf{T}}$, and the state space is $\Omega_x = \{x_k | -1 \leq x_1(k) \leq 1, -1 \leq x_2(k) \leq 1\}$. Three-layer feedforward NNs are chosen as the critic network and action network with the structures of 2–8–1, 2–8–2, respectively. The error bound of the iteration ADP is set as $\varepsilon = 10^{-5}$. The training sets are selected from Ω_x and the weights are initialized by $[-1, 1]$. The learning rates of the critic network and action network are both 0.05 and the networks are trained for 10 iterations. For each iterations, there are 1500 training steps to guarantee the NN training error less than ε.

The iteration sequence $\{M_i\}$ is set to 10. Figure 2(a) shows the changing process of V_{i,j_i} for $k = 0$ and Figure 2(b) shows the changing curve of the iterative cost function V_i for all x_k, where ”Lm” indicates limiting iteration and ”In” means initial iteration. The monotonicity and convergence characteristics of the cost function sequence $\{V_{i,j_i}(x_k)\}$ and the subsequence $\{V_i(x_k)\}$ can be clearly obtained from Figure 2.

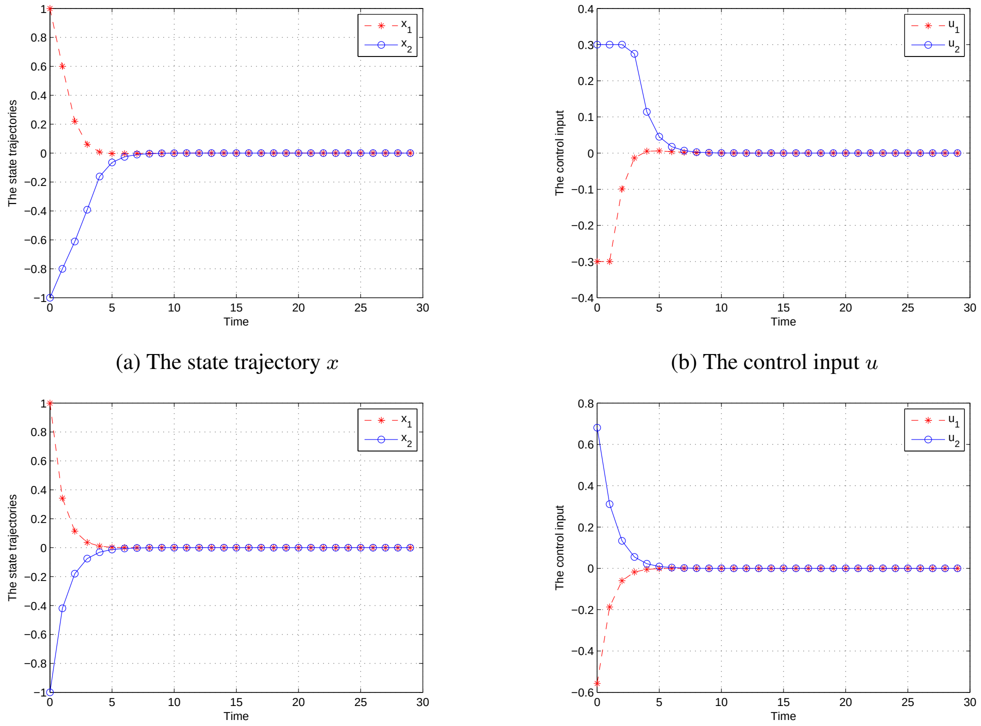


(a) V_{i,j_i}

(b) V_i for the entire state space

Figure 2. Iterative cost function

In order to contrast to the controller without considering the actuator saturation, we design two controllers for system (52) with 30 time steps. Figures 3(a) and 3(b) show the state trajectories and the control input curves for the system (52) with actuator saturation. The other case without actuator saturation is shown in Figures 3(c) and 3(d). By comparing Figures 3(b) and 3(d), we can find that the restriction of actuator saturation has been solved successfully. Moreover, we can find that if considering the actuator saturation, the time to achieve system stability increases. The simulation results verify the effectiveness of the generalized policy iteration algorithm for the discrete-time nonlinear systems with actuator saturation.



(a) The state trajectory x

(b) The control input u

(c) The state trajectory x without saturating actuator

(d) The control input u without saturating actuator

Figure 3. The simulation trajectories in Example 1

5.2. *Example 2*

The following nonlinear system is mass-spring system [36]:

$$x(k+1) = f(x(k)) + g(x(k))u(k), \tag{53}$$

where

$$x(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix},$$

$$f(x(k)) = \begin{bmatrix} x_1(k) + 0.05x_2(k) \\ -0.0005x_1(k) - 0.0335x_1^3(k) + x_2(k) \end{bmatrix},$$

$$g(x(k)) = \begin{bmatrix} 0 \\ 0.05 \end{bmatrix},$$

and the control constraint is set to $|u| \leq 0.6$. The cost function is defined as

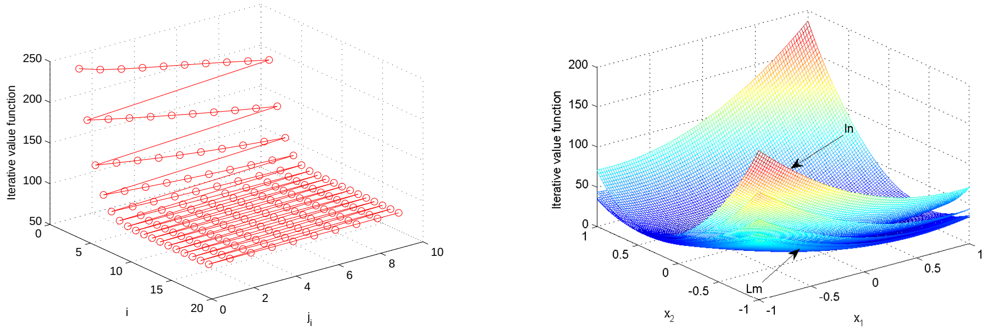
$$J(x_k) = \sum_{i=k}^{\infty} \left\{ x_i^{\mathsf{T}} Q x_i + 2\int_0^{u_i} \tanh^{-\mathsf{T}}(\overline{U}^{-1}s)\overline{U}R ds \right\},$$

where $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $R = 0.5$, $\overline{U} = 0.6$.

NNs are used to implement the developed generalized policy iteration ADP algorithm. The critic network and action network have three layers and the structures are 2–10–1, 2–10–1. We take 1000 groups of sampling data to train the network. The networks are trained for 17 iterations. In order to make the training error reach the given bound ε, the networks are trained for 4000 training steps

DOI: 10.1002/oca

with the learning rate of $\alpha_c = \beta_a = 0.01$. When $k = 0$, the convergent process of the cost function $V_{i,j_i}(x_k)$ is depicted in Figure 4(a). And the subsequence $V_i(x_k)$ for the entire state space is shown in Figure 4(b).

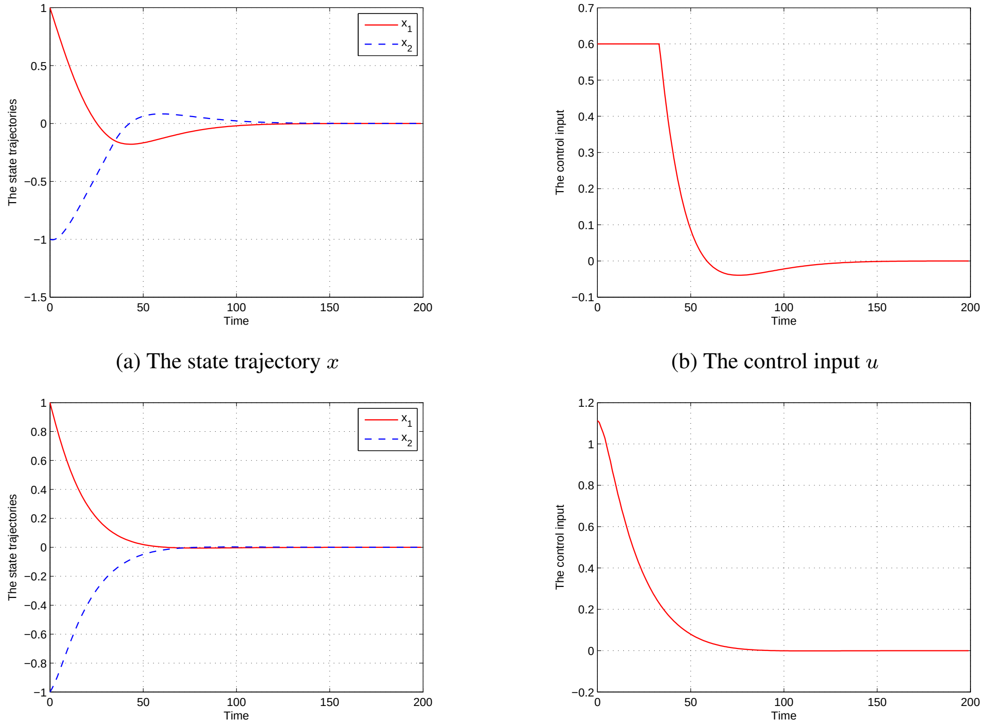


(a) V_{i,j_i}

(b) V_i for the entire state space

Figure 4. Iterative cost function

Next, we apply the optimal control laws designed by the developed ADP algorithm to system (53) with the initial state $x(0) = [1, -1]^{\mathsf{T}}$ for 200 time steps. Similarly, we discuss the optimal control problem in two different conditions. Figures 5(a) and 5(b) show the state trajectories and the control input curves for the system (53) with actuator saturation. Figures 5(c) and 5(d) show the other case without considering actuator saturation. So we can conclude that the generalized policy iteration ADP algorithm is effective in dealing with the optimal control problem for discrete-time nonlinear systems with actuator saturation.



(a) The state trajectory x

(b) The control input u

(c) The state trajectory x without saturating actuator

(d) The control input u without saturating actuator

Figure 5. The simulation trajectories in Example 2

DOI: 10.1002/oca

6. CONCLUSION

In this paper, a generalized policy iteration ADP algorithm is proposed to deal with the optimal control problem for discrete-time nonlinear systems with actuator saturation. The monotonicity and convergence characteristics of the developed algorithm are be analyzed. The critic network is given to approximate the cost function and the action network is used to compute the control law. The numerical examples demonstrate the effectiveness of the developed algorithm. Considering that the time-delay problem is another important topic of control field, so it's important to expand the developed algorithm to manage the optimal control problem for time-delay systems in the future.

7. ACKNOWLEDGEMENTS

This work was supported in part by the National Natural Science Foundation of China under Grants 61374105, 61233001, 61273140, 61503379, 61304079, 61533017, 61603387 and U1501251.

REFERENCES

1. Saberi A, Lin Z, Teel A. Control of linear systems with saturating actuators. *IEEE Transactions on Automatic Control* 1996; **41**(3):368–378.
2. Sussmann H, Sontag E, Yang Y. A general result on the stabilization of linear systems using bounded controls. *IEEE Transactions on Automatic Control* 1994; **39** (12):2411–2425.
3. Bernstein D. Optimal nonlinear, but continuous, feedback control of systems withsaturating actuators. *International Journal of Control* 1995; **62**(5):1209–1216.
4. Abu-Khalaf M, Lewis F. Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 2005; **41**(5):779–791.
5. Werbos P. Approximate dynamic programming for real-time control and neural modeling. *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches* 1992.
6. Werbos P, Miller W, Sutton R. A menu of designs for reinforcement learning over time. *Neural Networks for Control* 1991.
7. Wang D, Liu D, Zhang Q, Zhao D. Data-based adaptive critic designs for nonlinear robust optimal control with uncertain dynamics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 2016; **46**(11):1544–1555.
8. Si J, Barto A, Powell W, et al. *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.
9. Al-Tamimi A, Lewis FL, Abu-Khalaf M. Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics* 2008; **38**(4):943–949.
10. Wang D, Liu D, Wei Q, et al. Optimal control of unknown nonaffine nonlinear discretetime systems based on adaptive dynamic programming. *Automatica* 2012; **48**(8):1825–1832.
11. Wei Q, Song R, Yan P. Data-driven zero-sum neuro-optimal control for a class of continuous-time unknown nonlinear systems with disturbance using ADP. *IEEE Transactions on Neural Networks & Learning Systems* 2016; **27**(2):444–458.
12. Wang D, Liu D, Li H, et al. Neural-network-based robust optimal control design for a class of uncertain nonlinear systems via adaptive dynamic programming. *Information Sciences* 2014; **282**:167–179.
13. Liang J, Venayagamoorthy G, Harley R. Wide-area measurement based dynamic stochastic optimal power flow control for smart grids with high variability and uncertainty. *IEEE Transactions on Smart Grid* 2012; **3**(1):59–69.
14. Xu X, Hou Z, Lian C, et al. Online learning control using adaptive critic designs with sparse kernel machines. *IEEE Transactions on Neural Networks & Learning Systems* 2013; **24**(5):762–775.
15. Enns R, Si J. Helicopter trimming and tracking control using direct neural dynamic programming. *IEEE Transactions on Neural Networks* 2003; **14**(4):929–939.
16. Zhang H, Wang Z, Liu D. A Comprehensive Review of Stability Analysis of Continuous-Time Recurrent Neural Networks. *IEEE Transactions on Neural Networks & Learning Systems* 2014; **5**(7):1229–1262.
17. Bertsekas D, Tsitsiklis J. *Neuro-Dynamic Programming*, Athena Scientific, 1996.
18. Bertsekas D. Temporal difference methods for general projected equations. *IEEE Transactions on Automatic Control* 2011; **56**(9):2128–2139.
19. Wong W C, Lee J H. A reinforcement learning-based scheme for direct adaptive optimal control of linear stochastic systems. *Optimal Control Applications and Methods* 2010; **31**(4): 365–374.
20. Al-Tamimi A, Lewis F, Abu-Khalaf M. Model-free Q-learning designs for linear discrete-time zero-sum games with application to H_∞ control'. *Automatica* 2007; **43**(3): 473–481.
21. Lewis F. *Applied Optimal Control and Estimation*, Prentice-Hall, 1992.
22. Lewis F, Vrabie D. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine* 2009; **9**(3):40–58.
23. Zheng C, Jagannathan S. Generalized Hamilton-Jacobi-Bellman formulation-based neural network control of affine nonlinear discrete-time systems. *IEEE Transactions on Neural Networks* 2008; **19**(1):90–106.
24. He P, Jagannathan S. Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints. *IEEE Transactions on Systems Man & Cybernetics Part B* 2007; **37**(2):425–436.

25. Murray J, Cox C, Saeks R. Adaptive dynamic programming. *IEEE Transactions on Systems Man & Cybernetics Part C* 2002; **32**(2):140–153.
26. Powell W. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, NewYork, 2009.
27. Liu D, Wei Q. Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems. *IEEE Transactions on Systems Man & Cybernetics Part B* 2013; **43**(2):779–789.
28. Liu D, Wang D, Wang F, et al. Neural-network-based online HJB solution for optimal robust guaranteed cost control of continuous-time uncertain nonlinear systems. *IEEE Transactions on Cybernetics* 2014; **44**(12):2834–2847.
29. Wang D, Mu C, He H, Liu D. Event-driven adaptive robust control of nonlinear systems with uncertainties through NDP strategy. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 2016; **PP**(99):1–13.
30. Wang D, Li C, Liu D, Mu C. Data-based robust optimal control of continuous-time affine nonlinear systems with matched uncertainties. *Information Sciences* 2016; **366**:121–133.
31. Qiao W, Venayagamoorthy G, Harley R. DHP-based wide-area coordinating control of a power system with a large wind farm and multiple FACTS devices. *International Joint Conference on Neural Networks* 2007; 2093–2098.
32. Mu C, Sun C, Song A, Yu H. Iterative GDHP-based approximate optimal tracking control for a class of discrete-time nonlinear systems. *Neurocomputing* 2016; **214**:775–784.
33. Bai X, Zhao D, Yi J. The application of ADHDP method to coordinated multiple ramps metering. *International Journal of Innovative Computing, Information and Control* 2009; **5**(10B):34713481.
34. Li H, Liu D. Optimal control for discrete-time affine non-linear systems using general value iteration. *IET Control Theory & Applications* 2012; **6**(18):2725-2736.
35. Luo B, Wu H. Computationally efficient simultaneous policy update algorithm for nonlinear H∞ state feedback control with Galerkins method. *International Journal of Robust and Nonlinear Control* 2013; **23**(7):991–1012.
36. Zhang H, Luo Y, Liu D. Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints. *IEEE Transactions on Neural Networks* 2009; **20**(9):1490–1503.
37. Song R, Zhang H, Luo Y, et al. Optimal control laws for time-delay systems with saturating actuators based on heuristic dynamic programming. *Neurocomputing* 2010; **73**(16-18): 3020–3027.
38. Liu D, Wang D, Yang X. An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs. Information Sciences 2013; **220**(1):331–342.
39. Sutton R, Barto A. *Reinforcement Learning: An Introduction*, MIT Press, 1998.
40. Wei Q, Liu D, Yang X. Infinite horizon self-learning optimal control of nonaffine discrete-time nonlinear systems. *IEEE Transactions on Neural Networks & Learning Systems* 2015; **26**(4):866–879.
41. Liu D, Wei Q, Yan P. Generalized Policy Iteration Adaptive Dynamic Programming for Discrete-Time Nonlinear Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 2015; **45**(12):1577–1591.
42. Vrabie D, Lewis F. Generalized policy iteration for continuous time systems. *International Joint Conference on Neural Networks* 2009:3224–3231.
43. Vrabie D, Vamvoudakis K, Lewis F. Adaptive optimal controllers based on generalized policy iteration in a continuous-time framework. *in 17th Mediterranean Conference on Control & Automation, Thessaloniki, Greece* 2009:1402–1409;
44. Lin Q, Wei Q, Liu D. A novel optimal tracking control scheme for a class of discrete-time nonlinear systems using generalized policy iteration adaptive dynamic programming algorithm. *International Journal of Systems Science*, DOI: 10.1080/00207721.2016.1188177
45. Liu D, Wei Q. Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Transactions on Neural Networks & Learning Systems* 2014; **25**(3):621–634.
46. Wang F, Zhang H, Liu D. Adaptive Dynamic Programming: An Introduction. *IEEE Computational Intelligence Magazine* 2009; **4**(2):39–47.
47. Apostol T. *Mathematical Analysis*, Addison-Wesley Press.
48. Heydari A, Balakrishnan S. Fixed-final-time optimal tracking control of input-affine nonlinear systems. *Neurocomputing* 2014; **129**(4):528–539.