Brief paper

Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming

Ding Wang\textsuperscript{a}, Derong Liu\textsuperscript{a,1}, Qinglai Wei\textsuperscript{a}, Dongbin Zhao\textsuperscript{a}, Ning Jin\textsuperscript{b}

\textsuperscript{a} State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, PR China
\textsuperscript{b} Department of Electrical and Computer Engineering, University of Illinois, Chicago, IL 60607, USA

A R T I C L E   I N F O

Article history:
Received 31 October 2010
Received in revised form 22 January 2012
Accepted 8 February 2012
Available online 25 June 2012

Keywords:
Adaptive critic designs
Adaptive dynamic programming
Approximated dynamic programming
Globalized dual heuristic programming
Intelligent control
Neural network
Optimal control

A B S T R A C T

An intelligent-optimal control scheme for unknown nonaffine nonlinear discrete-time systems with discount factor in the cost function is developed in this paper. The iterative adaptive dynamic programming algorithm is introduced to solve the optimal control problem with convergence analysis. Then, the implementation of the iterative algorithm via globalized dual heuristic programming technique is presented by using three neural networks, which will approximate at each iteration the cost function, the control law, and the unknown nonlinear system, respectively. In addition, two simulation examples are provided to verify the effectiveness of the developed optimal control approach.

1. Introduction

The main difference between optimal control of linear systems and nonlinear systems lies in that the latter often requires solving the nonlinear Hamilton–Jacobi–Bellman (HJB) equation instead of the Riccati equation (Abu-Khalaf & Lewis, 2005; Al-Tamimi, Lewis, & Abu-Khalaf, 2008; Primbs, Nevistic, & Doyle, 2000; Wang, Zhang, & Liu, 2009). For example, the discrete-time HJB (DTHJB) equation is more difficult to deal with than Riccati equation because it involves solving nonlinear partial difference equations. Although there were some methods that did not need to solve the HJB equation directly (e.g., Beard, Saridis, & Wen, 1997; Chen, Edgar, & Manousiouthakis, 2004), they were limited to handle some special classes of systems or they needed to perform very complex calculations. On the other hand, dynamic programming (DP) has been a useful technique in solving optimal control problems for many years (Bellman, 1957). However, it is often computationally untenable to run DP to obtain optimal solutions due to the “curse of dimensionality” (Bellman, 1957). Moreover, the backward direction of search precludes the application of DP in real-time control.

Artificial neural networks (ANN or NN) are an effective tool to implement intelligent control due to the properties of nonlinearity, adaptivity, self-learning, fault tolerance, and universal approximation of input–output mapping (Jagannathan, 2006; Werbos, 1992, 2008, 2009). Thus, it has been used for universal function approximation in adaptive/approximate dynamic programming (ADP) algorithms, which were proposed in Werbos (1992, 2008, 2009) as a method to solve optimal control problems forward-in-time. There are several synonyms used for ADP including “adaptive dynamic programming” (Lewis & Vrabie, 2009; Liu & Jin, 2008; Murray, Cox, Lendaris, & Saeks, 2002; Wang et al., 2009), “approximate dynamic programming” (Al-Tamimi et al., 2008; Werbos, 1992), “neuro-dynamic programming” (Bertsekas & Tsitsiklis, 1996), “neural dynamic programming” (Si & Wang, 2001), “adaptive critic designs” (Prokhorov & Wunsch, 1997), and “reinforcement learning” (Watkins & Dayan, 1992). As an effective intelligent control method, in recent years, ADP and the related research have gained much attention from researchers (Balakrishnan & Biega, 1996; Balakrishnan, Ding, & Lewis, 2008; Dierks, Thumati, & Jagannathan, 2009; Jagannathan &
An onlineardynamical system is said to be stabilizable with respect to (2) on \( \Omega \) if \( u(x) \) is continuous on a compact set \( \Omega_c \subset \mathbb{R}^m \), \( u(0) = 0 \), \( u \) stabilizes (1) on \( \Omega \), and \( \forall x_0 \in \Omega, J(x_0) \) is finite.

Note that Eq. (2) can be written as

\[
J(x_k) = x_k^T Q x_k + u_k^T R u_k + \gamma \sum_{p=k+1}^{\infty} \gamma^{p-k-1} U(x_p, u_p)
\]

\[
= x_k^T Q x_k + u_k^T R u_k + \gamma J(x_{k+1}).
\]

According to Bellman's optimality principle, the optimal cost function \( J^*(x_k) \) satisfies the DTHJB equation

\[
J^*(x_k) = \min_{u_k} \left\{ x_k^T Q x_k + u_k^T R u_k + \gamma J^*(x_{k+1}) \right\}.
\]

Besides, the optimal control \( u^* \) can be expressed as

\[
u^*(x_k) = \arg \min_{u_k} \left\{ x_k^T Q x_k + u_k^T R u_k + \gamma J^*(x_{k+1}) \right\}.
\]

By substituting (5) into (4), the DTHJB equation becomes

\[
J^*(x_k) = x_k^T Q x_k + u^T(x_k) R u^*(x_k) + \gamma J^*(x_{k+1}).
\]

It should be noticed that Definitions 1 and 2 are the same for linear systems. Moreover, when dealing with linear quadratic regulator problems, the DTHJB equation reduces to the Riccati equation which can be efficiently solved. For the general nonlinear case, however, it is considerably difficult to cope with the DTHJB equation directly. Therefore, we will develop an iterative ADP algorithm to solve it in the next section, based on Bellman's optimality principle and the greedy iteration approach.

3. Neuro-optimal control scheme based on iterative ADP algorithm via the GDHP technique

3.1. Derivation of the iterative algorithm

First, we start with the initial cost function \( V_0(x) = 0 \) and obtain the law of the single control vector \( v_0(x_k) \) as follows:

\[
v_0(x_k) = \arg \min_{u_k} \left\{ x_k^T Q x_k + u_k^T R u_k + \gamma V_0(x_{k+1}) \right\}.
\]

Then, we update the cost function as

\[
V_1(x_k) = x_k^T Q x_k + v_k^T(x_k) R v_k(x_k).
\]

Next, for \( i = 1, 2, \ldots \), the algorithm iterates between

\[
v_i(x_k) = \arg \min_{u_k} \left\{ x_k^T Q x_k + u_k^T R u_k + \gamma V_i(x_{k+1}) \right\}
\]

and

\[
V_{i+1}(x_k) = x_k^T Q x_k + v_i^T(x_k) R v_i(x_k) + \gamma V_i(F(x_k, v_i(x_k))).
\]

In the above recurrent iteration, \( i \) is the iteration index, while \( k \) is the time index. The cost function and control law are updated until they converge to the optimal ones. In the following, we will present the convergence proof of the iteration between (9) and (10) with the cost function \( V_i \rightarrow J^* \) and the control law \( v_i \rightarrow u^* \) as \( i \rightarrow \infty \).

3.2. Convergence analysis of the iterative algorithm

The convergence analysis provided here is an extension of that given in Al-Tamimi et al. (2008).
Lemma 1. Let $\{\mu_i\}$ be any arbitrary sequence of control laws and $\{v_i\}$ be the control laws as in (9). Define $V_i$ as in (10) and define $A_i$ as
\[ A_{i+1}(x_k) = x_k^TQx_k + \mu_i^T(x_k)Ru_k(x_k) + \gamma A_i(F(x_k, \mu_i(x_k))). \]  
(11)
If $V_0(\cdot) = A_0(\cdot) = 0$, then $V_{i+1}(x) \leq A_{i+1}(x)$, $\forall i$.
\[ \text{Proof.} \quad \text{It can be derived by noticing that } V_{i+1} \text{ is the result of minimizing the right-hand side of (10) with respect to the control input } u_k, \text{ while } A_{i+1} \text{ is a result of an arbitrary control input.} \]

Lemma 2. Let the sequence $\{V_i\}$ be defined as in (10). If the system is controllable, there is an upper bound $Y$ such that $0 \leq V_i(x_k) \leq Y$, $\forall i$.
\[ \text{Proof.} \quad \text{Let } \eta(x_k) \text{ be any admissible control input, and let } V_0(\cdot) = Z_0(\cdot) = 0, \text{ where } V_i \text{ is updated as in (10) and } Z_i \text{ is updated by}
\]
\[ Z_{i+1}(x_k) = x_k^TQx_k + \eta(x_k)Z_i(x_k) + \gamma Z_i(x_k+1). \]  
(12)
Noticing the difference
\[ Z_{i+1}(x_k) - Z_i(x_k) = \eta(x_k)Z_i(x_k+1) = \gamma^i(Z_{i+1}(x_k+1) - Z_{i-1}(x_k+1)) \]
\[ = \gamma^i(Z_{i-1}(x_k+2) - Z_{i-2}(x_k+2)) \]
\[ = \gamma^i(Z_{i-2}(x_k+3) - Z_{i-3}(x_k+3)) \]
\[ \vdots \]
\[ = \gamma^iZ_i(x_k+1) - Z_0(x_k+i) \]
\[ = \gamma^iZ_i(x_k+i), \]  
(13)
we can obtain
\[ Z_{i+1}(x_k) = \gamma^iZ_i(x_k+i) + Z_i(x_k) \]
\[ = \gamma^iZ_i(x_k+i) + \gamma^{i-1}Z_i(x_k+i-1) + Z_{i-1}(x_k) \]
\[ = \gamma^iZ_i(x_k+i) + \gamma^{i-1}Z_i(x_k+i-1) + \gamma^{i-2}Z_i(x_k+i-2) + \cdots + \gamma Z_i(x_k+1) + Z_i(x_k), \]  
(14)
and therefore,
\[ Z_{i+1}(x_k) = \sum_{j=0}^{i} \gamma^jZ_i(x_k+i) \]
\[ = \sum_{j=0}^{i} \gamma^j(x_k^TQx_k + \eta(x_k+i)R\Phi_k(x_k+i)) \]
\[ \leq \sum_{j=0}^{\infty} \gamma^j(x_k^TQx_k + \eta(x_k+i)R\Phi_k(x_k+i)). \]  
(15)
Since $\eta(x_k)$ is an admissible control input, i.e., $x_k \to 0$ as $k \to \infty$, there exists a finite $Y$ such that
\[ Z_{i+1}(x_k) \leq \sum_{j=0}^{\infty} \gamma^jZ_i(x_k+i) \leq Y, \quad \forall i. \]  
(16)
By using Lemma 1, we get
\[ V_{i+1}(x_k) \leq Z_{i+1}(x_k) \leq Y, \quad \forall i, \]  
(17)
and so the proof is completed. \[ \square \]

Based on Lemmas 1 and 2, we now present the convergence proof of the cost function sequence.

Theorem 1. Define the sequence $\{V_i\}$ as in (10) with $V_0(\cdot) = 0$, and the control law sequence $\{v_i\}$ as in (9). Then, we can conclude that $\{V_i\}$ is a nondecreasing sequence satisfying $V_i \leq V_{i+1}$, $\forall i$.
\[ \text{Proof.} \quad \text{Define a new sequence as}
\]
\[ \Phi_{i+1}(x_k) = x_k^TQx_k + v_{i+1}^TRv_{i+1}(x_k) + \gamma \Phi_i(x_k+i) \]  
(18)
with $\Phi_0(\cdot) = V_0(\cdot) = 0$. Now, we show that $\Phi_i(x_k) \leq V_{i+1}(x_k)$.
First, we prove that it holds for $i = 0$. Since $V_1(x_k) - \Phi_0(x_k) = x_k^TQx_k + v_0^TRv_0(x_k) \geq 0$, we have $\Phi_0(x_k) \leq V_1(x_k)$.
Second, we assume that it holds for $i - 1$, i.e., $\Phi_{i-1}(x_k) \leq V_{i-1}(x_k)$, $\forall x_k$. Then, for $i$, from (10) and (18), we get
\[ V_{i+1}(x_k) - \Phi_i(x_k) = \gamma(V_i(x_k+i) - \Phi_{i-1}(x_k+i)) \geq 0, \]  
(19)
i.e., $\Phi_i(x_k) \leq V_{i+1}(x_k)$. Thus, (22) is true for any $i$ by mathematical induction. Furthermore, according to Lemma 1, we know that $V_i(x_k) \leq \Phi_i(x_k)$. Combining with (22), we have
\[ V_i(x_k) \leq \Phi_i(x_k) \leq V_{i+1}(x_k), \]  
(23)
which completes the proof. \[ \square \]

According to Lemma 2 and Theorem 1, we can obtain that $\{V_i\}$ is a monotonically nondecreasing sequence with an upper bound, and therefore, its limit exists. Here, we define it as $\lim_{i \to \infty} V_i(x_k) = V_\infty(x_k)$ and present the following theorem.

Theorem 2. Let the cost function sequence $\{V_i\}$ be defined as in (10). Then, its limit satisfies
\[ V_\infty(x_k) = \min_{u_k} \{x_k^TQx_k + u_k^TRu_k + \gamma V_\infty(x_k+1)\}. \]  
(24)
\[ \text{Proof.} \quad \text{For any } u_k \text{ and } i, \text{ according to (10), we can derive}
\]
\[ V_i(x_k) \leq x_k^TQx_k + u_k^TRu_k + \gamma V_{i-1}(x_k+i). \]  
(25)
Combining with
\[ V_i(x_k) \leq V_\infty(x_k), \quad \forall i, \]  
(26)
which is obtained from (23), we have
\[ V_i(x_k) \leq x_k^TQx_k + u_k^TRu_k + \gamma V_\infty(x_k+1), \quad \forall i. \]  
(27)
Let $i \to \infty$, then we can obtain
\[ V_\infty(x_k) \leq x_k^TQx_k + u_k^TRu_k + \gamma V_\infty(x_k+1). \]  
(28)
Note that in the above equation, $u_k$ is chosen arbitrarily, thus, it implies that
\[ V_\infty(x_k) \leq \min_{u_k} \{x_k^TQx_k + u_k^TRu_k + \gamma V_\infty(x_k+1)\}. \]  
(29)
On the other hand, since the cost function sequence satisfies
\[ V_i(x_k) = \min_{u_k} \{x_k^TQx_k + u_k^TRu_k + \gamma V_{i-1}(x_k+i)\} \]  
(30)
for any $i$, considering (26), we have
\[ V_\infty(x_k) \geq \min_{u_k} \{x_k^TQx_k + u_k^TRu_k + \gamma V_{i-1}(x_k+i)\}, \quad \forall i. \]  
(31)
Let $i \to \infty$, then we can get
\[ V_\infty(x_k) \geq \min_{u_k} \{x_k^TQx_k + u_k^TRu_k + \gamma V_\infty(x_k+1)\}. \]  
(32)
Based on (29) and (32), we can conclude that (24) is true. \[ \square \]
Remark 1. Let \( \lim_{k \to \infty} \nu_i(x_k) = v_\infty(x_k) \). According to Theorem 2 and the relationship between (9) and (10), we have \( V_{\infty}(x_k) = \min_{u_k} \{ x_k^T Q x_k + u_k^T R u_k + \gamma V_{\infty}(x_{k+1}) \} = x_k^T Q x_k + v_\infty^T(x_k) R v_\infty(x_k) + \gamma V_{\infty}(F(x_k, v_\infty(x_k))) \). (33) where \( v_\infty(x_k) = \arg \min_{u_k} \{ x_k^T Q x_k + u_k^T R u_k + \gamma V_{\infty}(x_{k+1}) \} \). (34)

Observing (33) and (34), and then (4) and (5), we can find that \( V_{\infty}(x_k) = f^*(x_k) \) and \( v_\infty(x_k) = u^*(x_k) \). In other words, \( \lim_{k \to \infty} V_i(x_k) = f^*(x_k) \) and \( \lim_{k \to \infty} v_i(x_k) = u^*(x_k) \).

3.3. NN implementation of the iterative algorithm

For carrying out the iterative ADP algorithm, we need to use a function approximation structure, such as NN, to approximate both \( v_i(x_k) \) and \( V_i(x_k) \).

Let the number of hidden layer neurons be denoted by \( l \), the weight matrix between the input layer and hidden layer be denoted by \( \lambda \), and the weight matrix between the hidden layer and output layer be denoted by \( \omega \). Then, the output of three-layer NN is formulated as

\[
\hat{F}(X, \nu, \alpha) = \omega^T \sigma (v^T X)
\]

where \( \sigma (v^T X) \in \mathbb{R}^l, [\sigma (z)]_q = (e^{vq} - e^{-vq}) / (e^{vq} + e^{-vq}), \) \( q = 1, 2, \ldots, l \), are the activation functions.

Now, we implement the iterative ADP algorithm via the GDHP technique. It consists of a model network, two critic networks and an action network, which are all chosen as three-layer feedforward NNS. The whole structure diagram is shown in Fig. 1, where

\[
\text{DER} = \left( \frac{\partial \hat{X}_k+1}{\partial x_k} + \frac{\partial \hat{X}_k+1}{\partial \nu_i(x_k)} \right)^T
\]

In order to avoid the requirement of knowing \( F(x_k, u_k) \), we should train the model network before carrying out the main iterative process. For given \( x_k \) and \( \hat{e}_i(x_k) \), we can obtain the output of the model network as

\[
\hat{X}_{k+1} = \omega^T \sigma (v^T x_k) + \partial \hat{X}_k+1(x_k) \partial \hat{e}_i(x_k)^T
\]

We define the error function of the model network as

\[
e_{mk} = \hat{X}_{k+1} - X_{k+1}
\]

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

\[
E_{mk} = \frac{1}{2} e_{mk}^T e_{mk}
\]

Using the gradient-based adaptation rule, the weights can be updated as

\[
\omega_m(j + 1) = \omega_m(j) - \alpha_m \frac{\partial E_{mk}}{\partial \omega_m(j)}
\]

\[
v_m(j + 1) = v_m(j) - \alpha_m \frac{\partial E_{mk}}{\partial v_m(j)}
\]

where \( \alpha_m > 0 \) is the learning rate of the model network, and \( j \) is the iterative step for updating the weight parameters.

The weights of the model network are kept unchanged after the training process is finished.

The critic network is used to approximate both \( V_i(x_k) \) and its derivative \( \partial V_i(x_k) / \partial x_k \), which is denoted as \( \lambda_i(x_k) \). The input of critic network is \( x_k \), while the output is given by

\[
\hat{V}_i(x_k) = \omega^T_1 \sigma (v^T x_k)
\]

and

\[
\hat{\lambda}_i(x_k) = \omega^T_2 \sigma (v^T x_k)
\]

The target functions can be written as

\[
V_{i+1}(x_k) = x_k^T Q x_k + v^T_1(x_k) R v_1(x_k) + \gamma \hat{V}_i(\hat{X}_{i+1})
\]

and

\[
\lambda_{i+1}(x_k) = \frac{\partial (x_k^T Q x_k + v^T_1(x_k) R v_1(x_k))}{\partial x_k} + \gamma \frac{\partial \hat{V}_i(\hat{X}_{i+1})}{\partial x_k}
\]

\[
= 2Q x_k + 2 \left( \frac{\partial \hat{v}_1(x_k)}{\partial x_k} \right)^T R v_1(x_k)
\]

\[
+ \gamma \left( \frac{\partial \hat{X}_{i+1}}{\partial x_k} + \frac{\partial \hat{\lambda}_i(x_k)}{\partial \hat{v}_1(x_k)} \frac{\partial \hat{v}_1(x_k)}{\partial x_k} \right)^T \hat{\lambda}_i(\hat{X}_{i+1})
\]

Note that Eq. (46) is simply the derivative form of (45), and therefore, the two are equivalent in principle. Then, the error functions can be defined as

\[
e^1_{ck} = \hat{V}_i(x_k) - V_{i+1}(x_k)
\]

and

\[
e^2_{ck} = \hat{\lambda}_i(x_k) - \lambda_{i+1}(x_k)
\]

Since the GDHP technique is a combination of HDP and DHP techniques, we choose the objective function to be minimized by the critic network as

\[
E_{ck} = (1 - \theta) E^1_{ck} + \theta E^2_{ck}
\]

where

\[
E^1_{ck} = \frac{1}{2} e^1_{ck} e^1_{ck}
\]

and

\[
E^2_{ck} = \frac{1}{2} e^2_{ck} e^2_{ck}
\]
The weight update rule for the critic network is also a gradient-based adaptation given by
\[
\omega_c(j + 1) = \omega_c(j) - \alpha_c \left[ (1 - \theta) \frac{\partial E_{\text{cik}}}{\partial \omega_i(j)} + \theta \frac{\partial E_{\text{cik}}}{\partial \nu_k(j)} \right],
\]
(52)

\[
v_c(j + 1) = v_c(j) - \alpha_c \left[ (1 - \theta) \frac{\partial E_{\text{cik}}}{\partial v_c(j)} + \theta \frac{\partial E_{\text{cik}}}{\partial v_c(j)} \right],
\]
(53)

where \(\alpha_c > 0\) is the learning rate of the critic network, \(j\) is the inner-loop iterative step for updating the weight parameters, and \(0 \leq \theta \leq 1\) is a parameter that adjusts how HDP and DHP are combined in GDHP. When \(\theta = 0\), the training of the critic network reduces to a pure DHP, while \(\theta = 1\) does the same for DHP.

In the action network, \(x_i\) is used as the input and the output is
\[
\hat{v}_i(x_k) = \omega_{\text{aT}}(j)\sigma(\nu_{\text{aT}}x_k).
\]
(54)

The target control input is given by
\[
v_i(x_k) = \arg\min_u \left\{ c_k^i Q x_k + u_k^i R u_k + \gamma \hat{V}_k(x_{k+1}) \right\}.
\]
(55)

The error function of the action network can be defined as
\[
e_{\text{aik}} = \hat{v}_i(x_k) - v_i(x_k).
\]
(56)

The weights of the action network are updated to minimize
\[
E_{\text{aik}} = \frac{1}{2} \hat{v}_{\text{aik}}^T e_{\text{aik}}.
\]
(57)

Similarly, the weight update algorithm is
\[
\omega_{\text{ai}}(j + 1) = \omega_{\text{ai}}(j) - \alpha_{\text{ai}} \left[ \frac{\partial E_{\text{aik}}}{\partial \omega_{\text{ai}}(j)} \right],
\]
(58)

\[
v_{\text{ai}}(j + 1) = v_{\text{ai}}(j) - \alpha_{\text{ai}} \left[ \frac{\partial E_{\text{aik}}}{\partial v_{\text{ai}}(j)} \right],
\]
(59)

where \(\alpha_{\text{ai}} > 0\) is the learning rate of the action network, and \(j\) is the inner-loop iterative step for updating the weight parameters.

**Remark 2.** According to Remark 1, \(V_i \to \nu_i^*\) as \(i \to \infty\). Since \(\lambda_i(x_k) = \partial V_i(x_k)/\partial x_k\), we can conclude that the sequence \(\{\lambda_i\}\) is also convergent with \(\lambda_i \to \lambda^*\) as \(i \to \infty\).

**Remark 3.** Since we cannot implement the iteration until \(i \to \infty\) in practical applications, we should run the algorithm with a prespecified accuracy \(\varepsilon\) to test the convergence of the cost function sequence. When \(|V_{i+1}(x_k) - V_i(x_k)| < \varepsilon\), we consider the cost function sequence has converged sufficiently and stop running the iterative GDHP algorithm.

### 4. Simulation studies

In this section, two examples are provided to demonstrate the effectiveness of the iterative GDHP algorithm.

#### 4.1. Example 1

Consider the following nonlinear system:
\[
x_{k+1} = x_k + \sin(x_k + u_k),
\]
(60)

where \(x_k \in \mathbb{R}, u_k \in \mathbb{R}, k = 1, 2, \ldots\). The utility function is chosen as \(U(x_k, u_k) = x_k^2 + u_k^2\). It can be seen that \(x_k = 0\) is an equilibrium state of system (60). However, the system is unstable at this equilibrium, since \(\partial x_{k+1}/\partial x_k|_{(0,0)} = 2 > 1\).

We choose three-layer feedforward NNs as model network, critic network and action network with structures 2–8–1, 1–8–2, and 1–8–1, respectively, and implement the algorithm at time instant \(k = 0\). The initial weights of the three NNs are all set to be random in \([-1, 1]\). Note that the model network should be trained first. We train the model network for 100 time steps using 500 data samples under the learning rate \(\alpha_m = 0.1\). After the model network is trained, its weights are kept unchanged. Then, let the discount factor \(\gamma = 1\) and the adjusting parameter \(\theta = 0.5\), we train the critic network and action network for 120 iterations (i.e., for \(i = 1, 2, \ldots, 120\)) with 2000 training epochs for each iteration to make sure the given accuracy \(\varepsilon = 10^{-6}\) is reached. In the training process, the learning rate \(\alpha_a = \alpha_a = 0.05\). The convergence processes of the cost function and its derivative of GDHP algorithm are shown in Fig. 2, for \(k = 0\) and \(x_0 = 1.5\). We can see that the iterative cost function sequence does converge to the optimal value quite rapidly, which also indicates the validity of the iterative GDHP algorithm. For the same problem, the iterative GDHP algorithm takes about 16 s while HDP takes about 117 s before satisfactory results are obtained.

Moreover, in order to make comparison with DHP algorithm, we also present the controller designed by DHP algorithm. Then, for given initial state \(x_0 = 1.5\), we apply the optimal control laws designed by GDHP and DHP techniques to the system for 15 time steps, respectively, and obtain the state curves as shown in Fig. 3. The corresponding control curves are shown in Fig. 4. It can be seen...
from the simulation results that the controller derived by the GDHP algorithm has a better performance than the DHP algorithm.

To show the discount factor has evident impact on our iterative algorithm, in this case, we choose the discount factor $\gamma = 0.9$ and set the other parameters the same as above. Then, we train the critic network and action network for 80 iterations and find that the given accuracy $\varepsilon = 10^{-6}$ has been reached, which demonstrates that smaller discount factor can insure quicker convergence of the cost function sequence. Next, we will show the discrepancy of the state and control curves under different iterations to prove the usefulness of the iterative algorithm. For the same initial state $x_0 = 1.5$, we apply different control laws to the controlled plant for 15 time steps and obtain simulation results as follows. The state curves are shown in Fig. 5, and the corresponding control inputs are shown in Fig. 6. From the simulation results, we can see that the closed-loop system is divergent when using the control law obtained in the first iteration. However, the system responses become better and better as the iteration numbers increasing from 3 to 80. Besides, the responses basically remain unchanged when the iteration number is larger than 5, which verifies the effectiveness of the proposed iterative GDHP algorithm.

4.2. Example 2

Consider the nonlinear discrete-time system given by

$$\begin{align*}
    x_{k+1} &= \begin{bmatrix}
        -x_{1k} x_{2k} \\
        1.5 x_{2k} + \sin(x_{2k}^2 + u_k)
    \end{bmatrix} \\
    x_k &= [x_{1k} \ x_{2k}]^T \in \mathbb{R}^2, \ u_k \in \mathbb{R}, \ k = 1, 2, \ldots
\end{align*}$$

(61)

where $x_k = [x_{1k} \ x_{2k}]^T \in \mathbb{R}^2$, $u_k \in \mathbb{R}$, $k = 1, 2, \ldots$. The utility function is also set as $U(x_k, u_k) = x_k^T x_k + u_k^T u_k$.

In this example, we also choose three-layer feedforward NNs as the model network, the critic network and the action network, but with structures 3–8–2, 2–8–3, and 2–8–1, respectively. Here, we train the critic network and action network for 50 iterations while keeping the other parameters the same as the above example. The convergence processes of the cost function and its derivative of the iterative GDHP algorithm are shown in Fig. 7, which verify the theoretical conjectures of Theorems 1–2 and Remarks 1–2. Furthermore, for given initial state $x_{10} = 0.5$ and $x_{20} = -1$, we apply the optimal control law designed by the iterative GDHP algorithm to (61) for 25 time steps, and obtain the state curves and the corresponding control curves as shown in Figs. 8 and 9, respectively. These simulation results verify the excellent performance of the controller derived by the iterative GDHP algorithm.

5. Conclusion

In this paper, an effective iterative ADP algorithm with convergence analysis is given to design the near optimal controller for unknown nonaffine nonlinear discrete-time systems with discount factor in the cost function. The GDHP technique is introduced to implement the algorithm. Three NNs are used as
parametric structures to approximate the cost function and its derivative, the control law and identify the unknown nonlinear system, respectively. The simulation studies demonstrated the validity of the proposed optimal control scheme.

References


Ding Wang received his B.S. degree in mathematics from Zhengzhou University of Light Industry, Zhengzhou, China and his M.S. degree in operational research and cybernetics from Northeastern University, Shenyang, China, in 2007 and 2009, respectively. He is currently working toward a Ph.D. degree in the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include adaptive dynamic programming, neural networks, and intelligent control.

Derong Liu received his Ph.D. degree in electrical engineering from the University of Notre Dame in 1994. Liu was a Staff Fellow with General Motors Research and Development Center, Warren, MI, from 1993 to 1995. He was an Assistant Professor in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, from 1995 to 1999. He joined the University of Illinois at Chicago in 1999, and became a Full Professor of electrical and computer engineering and of computer science in 2006. He was selected for the “100 Talents Program” by the Chinese Academy of Sciences in 2008. He has published 10 books. Dr. Liu has been an Associate Editor of several IEEE publications. Currently, he is the Editor-in-Chief of the IEEE Transactions on Neural Networks and Learning Systems, and an Associate Editor of the IEEE Transactions on Control Systems Technology and several other journals. He was an elected AdCom member of the IEEE Computational Intelligence Society (2006–2008). He received the Faculty Early Career Development (CAREER) award from the National Science Foundation (1999), the University Scholar Award from University of Illinois (2006–2009), and the Overseas Outstanding Young Scholar Award from the National Natural Science Foundation of China (2008). He is a Fellow of the IEEE.
Qinglai Wei received his B.S. degree in automation, his M.S. degree in control theory and control engineering, and his Ph.D. degree in control theory and control engineering, from the Northeastern University, Shenyang, China, in 2002, 2005, and 2008, respectively. From 2009 to 2010, he was a post-doctoral fellow with the Institute of Automation, Chinese Academy of Sciences. He is currently an Assistant Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. His research interests include neural-networks-based control, nonlinear control, adaptive dynamic programming, and their industrial applications.

Dongbin Zhao received his B.S., M.S., and Ph.D. degrees in material processing engineering from the Harbin Institute of Technology, Harbin, China, in 1994, 1996, and 2000, respectively. Dr. Zhao was a post-doctoral fellow with Tsinghua University, Beijing, China, from May 2000 to January 2002. He is currently an Associate Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing. He has published one book and over 30 international journal papers. His current research interests include the area of computational intelligence, adaptive dynamic programming, robotics, intelligent transportation systems, and process simulation. Dr. Zhao has been a Senior Member of the Chinese Mechanical Engineering Society since 2006. He served as many international program committee member and the Finance Chair of the International Symposium on Neural Networks in 2011. He was Guest Editor of several international journals. He received the Second Award for Scientific Progress of National Defense from the Commission of Science Technology and Industry for National Defense of China in 1999, the First Award for Scientific Progress of Chinese Universities, Ministry of Education of China in 2001, the Third Award for Scientific and Technology Progress from the China Petroleum and Chemical Industry Association in 2009, and the First Award for Scientific and Technology Progress from the China Petroleum and Chemical Industry Association in 2010.

Ning Jin received his Ph.D. degree in electrical and computer engineering from the University of Illinois at Chicago in 2012. He was an Associate Professor in the Department of Mathematics at Nanjing Normal University. From 2002 to 2005, he was a visiting scholar in the Department of Mathematics, Statistics, and Computer Science at the University of Illinois at Chicago. He is currently an Instructor in the Department of Electrical and Computer Engineering at the University of Illinois at Chicago. His research interests include optimal control and dynamic programming, artificial intelligence, pattern recognition, neural networks, and wavelet analysis.