Available online at www.sciencedirect.com
ScienceDirect
Progress in Natural Science 18 (2008) 1023–1029
Progress in Natural Science
www.elsevier.com/locate/pnsc

Approximate optimal control for a class of nonlinear discrete-time systems with saturating actuators

Yanhong Luo, Huaguang Zhang *

School of Information Science and Engineering, Northeastern University, Shenyang 110004, China

Received 22 December 2007; received in revised form 23 February 2008; accepted 12 March 2008

Abstract

In this paper, we solve the approximate optimal control problem for a class of nonlinear discrete-time systems with saturating actuators via greedy iterative Heuristic Dynamic Programming (GI-HDP) algorithm. In order to deal with the saturating problem of actuators, a novel nonquadratic functional is developed. Based on the nonquadratic functional, the GI-HDP algorithm is introduced to obtain the optimal saturated controller with a rigorous convergence analysis. For facilitating the implementation of the iterative algorithm, three neural networks are used to approximate the value function, compute the optimal control policy and model the unknown plant, respectively. An example is given to demonstrate the validity of the proposed optimal control scheme.

Keywords: Saturating; GI-HDP algorithm; Nonquadratic functional; Convergence analysis; Neural networks

1. Introduction

Saturation, dead-zone, backlash, and hysteresis are the most common actuator nonlinearities in practical control system applications. Saturation nonlinearity is unavoidable in most actuators. Several methods for deriving control laws considering the saturation phenomena were found in Refs. [1–3]. However, most of these methods did not consider optimal control laws for general nonlinear discrete-time systems.

In recent years, in order to obtain the approximate optimal control law, the approximate dynamic programming (ADP) algorithm has been paid much attention by many researchers [4–15]. ADP combines adaptive critic design, reinforcement learning technique with dynamic programming. ADP approaches were classified into four main schemes: Heuristic Dynamic Programming (HDP), Dual Heuristic Dynamic Programming (DHP), Action Dependent Heuristic Dynamic Programming (ADHDP), also known as Q-learning [5], and Action Dependent Dual Heuristic Dynamic Programming (ADDHP). In Ref. [16], two new ADP schemes known as Globalized-DHP (GDHP) and ADGDHP were developed. In Ref. [14], a greedy HDP iteration scheme was proposed to solve the optimal control problem for nonlinear discrete-time systems with known mathematical model.

Though the ADP algorithm has made great progress in the optimal control field, there are still some problems unsolved, such as if the actuator has saturating characteristic and how to find a constrained optimal control by ADP algorithm. In Refs. [17,18], the nonquadratic functional was used to confront the input constraint. Via the nonquadratic functional, the special HJB equation was formulated and its solution resulted in a smooth saturated controller. However it remains difficult to actually solve for the value function of the HJB equation. Therefore, in this study the constrained optimal control problem is solved by the framework of the Hamilton–Jacobi–Bellman

* Corresponding author. Tel.: +86 024 83687762; fax: +86 024 83671498.
E-mail address: hg_zhang@21cn.com (H. Zhang).

1002-0071/$ - see front matter © 2008 National Natural Science Foundation of China and Chinese Academy of Sciences. Published by Elsevier Limited and Science in China Press. All rights reserved.
doi:10.1016/j.pnsc.2008.03.006

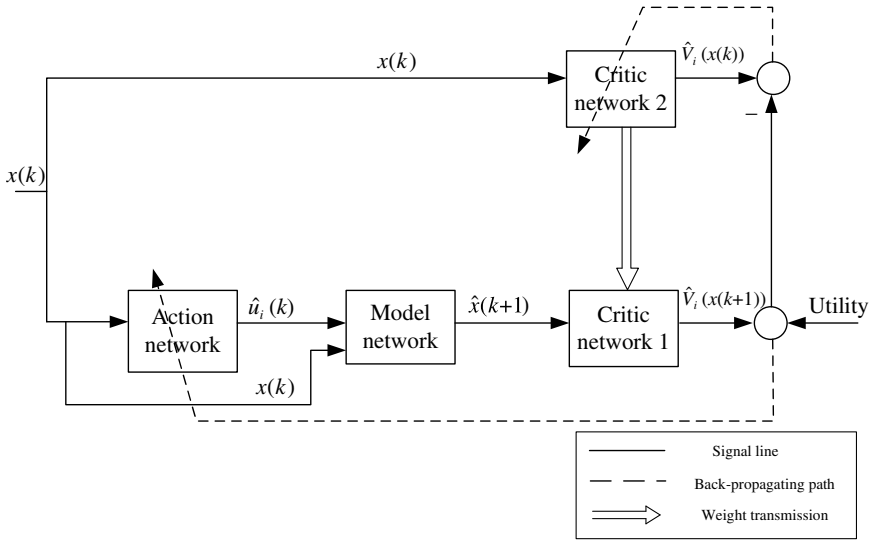

Fig. 1. The structure diagram of the GI-HDP algorithm.

(HJB) equation. After the special HJB equation is derived from the nonquadratic functional, a new ADP algorithm named greedy iterative HDP (GI-HDP) algorithm is proposed with rigorous convergence proof. Furthermore, in order to facilitate the implementation of the GI-HDP algorithm, two neural networks are used to approximate the value function and the corresponding optimal control policy, and a model network is introduced to approximate the unknown nonlinear plant.

2. Discrete-time HJB equation for constrained nonlinear systems

Consider a class of discrete-time nonlinear systems as follows:

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

where $\boldsymbol{x}(k) \in \mathbb{R}^n$ is the state vector, $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ are differentiable in their argument with $\boldsymbol{f}(0) = 0, \boldsymbol{g}(0) = 0$. Assume that $\boldsymbol{f} + \boldsymbol{g}\boldsymbol{u}$ is Lipschitz continuous on a set Ω in $\mathbb{R}^n$ containing the origin, and that the system (1) is controllable in the sense that there exists a continuous control on Ω that asymptotically stabilizes the system. And the control $\boldsymbol{u}(k) \in \Omega_u, \Omega_u = \{\boldsymbol{u}(k) = [u_1(k), u_2(k), \ldots, u_m(k)]^{\mathrm{T}} \in \mathbb{R}^m :| u_i(k) | \leqslant \bar{u}_i, i = 1, \ldots, m\}$, here $\bar{u}_i$ denotes the saturating bound for the ith actuator. Let $\overline{\boldsymbol{U}} \in \mathbb{R}^{m\times m}$ be the constant diagonal matrix described as $\overline{\boldsymbol{U}} = \mathrm{diag}\{\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_m\}$.

In this paper, we mainly discuss how to design an optimal state feedback controller for this class of constrained discrete-time systems. Therefore, it is desired to find the control policy $\boldsymbol{u}(\boldsymbol{x})$ which minimizes a generalized performance functional as follows:

$$J(\boldsymbol{x}(k), \boldsymbol{u}) = \sum_{i=k}^{\infty} \left\{ \boldsymbol{x}(i)^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x}(i) + W(\boldsymbol{u}(i)) \right\} \tag{2}$$

where both $W(\boldsymbol{u}(i))$ and $\boldsymbol{Q}$ are positive definite.

For optimal control problem, the state feedback control $\boldsymbol{u}(\boldsymbol{x})$ must not only stabilize the system on Ω but also guarantee that (2) is finite, i.e., admissible control, see Ref. [19]. From now on, we let $V^*(\boldsymbol{x}(k))$ denote the minimum value of the performance functional $J(\boldsymbol{x}(k), \boldsymbol{u})$, which is called value function or the optimal cost function in the later parts.

Definition 1. A control $\boldsymbol{u}(\boldsymbol{x})$ is defined to be admissible control with respect to (2) on Ω if $\boldsymbol{u}$ stabilizes (1) on $\Omega, \boldsymbol{u}(0) = 0$, and for all $\boldsymbol{x}(0) \in \Omega, J(\boldsymbol{x}(0), \boldsymbol{u})$ is finite.

According to Bellman optimality principle, we can obtain

$$\begin{aligned} V^*(\boldsymbol{x}(k)) &= \min_{\boldsymbol{u}(i)} \sum_{i=k}^{\infty} \left\{ \boldsymbol{x}(i)^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x}(i) + W(\boldsymbol{u}(i)) \right\} \\ &= \min_{\boldsymbol{u}(k)} \left\{ \boldsymbol{x}(k)^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x}(k) + W(\boldsymbol{u}(k)) + V^*(\boldsymbol{x}(k+1)) \right\} \end{aligned} \tag{3}$$

For unconstrained control problem, a common choice for $W(\boldsymbol{u}(i))$ is $W(\boldsymbol{u}(i)) = \boldsymbol{u}(i)^{\mathrm{T}} \boldsymbol{R} \boldsymbol{u}(i)$, where $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ is positive definite. With the first order necessity condition, we compute the gradient of right-hand side of (3) with respect to $\boldsymbol{u}$ as

$$\begin{aligned} \frac{\partial V^*(\boldsymbol{x}(k))}{\partial \boldsymbol{u}(k)} &= \frac{\partial \left(\boldsymbol{x}(k)^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x}(k) + \boldsymbol{u}(k)^{\mathrm{T}} \boldsymbol{R} \boldsymbol{u}(k) \right)}{\partial \boldsymbol{u}(k)} \\ &\quad + \left(\frac{\partial \boldsymbol{x}(k+1)}{\partial \boldsymbol{u}(k)} \right)^{\mathrm{T}} \frac{\partial V^*(\boldsymbol{x}(k+1))}{\partial \boldsymbol{x}(k+1)} = 0 \end{aligned} \tag{4}$$

Therefore we can obtain

$$\boldsymbol{u}^*(k) = -\frac{1}{2} \boldsymbol{R}^{-1} \boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x}(k)) \frac{\partial V^*(\boldsymbol{x}(k+1))}{\partial \boldsymbol{x}(k+1)} \tag{5}$$

where V^* is the value function corresponding to the optimal control policy $\boldsymbol{u}^*$.

However, for constrained control problem, the above derivation is unfeasible. To confront this bounded control problem, we introduce a nonquadratic functional motivated by Refs. [12,18].

$$W(\boldsymbol{u}(i)) = 2\int_0^{\boldsymbol{u}(i)} \boldsymbol{\varphi}^{-\mathrm{T}}(\overline{\boldsymbol{U}}^{-1}\boldsymbol{s})\overline{\boldsymbol{U}}\boldsymbol{R}\mathrm{d}\boldsymbol{s}$$
$$\boldsymbol{\varphi}^{-1}(\boldsymbol{u}(i)) = \left[\phi^{-1}(\boldsymbol{u}_1(i)), \phi^{-1}(\boldsymbol{u}_2(i)), \ldots, \phi^{-1}(\boldsymbol{u}_m(i))\right]^{\mathrm{T}} \tag{6}$$

where $\boldsymbol{R}$ is positive definite and assumed to be diagonal for simplicity of analysis, $\boldsymbol{s} \in \mathbb{R}^m, \boldsymbol{\varphi} \in \mathbb{R}^m, \phi(\cdot)$ is a bounded one-to-one function satisfying $|\phi(\cdot)| \leqslant 1$ and belonging to $C^p(p \geqslant 1)$ and $L_2(\Omega)$. Moreover, it is a monotonic increasing odd function with its first derivative bounded by a constant M. Such function is easy to find, one example is the hyperbolic tangent function $\phi(\cdot) = \tanh(\cdot)$. It should be noticed that by the definition above, $W(\boldsymbol{u}(i))$ is assured to be positive definite because $\boldsymbol{\varphi}^{-1}(\boldsymbol{u}(i))$ is monotonic odd function and R is positive definite.

Substituting (6) into (3), we can obtain

$$\begin{aligned} V^*(\boldsymbol{x}(k)) &= \min_{\boldsymbol{u}(i)} \sum_{i=k}^{\infty} \left\{ \boldsymbol{x}(i)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(i) + 2\int_0^{\boldsymbol{u}(i)} \boldsymbol{\varphi}^{-\mathrm{T}}(\overline{\boldsymbol{U}}^{-1}\boldsymbol{s})\overline{\boldsymbol{U}}\boldsymbol{R}\mathrm{d}\boldsymbol{s} \right\} \\ &= \min_{\boldsymbol{u}(k)} \left\{ \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + 2\int_0^{\boldsymbol{u}(k)} \boldsymbol{\varphi}^{-\mathrm{T}}(\overline{\boldsymbol{U}}^{-1}\boldsymbol{s})\overline{\boldsymbol{U}}\boldsymbol{R}\mathrm{d}\boldsymbol{s} \right. \\ &\quad \left. + V^*(\boldsymbol{x}(k+1)) \right\} \end{aligned} \tag{7}$$

According to the first order necessary condition of the optimal control, the following equation holds:

$$\begin{aligned} \frac{\partial V^*(\boldsymbol{x}(k))}{\partial \boldsymbol{u}(k)} &= 2\overline{\boldsymbol{U}}\boldsymbol{R}\boldsymbol{\varphi}^{-1}(\overline{\boldsymbol{U}}^{-1}\boldsymbol{u}(k)) + \left(\frac{\partial \boldsymbol{x}(k+1)}{\partial \boldsymbol{u}(k)}\right)^{\mathrm{T}} \\ &\quad \times \frac{\partial V^*(\boldsymbol{x}(k+1))}{\partial \boldsymbol{x}(k+1)} = 0 \end{aligned} \tag{8}$$

Therefore, the following formulation can be obtained:

$$\boldsymbol{u}^*(k) = \overline{\boldsymbol{U}}\boldsymbol{\varphi}\left(-\frac{1}{2}(\overline{\boldsymbol{U}}\boldsymbol{R})^{-1}\boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x}(k))V_{\boldsymbol{x}}^*(\boldsymbol{x}(k+1))\right) \tag{9}$$

Substituting (9) into (7), we obtain the special discrete-time HJB equation as follows:

$$\begin{aligned} V^*(\boldsymbol{x}(k)) &= \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) \\ &\quad + 2\int_0^{\overline{\boldsymbol{U}}\boldsymbol{\varphi}\left(-\frac{1}{2}(\overline{\boldsymbol{U}}\boldsymbol{R})^{-1}\boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x}(k))V_{\boldsymbol{x}}^*(\boldsymbol{x}(k+1))\right)} \boldsymbol{\varphi}^{-\mathrm{T}}(\overline{\boldsymbol{U}}^{-1}\boldsymbol{s})\overline{\boldsymbol{U}}\boldsymbol{R}\mathrm{d}\boldsymbol{s} \\ &\quad + V^*\left(\boldsymbol{f}(\boldsymbol{x}(k)) + \boldsymbol{g}(\boldsymbol{x}(k))\overline{\boldsymbol{U}} \right. \\ &\quad \left. \times \boldsymbol{\varphi}\left(-\frac{1}{2}(\overline{\boldsymbol{U}}\boldsymbol{R})^{-1}\boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x}(k))V_{\boldsymbol{x}}^*(\boldsymbol{x}(k+1))\right)\right) \end{aligned} \tag{10}$$

The optimal control $\boldsymbol{u}^*(k)$ can be computed if the value function $V^*(\boldsymbol{x}(k+1))$ can be solved from HJB equation (10). However, there is currently no method for rigorously solving the value function of this constrained optimal control problem. Therefore, in the next section we will discuss how to use the approximate dynamic programming algorithm named GI-HDP algorithm to solve the near-optimal control solution.

3. The derivation and implementation of near-optimal control scheme based on GI-HDP algorithm

3.1. Derivation of the GI-HDP algorithm and the corresponding convergence analysis

In the GI-HDP algorithm, for dealing with the control constraint, we choose $W(\boldsymbol{u}(k)) = 2\int_0^{\boldsymbol{u}(k)} \boldsymbol{\varphi}^{-\mathrm{T}}(\overline{\boldsymbol{U}}^{-1}\boldsymbol{s})\overline{\boldsymbol{U}}\boldsymbol{R}\mathrm{d}\boldsymbol{s}$. First, we start with initial cost function $V_0(\boldsymbol{x}(k)) = 0$ which is not necessarily the value function, and then find the control $\boldsymbol{u}_0(\boldsymbol{x}(k))$ as follows:

$$\boldsymbol{u}_0(\boldsymbol{x}(k)) = \arg\min_{\boldsymbol{u}(k)} \left(\boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}(k)) + V_0(\boldsymbol{x}(k+1))\right) \tag{11}$$

then update the cost function as

$$\begin{aligned} V_1(\boldsymbol{x}(k)) &= \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}_0(k)) + V_0(\boldsymbol{f}(\boldsymbol{x}(k)) \\ &\quad + \boldsymbol{g}(\boldsymbol{x}(k))\boldsymbol{u}_0(k)) \end{aligned} \tag{12}$$

The GI-HDP algorithm therefore iterates between

$$\begin{aligned} \boldsymbol{u}_i(\boldsymbol{x}(k)) &= \arg\min_{\boldsymbol{u}(k)} \left(\boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}(k)) \right. \\ &\quad \left. + V_i(\boldsymbol{f}(\boldsymbol{x}(k)) + \boldsymbol{g}(\boldsymbol{x}(k))\boldsymbol{u}(k))\right) \\ &= \overline{\boldsymbol{U}}\boldsymbol{\varphi}\left(-\frac{1}{2}(\overline{\boldsymbol{U}}\boldsymbol{R})^{-1}\boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x}(k))\frac{\partial V_i(\boldsymbol{x}(k+1))}{\partial \boldsymbol{x}(k+1)}\right) \end{aligned} \tag{13}$$

and

$$\begin{aligned} V_{i+1}(\boldsymbol{x}(k)) &= \min_{\boldsymbol{u}(k)} \left(\boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}(k)) \right. \\ &\quad \left. + V_i(\boldsymbol{f}(\boldsymbol{x}(k)) + \boldsymbol{g}(\boldsymbol{x}(k))\boldsymbol{u}(k))\right) \\ &= \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}_i(k)) \\ &\quad + V_i(\boldsymbol{f}(\boldsymbol{x}(k)) + \boldsymbol{g}(\boldsymbol{x}(k))\boldsymbol{u}_i(k)) \end{aligned} \tag{14}$$

In this way, the cost function and control policy are updated by recurrent iteration until they converge to the optimal ones, with the iteration number i increasing from 0 to ∞. In the following part, we shall present a proof of convergence of the iteration between (13) and (14) with the cost function $V_i \Rightarrow V^*$ and the control policy $\boldsymbol{u}_i \Rightarrow \boldsymbol{u}^*$ as $i \Rightarrow \infty$.

Lemma 1. *Let $\boldsymbol{\mu}_i$ be any arbitrary sequence of control policies and $\boldsymbol{u}_i$ be the policies as* (13). *Let V_i be as* (14) *and Λ_i as*

$$\begin{aligned} \Lambda_{i+1}(\boldsymbol{x}(k)) &= \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{\mu}_i(k)) + \Lambda_i(\boldsymbol{f}(\boldsymbol{x}(k)) \\ &\quad + \boldsymbol{g}(\boldsymbol{x}(k))\boldsymbol{\mu}_i(k)) \end{aligned} \tag{15}$$

If $V_0 = \Lambda_0 = 0$, then $V_i \leqslant \Lambda_i, \forall i$.

Proof. It is quite clear from the fact that V_{i+1} is a result of minimizing the right-hand side of (14) with respect to the control input $\boldsymbol{u}$, while Λ_{i+1} is a result of any arbitrary control input. □

Lemma 2. *Let the sequence* $\{V_i\}$ *be defined as* (14). *If the system is controllable, there is an upper bound* Y *such that* $0 \leqslant V_i \leqslant Y, \quad \forall i$.

Proof. Let $\boldsymbol{\eta}(\boldsymbol{x}(k))$ be any stabilizing and admissible control input, and let $V_0(\cdot) = Z_0(\cdot) = 0$, where V_i is updated as (14) and Z_i is updated by

$$Z_{i+1}(\boldsymbol{x}(k)) = \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{\eta}(k)) + Z_i(\boldsymbol{x}(k+1)) \tag{16}$$

It follows that the difference

$$\begin{aligned} Z_{i+1}(\boldsymbol{x}(k)) - Z_i(\boldsymbol{x}(k)) &= Z_i(\boldsymbol{x}(k+1)) - Z_{i-1}(\boldsymbol{x}(k+1)) \\ &= Z_{i-1}(\boldsymbol{x}(k+2)) - Z_{i-2}(\boldsymbol{x}(k+2)) \\ &= Z_{i-2}(\boldsymbol{x}(k+3)) - Z_{i-3}(\boldsymbol{x}(k+3)) \dots \\ &= Z_1(\boldsymbol{x}(k+i)) - Z_0(\boldsymbol{x}(k+i)) \end{aligned} \tag{17}$$

Then the following relation can be obtained

$$Z_{i+1}(\boldsymbol{x}(k)) - Z_i(\boldsymbol{x}(k)) = Z_1(\boldsymbol{x}(k+i)) - Z_0(\boldsymbol{x}(k+i)) \tag{18}$$

Since $Z_0(\cdot) = 0$, so we have

$$\begin{aligned} Z_{i+1}(\boldsymbol{x}(k)) &= Z_1(\boldsymbol{x}(k+i)) + Z_i(\boldsymbol{x}(k)) \\ &= Z_1(\boldsymbol{x}(k+i)) + Z_1(\boldsymbol{x}(k+i-1)) + Z_{i-1}(\boldsymbol{x}(k)) \\ &= Z_1(\boldsymbol{x}(k+i)) + Z_1(\boldsymbol{x}(k+i-1)) \\ &\quad + Z_1(\boldsymbol{x}(k+i-2)) + Z_{i-2}(\boldsymbol{x}(k)) \\ &= Z_1(\boldsymbol{x}(k+i)) + Z_1(\boldsymbol{x}(k+i-1)) \\ &\quad + Z_1(\boldsymbol{x}(k+i-2)) + \cdots + Z_1(\boldsymbol{x}(k)) \end{aligned} \tag{19}$$

So (19) can be written as

$$\begin{aligned} Z_{i+1}(\boldsymbol{x}(k)) &= \sum_{j=0}^{i} Z_1(\boldsymbol{x}(k+j)) \\ &= \sum_{j=0}^{i} \left(\boldsymbol{x}(k+j)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k+j) + W(\boldsymbol{\eta}(\boldsymbol{x}(k+j)))\right) \\ &\leqslant \sum_{j=0}^{\infty} \left(\boldsymbol{x}(k+j)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k+j) + W(\boldsymbol{\eta}(\boldsymbol{x}(k+j)))\right) \end{aligned} \tag{20}$$

Note that $\boldsymbol{\eta}(\boldsymbol{x}(k))$ is a stabilizing and admissible control input, i.e., $\boldsymbol{x}(k) \to 0$ as $k \to \infty$, therefore we have

$$\forall i: \quad Z_{i+1}(\boldsymbol{x}(k)) \leqslant \sum_{i=0}^{\infty} Z_1(\boldsymbol{x}(k+i)) \leqslant Y \tag{21}$$

Combining with Lemma 1, we can obtain

$$\forall i: \quad V_{i+1}(\boldsymbol{x}(k)) \leqslant Z_{i+1}(\boldsymbol{x}(k)) \leqslant Y \tag{22}$$

This completes the proof. □

With Lemma 1 and Lemma 2, the next main theorem can be derived.

Theorem 1. *Define the sequence* $\{V_i\}$ *as* (14), *with* $V_0(\cdot) = 0$. *Then* $\{V_i\}$ *is a nondecreasing sequence satisfying* $V_{i+1}(\boldsymbol{x}(k)) \geqslant V_i(\boldsymbol{x}(k)), \forall i$ *and converging to the value function of the discrete-time HJB* equation (10), *i.e.*, $V_i \Rightarrow V^*$ *as* $i \Rightarrow \infty$. *Meanwhile, the control policy also converges to the optimal control policy* (9), *i.e.*, $\boldsymbol{u}_i \Rightarrow \boldsymbol{u}^*$ *as* $i \Rightarrow \infty$.

Proof. For the convenience of analysis, define a new sequence Φ_i as follows:

$$\Phi_{i+1}(\boldsymbol{x}(k)) = \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}_{i+1}(k)) + \Phi_i(\boldsymbol{x}(k+1)) \tag{23}$$

with $\Phi_0 = V_0 = 0$ and the policies $\boldsymbol{u}_i$ defined as (13), the cost function V_i is updated by (14).

In the following part, we prove $\Phi_i(\boldsymbol{x}(k)) \leqslant V_{i+1}(\boldsymbol{x}(k))$ by mathematical induction.

First, we prove that it holds for $i = 0$. Noticing that

$$V_1(\boldsymbol{x}(k)) - \Phi_0(\boldsymbol{x}(k)) = \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}_0(k)) \geqslant 0 \tag{24}$$

thus for $i = 0$, we get

$$V_1(\boldsymbol{x}(k)) \geqslant \Phi_0(\boldsymbol{x}(k)) \tag{25}$$

Second, we assume that it holds for $i-1$, i.e., $V_i(\boldsymbol{x}(k)) \geqslant \Phi_{i-1}(\boldsymbol{x}(k)), \forall \boldsymbol{x}(k)$. Then for i, since

$$\Phi_i(\boldsymbol{x}(k)) = \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}_i(k)) + \Phi_{i-1}(\boldsymbol{x}(k+1)) \tag{26}$$

and

$$V_{i+1}(\boldsymbol{x}(k)) = \boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}_i(k)) + V_i(\boldsymbol{x}(k+1)) \tag{27}$$

hold, then we obtain

$$V_{i+1}(\boldsymbol{x}(k)) - \Phi_i(\boldsymbol{x}(k)) = V_i(\boldsymbol{x}(k+1)) - \Phi_{i-1}(\boldsymbol{x}(k+1)) \geqslant 0 \tag{28}$$

i.e., the following equation holds

$$\Phi_i(\boldsymbol{x}(k)) \leqslant V_{i+1}(\boldsymbol{x}(k)) \tag{29}$$

Therefore, the mathematical induction proof is completed.

Furthermore, from Lemma 1 we know that $V_i(\boldsymbol{x}(k)) \leqslant \Phi_i(\boldsymbol{x}(k))$, therefore we have

$$V_i(\boldsymbol{x}(k)) \leqslant \Phi_i(\boldsymbol{x}(k)) \leqslant V_{i+1}(\boldsymbol{x}(k)) \tag{30}$$

hence we can draw the conclusion that $\{V_i\}$ is a nondecreasing sequence with upper bound as shown in Lemma 2, i.e., $V_i \to V^*$ as $i \to \infty$.

We have just proved that the cost function converges to the value function of the discrete-time HJB equation, and according to (9) and (13), we can conclude that the corresponding control policy converges to the optimal one as well.

This completes the proof. □

3.2. Neural network implementation for GI-HDP algorithm

In the case of linear systems the value function and the control policy are quadratic and linear, respectively. In the nonlinear case, this is not necessarily true and therefore we need to use parametric structures or neural networks to approximate both $\boldsymbol{u}_i(\boldsymbol{x}(k))$ and $V_i(\boldsymbol{x}(k))$.

Assume that the number of hidden layer neurons is denoted by l, the weight matrix between the input layer and the hidden layer denoted by $\boldsymbol{v}$, the weight matrix between the hidden layer and the output layer denoted by $\boldsymbol{w}$, then output of three-layer NN is represented by:

$$\hat{\boldsymbol{f}}(X,\boldsymbol{v},\boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\sigma}(\boldsymbol{v}^{\mathrm{T}}X) \tag{31}$$

where $\boldsymbol{\sigma}(\boldsymbol{v}^{\mathrm{T}}X) \in \boldsymbol{R}^{l}$, $[\boldsymbol{\sigma}(\boldsymbol{z})]_i = \frac{e^{z_i}-e^{-z_i}}{e^{z_i}+e^{-z_i}}$, $i = 1,\ldots,l$ are the activation functions.

In order to implement the GI-HDP algorithm on (13) and (14), we now employ neural networks to approximate the value function and the corresponding optimal control policy. In the GI-HDP algorithm, there are three networks, which are critic network, model network and action network. All the neural networks are chosen as the three-layer feedforward networks. The inputs of the critic network and action network are $\boldsymbol{x}(k)$, and the inputs of the model network are $\boldsymbol{x}(k)$ and $\hat{\boldsymbol{u}}_i(k)$. The whole structure diagram is shown in Fig. 1, where the utility term denotes $\boldsymbol{x}(k)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}_i(k))$.

For an unknown plant, before carrying out the GI-HDP algorithm, we should first train the model network. For given $\boldsymbol{x}(k)$ and $\hat{\boldsymbol{u}}_i(k)$, we can obtain $\hat{\boldsymbol{x}}(k+1)$, and the output of the model network is denoted as

$$\hat{\boldsymbol{x}}(k+1) = \boldsymbol{w}_m^{\mathrm{T}}\boldsymbol{\sigma}\big(\boldsymbol{v}_m^{\mathrm{T}}I_m(k)\big) \tag{32}$$

where $I_m(k) = [\boldsymbol{x}(k)\hat{\boldsymbol{u}}_i(k)]$ is the input vector of the model network.

We define the error function of the model network as

$$e_m(k) = \hat{\boldsymbol{x}}(k+1) - \boldsymbol{x}(k+1) \tag{33}$$

The weights in the model network are updated to minimize the following performance measure:

$$E_m(k) = \frac{1}{2}e_m^{\mathrm{T}}e_m(k) \tag{34}$$

The weight update rule for model network is chosen as a gradient-based adaptation rule

$$\boldsymbol{w}_m(k+1) = \boldsymbol{w}_m(k) - \alpha_m\left[\frac{\partial E_m(k)}{\partial \boldsymbol{w}_m(k)}\right] \tag{35}$$

$$\boldsymbol{v}_m(k+1) = \boldsymbol{v}_m(k) - \alpha_m\left[\frac{\partial E_m(k)}{\partial \boldsymbol{v}_m(k)}\right] \tag{36}$$

where α_m is the learning rate of the model network.

After the model network is trained, its weights are kept unchanged.

The critic network is used to approximate the cost function $V_i(k)$. The output of the critic network is denoted as

$$\hat{V}_i(k) = \boldsymbol{w}_{ci}^{\mathrm{T}}\boldsymbol{\sigma}\big(\boldsymbol{v}_{ci}^{\mathrm{T}}\boldsymbol{x}(k)\big) \tag{37}$$

The target function can be written as

$$V_{i+1}(k) = \boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{Q}\boldsymbol{x}(k) + W(\boldsymbol{u}_i(k)) + \hat{V}_i(k+1) \tag{38}$$

Then we define the error function for the critic network as

$$e_{ci}(k) = \hat{V}_i(k) - V_{i+1}(k) \tag{39}$$

And the objective function to be minimized in the critic network is

$$E_{ci}(k) = \frac{1}{2}e_{ci}^2(k) \tag{40}$$

The weight update rule for the critic network is a gradient-based adaptation given by

$$\boldsymbol{w}_{c(i+1)}(k) = \boldsymbol{w}_{ci}(k) - \alpha_c\left[\frac{\partial E_{ci}(k)}{\partial \boldsymbol{w}_{ci}(k)}\right] \tag{41}$$

$$\boldsymbol{v}_{c(i+1)}(k) = \boldsymbol{v}_{ci}(k) - \alpha_c\left[\frac{\partial E_{ci}(k)}{\partial \boldsymbol{v}_{ci}(k)}\right] \tag{42}$$

where $\alpha_c > 0$ is the learning rate of the critic network.

In the action network the state $\boldsymbol{x}(k)$ is used as input to obtain the optimal control as the output of the network. The output can be formulated as

$$\hat{\boldsymbol{u}}_i(k) = \boldsymbol{w}_{ai}^{\mathrm{T}}\boldsymbol{\sigma}(\boldsymbol{v}_{ai}^{\mathrm{T}}\boldsymbol{x}(k)) \tag{43}$$

And the target control input is given as

$$\boldsymbol{u}_i(k) = \overline{\boldsymbol{U}}\boldsymbol{\varphi}\left(-\frac{1}{2}(\overline{\boldsymbol{U}}\boldsymbol{R})^{-1}\boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x}(k))\frac{\partial V_i(\boldsymbol{x}(k+1))}{\partial \boldsymbol{x}(k+1)}\right) \tag{44}$$

So we can define the output error of the action network as

$$e_{ai}(k) = \hat{\boldsymbol{u}}_i(k) - \boldsymbol{u}_i(k) \tag{45}$$

The weights in the action network are updated to minimize the following performance measure:

$$E_{ai}(k) = \frac{1}{2}e_{ai}^{\mathrm{T}}e_{ai}(k) \tag{46}$$

The update algorithm is similar to that in the critic network. By the gradient descent rule

$$\boldsymbol{w}_{a(i+1)}(k) = \boldsymbol{w}_{ai}(k) - \beta_a\left[\frac{\partial E_{ai}(k)}{\partial \boldsymbol{w}_{ai}(k)}\right] \tag{47}$$

$$\boldsymbol{v}_{a(i+1)}(k) = \boldsymbol{v}_{ai}(k) - \beta_a\left[\frac{\partial E_{ai}(k)}{\partial \boldsymbol{v}_{ai}(k)}\right] \tag{48}$$

where $\beta_a > 0$ is the learning rate of action network.

4. Simulation study

In this section, an example is provided to demonstrate the effectiveness of the control scheme proposed in this paper.

Consider the following nonlinear system in Ref. [20]:

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k)) + \boldsymbol{g}(\boldsymbol{x}(k))\boldsymbol{u}(k) \tag{49}$$

where $\boldsymbol{f}(\boldsymbol{x}(k)) = \begin{bmatrix} -0.8\boldsymbol{x}_2(k) \\ \sin(0.8\boldsymbol{x}_1(k) - \boldsymbol{x}_2(k)) + 1.8\boldsymbol{x}_2(k) \end{bmatrix}$, $\boldsymbol{g}(\boldsymbol{x}(k)) = \begin{bmatrix} 0 \\ -\boldsymbol{x}_2(k) \end{bmatrix}$, and the control constraint is set to $|\,\boldsymbol{u}\,| \leqslant 0.3$.

Define the performance functional as

$$J(\boldsymbol{x}(k),\boldsymbol{u}) = \sum_{i=k}^{\infty}\left\{\boldsymbol{x}(i)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(i) + 2\int_0^{\boldsymbol{u}(i)}\tanh^{-\mathrm{T}}(\overline{\boldsymbol{U}}^{-1}\boldsymbol{s})\overline{\boldsymbol{U}}\boldsymbol{R}\mathrm{d}\boldsymbol{s}\right\} \tag{50}$$

where the weight matrix is chosen as $\boldsymbol{Q} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$ and $\boldsymbol{R} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$.

For this plant, the control constraint is represented as $\overline{\boldsymbol{U}} = 0.3$. We choose three-layer feedforward neural net-

works as the critic network, the action network and the model network with the structures 2-8-1, 2-8-1 and 3-8-2, respectively. The initial weights of action network, critic network and model network are all set to be random in $[-1,1]$. It should be mentioned that the model network should be trained first. We train the model network for 1000 steps under the learning rate $\alpha_m = 0.1$. After the training of the model network is completed, the weights keep unchanged. Then the critic network and the action network are trained for 100 training cycles with each cycle of 1000 steps. In the training process, the learning rate $\beta_a = \alpha_c = 0.1$. Then for the given initial states $x_1(0) = 0.5$, $x_2(0) = 0.5$, we apply the saturated optimal control policy to the system for 100 time steps and obtain the state curves shown in Fig. 2, the control curve shown in Fig. 3 and the convergence process of the cost function shown in Fig. 4.

Moreover, in order to make comparison with the controller without considering the actuator saturation, we also present the controller designed by GI-HDP algorithm regardless of the saturation of the actuators. The state curves are shown in Fig. 5 and the control curve is shown in Fig. 6.

From the simulation results, we can see that the iterative cost function sequence does converge to the optimal value function quite rapidly, which also indicates the validity of the GI-HDP algorithm for dealing with constrained nonlinear system. Comparing Fig. 3 with Fig. 6, we can find that in Fig. 3 the saturation of the actuator has been overcome successfully.

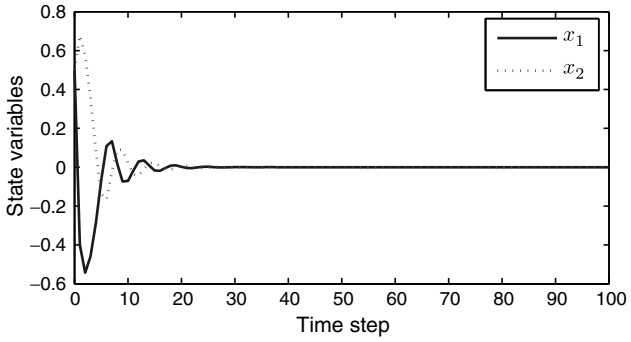

Fig. 2. The state variables curves.

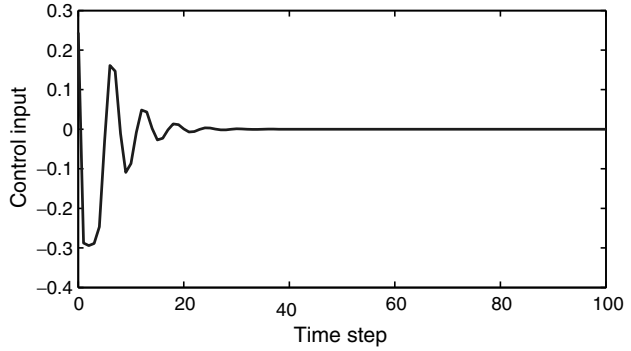

Fig. 3. The control input curve.

5. Conclusion

In this paper an effective algorithm is proposed to find the approximate optimal controller for a class of discrete-time constrained systems. First a new type of nonquadratic functional is defined to deal with the control constraint, and then the GI-HDP algorithm is introduced to solve the value function of the HJB equation with rigorous convergence analysis. Three neural networks are used as parametric structures to approximate the value function, compute the optimal control policy and model the unknown system, respectively. The simulation study has demonstrated the effectiveness of the proposed optimal control scheme.

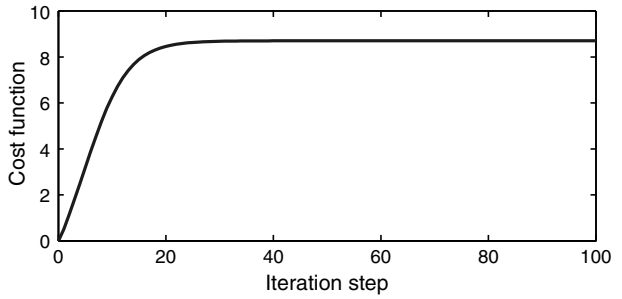

Fig. 4. The convergence process of the cost function.

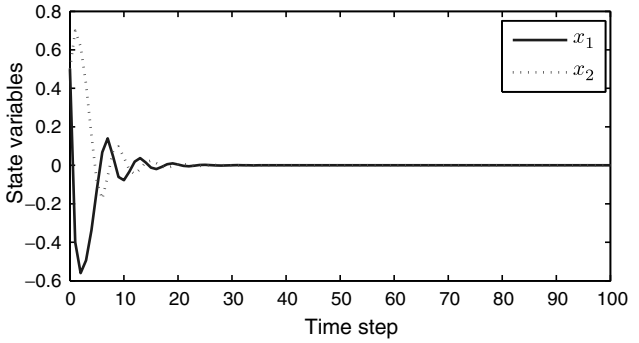

Fig. 5. The state variables curves.

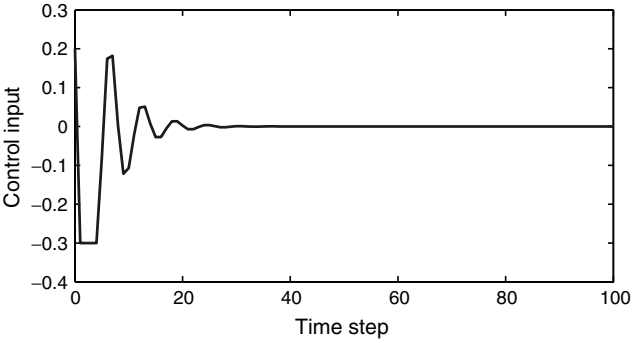

Fig. 6. The control input curve.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant Nos. 60534010, 60572070, 60774048, 60728307), the Program for Changjiang Scholars and Innovative Research Groups of China (60521003), the Research Fund for the Doctoral Program of China Higher Education (20070145015), and the National High Technology Research and Development Program of China (2006AA04Z183).

References

[1] Sussmann H, Sontag ED, Yang Y. A general result on the stabilization of linear systems using bounded controls. IEEE Trans Automatic Control 1994;39(12):2411–25.
[2] Bernstein DS. Optimal nonlinear, but continuous, feedback control of systems with saturating actuators. Int J Control 1995;62(5):1209–16.
[3] Saberi A, Lin Z, Teel A. Control of linear systems with saturating actuators. IEEE Trans Automatic Control 1996;41(3):368–78.
[4] Widrow B, Gupta N, Maitra S. Punish/reward: learning with a critic in adaptive threshold systems. IEEE Trans Syst Man Cybern 1973;3(5):455–65.
[5] Watkins C. Learning from delayed rewards. Ph.D. thesis, Cambridge University, Cambridge, England; 1989.
[6] Bertsekas DP, Tsitsiklis JN. Neuro-dynamic programming. Massachusetts: Athena Scientific; 1996.
[7] Liu X, Balakrishnan SN. Convergence analysis of adaptive critic based optimal control. In: Proceedings of American control conference, Chicago, IL, USA; 2000. p. 1929–33.
[8] Si J, Wang YT. On-line learning control by association and reinforcement. IEEE Trans Neural Netw 2001;12(2):264–76.
[9] Liu D, Xiong X, Zhang Y. Action-dependent adaptive critic designs. In: Proceedings of the INNS-IEEE international joint conference on neural networks, Washington, DC; 2001. p. 990–5.
[10] Liu D, Zhang HG. A neural dynamic programming approach for learning control of failure avoidance problems. Int J Intell Control Syst 2005;10(1):21–32.
[11] Murray JJ, Cox CJ, Lendaris GG, et al. Adaptive dynamic programming. IEEE Trans Syst Man Cybern C Appl Rev 2002;32(2):140–53.
[12] Abu-Khalaf M, Lewis FL. Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. Automatica 2005;41(5):779–91.
[13] Liu D. Approximate dynamic programming for self-learning control. Acta Automatica Sin 2005;31(1):13–8.
[14] Al-Tamimi A, Lewis FL. Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. In: Proceedings of the IEEE symposium on approximate dynamic programming and reinforcement learning, Honolulu, HI, USA; 2007. p. 38–43.
[15] Zhang HG, Wei QL, Liu D. On-line learning control for discrete nonlinear systems via an improved ADDHP method. In: Proceedings of the 4th International Symposium on Neural Networks. Nanjing, China; 2007, p. 387–96.
[16] Prokhorov DV, Wunsch DC. Adaptive critic designs. IEEE Trans Neural Netw 1997;8(5):997–1007.
[17] Lyshevski SE. Optimal control of nonlinear continuous-time systems: design of bounded controllers via generalized nonquadratic functionals. In: Proceedings of the American control conference, Philadelphia, USA; 1998. p. 205–9.
[18] Lyshevski SE. Nonlinear discrete-time systems: constrained optimization and application of nonquadratic costs. In: Proceedings of the American control conference, Philadelphia, USA; 1998. p. 3699–703.
[19] Beard R. Improving the closed-loop performance of nonlinear systems. Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, New York, USA; 1995.
[20] Chen Z, Jagannathan S. Generalized Hamilton–Jacobi–Bellman formulation-based neural network control of affine nonlinear discrete-time systems. IEEE Trans Neural Netw 2008;19(1):90–106.