

## Chapter 2

# Value Iteration ADP for Discrete-Time Nonlinear Systems

### 2.1 Introduction

The nonlinear optimal control has been the focus of control fields for many decades [7, 10, 23, 39]. It often needs to solve the nonlinear Bellman equation. The Bellman equation is more difficult to work with than the Riccati equation because it involves solving nonlinear partial difference equations. Although dynamic programming has been a useful technique in handling optimal control problems for nonlinear systems, it is often computationally untenable to perform it to obtain the optimal solutions because of the well-known “curse of dimensionality” [9, 14]. Fortunately, relying on the strong abilities of self-learning and adaptivity of artificial neural networks (ANNs), the ADP method was proposed by Werbos [46, 47] to deal with optimal control problems forward-in-time. In recent years, ADP and related research have gained much attention from scholars (see the recent books [22, 40, 50] and the references cited therein).

It is important to note that the iterative methods are often used in ADP to obtain the solution of Bellman equation indirectly and have received more and more attention. In [24], iterative ADP algorithms were classified into two main schemes, namely policy iteration (PI) and value iteration (VI) [38, 40], respectively. PI algorithms contain policy evaluation and policy improvement [18, 38, 40]. An initial stabilizing control law is required, which is often difficult to obtain. Comparing to VI algorithms, in most applications, PI would require fewer iterations as a Newton’s method, but every iteration is more computationally demanding. VI algorithms solve the optimal control problem without requirement of an initial stabilizing control law, which is easy to implement. However, the stabilizing control law cannot be obtained until the value function converges. This means that only the converged optimal control (function of the system state  $x_k$ )  $u^*(x_k)$  can be used to control the nonlinear system, where the iterative controls  $v_i(x_k)$ ,  $i = 0, 1, \dots$ , may be invalid. Hence, the computational efficiency of the VI ADP method is low. Besides, most of the VI algorithms are implemented off-line which limits their applications very much. In this chapter, the

VI ADP approach is employed to solve the optimal control problems of discrete-time nonlinear systems, where several value iteration schemes are developed to overcome the above difficulties.

In the beginning, an ADP scheme based on general value iteration (GVI) is developed to obtain optimal control for discrete-time affine nonlinear systems [25]. The selection of initial value function is different from the traditional VI algorithm, and a new method is introduced to demonstrate the convergence property and the convergence speed of value functions. The control law obtained at each iteration can stabilize the system under some conditions. To facilitate the implementation of the iterative scheme, three NNs with Levenberg–Marquardt (LM) training algorithm are used to approximate the unknown system, the value function, and the control law, respectively. Then, the GVI-based ADP method is generalized to solve infinite-horizon optimal tracking control problem for a class of discrete-time nonlinear systems [45]. The GVI-based ADP algorithm permits an arbitrary positive-semidefinite function to initialize it, and it is more advantageous than traditional VI algorithms which starts from a zero function. Next, the ADP approach is used for designing the optimal controller of discrete-time nonlinear systems with unknown dynamics and constrained inputs [28]. The iterative ADP algorithm is developed to solve the constrained optimal control problem based on VI algorithm, which can be regarded as a special case of GVI. Three NNs are employed for approximating the unknown nonlinear system dynamics, the value function and its derivatives, and the control law, respectively, under the framework of globalized dual heuristic programming (GDHP) technique. Finally, an iterative  $\theta$ -ADP technique is developed to solve optimal control problems for infinite-horizon discrete-time nonlinear systems [44]. The condition of initial admissible control in PI algorithm is avoided. It is proved that all the iterative controls obtained in the iterative  $\theta$ -ADP algorithm can stabilize the nonlinear system, which means that the iterative  $\theta$ -ADP algorithm is feasible for implementations both online and off-line. Convergence analysis of the value function is presented to guarantee that the iterative value function can converge to the optimum monotonically.

## 2.2 Optimal Control of Nonlinear Systems Using General Value Iteration

Consider the discrete-time nonlinear systems described by

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \dots, \quad (2.2.1)$$

where  $x_k \in \mathbb{R}^n$  is the state vector at time  $k$ ,  $u_k = u(x_k) \in \mathbb{R}^m$  is the state feedback control vector, and  $F(\cdot, \cdot)$  is the nonlinear system function. Let  $x_0$  be the initial state. Let the following assumptions hold throughout this chapter.

**Assumption 2.2.1**  $F(0, 0) = 0$ , and the state feedback control law  $u(\cdot)$  satisfies  $u(0) = 0$ , i.e.,  $x_k = 0$  is an equilibrium state of system (2.2.1) under the control  $u_k = 0$ .

**Assumption 2.2.2**  $F(x_k, u_k)$  is Lipschitz continuous on a compact set  $\Omega \subset \mathbb{R}^n$  containing the origin.

**Assumption 2.2.3** System (2.2.1) is controllable in the sense that there exists a continuous control law on  $\Omega$  that asymptotically stabilizes the system.

First, in Sects. 2.2.1 and 2.2.2, we develop a GVI-based optimal control scheme for discrete-time nonlinear systems with affine form [25]. Consider the following affine nonlinear systems

$$x_{k+1} = f(x_k) + g(x_k)u_k, \quad k = 0, 1, 2, \dots, \quad (2.2.2)$$

where  $f(\cdot) \in \mathbb{R}^n$  and  $g(\cdot) \in \mathbb{R}^{n \times m}$  are differentiable and  $f(0) = 0$ . Our goal is to find a state feedback control law  $u(\cdot)$  such that  $u_k = u(x_k)$  can stabilize the system (2.2.2) and simultaneously minimize the infinite-horizon cost function given by

$$J(x_0, u) = J^u(x_0) = \sum_{k=0}^{\infty} U(x_k, u_k), \quad (2.2.3)$$

where  $U(x_k, u_k)$  is a positive-definite utility function, i.e.,  $U(0, 0) = 0$  and for all  $(x_k, u_k) \neq (0, 0)$ ,  $U(x_k, u_k) > 0$ . Note that the control law  $u(\cdot)$  must not only stabilize the system on  $\Omega$  but also guarantee (2.2.3) to be finite, i.e., the control law must be admissible.

**Definition 2.2.1** (cf. [5, 51]) A control law  $u(\cdot)$  is said to be admissible with respect to (2.2.2) (or (2.2.1)) on  $\Omega$  if  $u(\cdot)$  is continuous on  $\Omega$ ,  $u(0) = 0$ ,  $u_k = u(x_k)$  stabilizes (2.2.2) (or (2.2.1)) on  $\Omega$ , and  $J(x_0, u)$  is finite,  $\forall x_0 \in \Omega$ .

Let  $\mathcal{A}(\Omega)$  be the set of admissible control laws associated with the controllable set  $\Omega$  of states. For optimal control problems we study in this book, the set  $\mathcal{A}(\Omega)$  is assumed to be nonempty, i.e.,  $\mathcal{A}(\Omega) \neq \emptyset$ .

Define the optimal cost function as

$$J^*(x_k) = \inf_u \{J(x_k, u) : u \in \mathcal{A}(\Omega)\}.$$

According to [9, 11, 14, 23], the optimal cost function  $J^*(x_k)$  satisfies the Bellman equation

$$J^*(x_k) = \min_{u_k} \{U(x_k, u_k) + J^*(x_{k+1})\}. \quad (2.2.4)$$

Equation (2.2.4) is the Bellman's principle of optimality for discrete-time systems. Its importance lies in the fact that it allows one to optimize over only one control

vector at a time by working backward in time. The optimal control law  $u^*(\cdot)$  should satisfy

$$u_k^* = u^*(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + J^*(x_{k+1})\}. \quad (2.2.5)$$

In general, the utility function can be chosen as the quadratic form given by

$$U(x_k, u_k) = x_k^\top Q x_k + u_k^\top R u_k, \quad (2.2.6)$$

where  $Q \in \mathbb{R}^{n \times n}$  and  $R \in \mathbb{R}^{m \times m}$  are positive-definite matrices. The optimal control  $u_k^*$  satisfies the first-order necessary condition, from which we obtain

$$u_k^* = -\frac{1}{2}R^{-1} \left( \frac{\partial x_{k+1}}{\partial u_k} \right)^\top \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}} = -\frac{1}{2}R^{-1} g^\top(x_k) \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}}.$$

Equation (2.2.4) reduces to Riccati equation in the case of linear quadratic regulator problem. However, in the nonlinear case, the cost function of the optimal control problem cannot be obtained directly. Therefore, we will solve the Bellman equation by the GVI algorithm.

### 2.2.1 Convergence Analysis

Since direct solution of the Bellman equation is computationally intensive, we present an iterative ADP algorithm in a general framework based on Bellman's principle of optimality. Define the value function for system (2.2.2) as

$$V(x_k) = J^u(x_k).$$

As we have explained in Chap. 1,  $V(x_k)$  is a short notation of  $V(x_k, u)$  or  $V^u(x_k)$  for convenience of presentation.

First, the initial value function is chosen as a quadratic form given by

$$V_0(x_k) = x_k^\top P_0 x_k, \quad (2.2.7)$$

where  $P_0$  is a positive-definite matrix. Then, for  $i = 0, 1, 2, \dots$ , the GVI-based ADP algorithm iterates between a sequence of control laws  $v_i(x_k)$ ,

$$\begin{aligned} v_i(x_k) &= \arg \min_{u_k} \{x_k^\top Q x_k + u_k^\top R u_k + V_i(x_{k+1})\} \\ &= \arg \min_{u_k} \{x_k^\top Q x_k + u_k^\top R u_k + V_i(f(x_k) + g(x_k)u_k)\}, \end{aligned} \quad (2.2.8)$$

and a sequence of value functions  $V_{i+1}(x_k)$ ,

$$\begin{aligned}
V_{i+1}(x_k) &= \min_{u_k} \{x_k^\top Q x_k + u_k^\top R u_k + V_i(x_{k+1})\} \\
&= x_k^\top Q x_k + v_i^\top(x_k) R v_i(x_k) + V_i(f(x_k) + g(x_k)v_i(x_k)). \quad (2.2.9)
\end{aligned}$$

From the  $i$ th iteration of the algorithm in (2.2.8)–(2.2.9), we obtain  $v_i(x_k)$  and  $V_{i+1}(x_k)$ .

In the above VI algorithm, (2.2.8) is called policy improvement (or policy update) and (2.2.9) is called value function update [38, 40]. In (2.2.8), an improved policy that is better or at least not worse than the previous policy is obtained using the current value function. In (2.2.9), an updated value function, to be used in the next iteration, is calculated using the current policy. It is a one-step procedure for approximating the value function corresponding to the current policy, and thus, (2.2.9) is also called one-step policy evaluation [38].

If we want the outcomes of the  $i$ th iteration to be  $V_i(x_k)$  and  $v_i(x_k)$ , the iterative algorithm (2.2.8)–(2.2.9) can be rewritten as follows.

From the initial value function given in (2.2.7), we obtain the control law  $v_0(x_k)$  by

$$\begin{aligned}
v_0(x_k) &= \arg \min_{u_k} \{x_k^\top Q x_k + u_k^\top R u_k + V_0(x_{k+1})\} \\
&= \arg \min_{u_k} \{x_k^\top Q x_k + u_k^\top R u_k + V_0(f(x_k) + g(x_k)u_k)\}, \quad (2.2.10)
\end{aligned}$$

where  $V_0(x_{k+1}) = x_{k+1}^\top P_0 x_{k+1}$  according to (2.2.7). For  $i = 1, 2, \dots$ , the GVI-based ADP algorithm iterates between value function update

$$\begin{aligned}
V_i(x_k) &= \min_{u_k} \{x_k^\top Q x_k + u_k^\top R u_k + V_{i-1}(x_{k+1})\} \\
&= x_k^\top Q x_k + v_{i-1}^\top(x_k) R v_{i-1}(x_k) + V_{i-1}(f(x_k) + g(x_k)v_{i-1}(x_k)), \quad (2.2.11)
\end{aligned}$$

and policy improvement

$$\begin{aligned}
v_i(x_k) &= \arg \min_{u_k} \{x_k^\top Q x_k + u_k^\top R u_k + V_i(x_{k+1})\} \\
&= \arg \min_{u_k} \{x_k^\top Q x_k + u_k^\top R u_k + V_i(f(x_k) + g(x_k)u_k)\}. \quad (2.2.12)
\end{aligned}$$

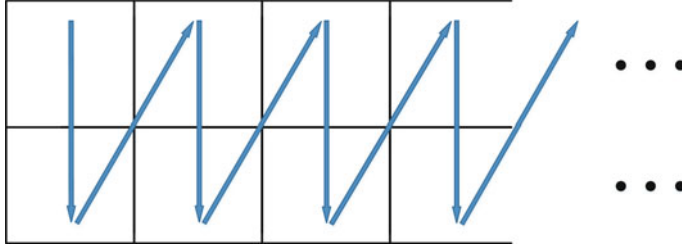
Now, from the  $i$ th iteration of the algorithm in (2.2.10)–(2.2.12), we obtain  $V_i(x_k)$  and  $v_i(x_k)$ . Note that it is a simple calculation in (2.2.11) to update the value function using the previous policy and previous value function, while in (2.2.12), it performs the minimization so that an improved policy that is better or at least not worse than the previous policy is obtained using the newly updated value function.

Since our goal in optimal control design is to obtain an optimal controller, it is desirable to have the outcome of an algorithm as  $v_i(x_k)$  at the end of the  $i$ th iteration, whereas  $V_i(x_k)$  becomes an internal variable.

The VI algorithm in (2.2.8)–(2.2.9) was originally given in [5] with  $V_0(\cdot) = 0$ . The iterative process is shown in Table 2.1, where each column of blocks represents

**Table 2.1** The iterative process of the VI algorithm in (2.2.8)–(2.2.9)

|   |   |   |     |
|---|---|---|-----|
| $V_0 \rightarrow v_0$ (2.2.8)<br>minimization | $V_1 \rightarrow v_1$ (2.2.8)<br>minimization | $V_2 \rightarrow v_2$ (2.2.8)<br>minimization | ... |
| $v_0 \rightarrow V_1$ (2.2.9)<br>calculation  | $v_1 \rightarrow V_2$ (2.2.9)<br>calculation  | $v_2 \rightarrow V_3$ (2.2.9)<br>calculation  | ... |
| $i = 0$                                       | $i = 1$                                       | $i = 2$                                       | ... |

**Fig. 2.1** The iteration flowchart of algorithm in Table 2.1**Table 2.2** The iterative process of the VI algorithm in (2.2.10)–(2.2.12)

|  |  |  |     |
|--|--|--|-----|
| (empty)  | $v_0 \rightarrow V_1$ (2.2.11)<br>calculation  | $v_1 \rightarrow V_2$ (2.2.11)<br>calculation  | ... |
| $V_0 \rightarrow v_0$ (2.2.10)<br>minimization | $V_1 \rightarrow v_1$ (2.2.12)<br>minimization | $V_2 \rightarrow v_2$ (2.2.12)<br>minimization | ... |
| $i = 0$  | $i = 1$  | $i = 2$  | ... |

an iteration. The iteration goes from top to bottom within each column and from the bottom block to the top block in the next column, as shown in Fig. 2.1.

Similarly, the ADP algorithm in (2.2.10)–(2.2.12) can be described by Table 2.2. Comparing between the two tables, one can see that they contain exactly the same contents of iterations, except the fact that Table 2.2 did not start in the very first block.

Note that  $i$  is the iteration index and  $k$  is the time index. As a VI algorithm, this iterative ADP algorithm does not require an initial stabilizing controller. The value function and control law are updated until they converge to the optimal ones. Furthermore, it should satisfy that  $V_i(0) = 0$ ,  $v_i(0) = 0$ ,  $\forall i \geq 0$ .

It should be mentioned that the initial value function here is chosen as  $V_0(x_k) = x_k^T P_0 x_k$  instead of  $V_0(\cdot) = 0$  as in most traditional VI algorithms [3–5, 51, 52]. In what follows, we will prove the convergence of the iterations between (2.2.11) and (2.2.12), i.e.,  $V_i \rightarrow J^*$  and  $v_i \rightarrow u^*$  as  $i \rightarrow \infty$ .

**Lemma 2.2.1** *Let  $\mu_i$  be an arbitrary control law and let  $\Lambda_i$  be obtained by*

$$\Lambda_{i+1}(x_k) = x_k^\top Q x_k + \mu_i^\top(x_k) R \mu_i(x_k) + \Lambda_i(f(x_k) + g(x_k) \mu_i(x_k)),$$

*for  $i = 0, 1, 2, \dots$ . Let  $V_i$  and  $v_i$  be defined in (2.2.10)–(2.2.12). If  $\Lambda_0(x_k) = V_0(x_k) = x_k^\top P_0 x_k$ , then*

$$V_i(x_k) \leq \Lambda_i(x_k), \quad \forall i.$$

The lemma can easily be proved by noting that  $V_i$  is the result of minimizing the right-hand side of (2.2.11) with respect to the control input  $u_k$ , while  $\Lambda_i$  is the result of an arbitrary control input.

**Theorem 2.2.1** *Define the value function sequence  $\{V_i(x_k)\}$  and the control law sequence  $\{v_i(x_k)\}$  as in (2.2.10)–(2.2.12) with  $V_0(x_k) = x_k^\top P_0 x_k$  in (2.2.7). If  $V_0(x_k) \geq V_1(x_k)$  holds for all  $x_k$ , the value function sequence  $\{V_i\}$  is a monotonically nonincreasing sequence, i.e.,  $V_{i+1}(x_k) \leq V_i(x_k)$ ,  $\forall x_k, \forall i \geq 0$ . If  $V_0(x_k) \leq V_1(x_k)$  holds for all  $x_k$ , the value function sequence  $\{V_i(x_k)\}$  is a monotonically nondecreasing sequence, i.e.,  $V_i(x_k) \leq V_{i+1}(x_k)$ ,  $\forall x_k, \forall i \geq 0$ .*

*Proof* First, suppose that  $V_0(x_k) \geq V_1(x_k)$  holds for any  $x_k$ . Define a new sequence  $\{\Phi_i\}$ , which is updated according to

$$\begin{cases} \Phi_1(x_k) = x_k^\top Q x_k + v_0^\top(x_k) R v_0(x_k) + \Phi_0(f(x_k) + g(x_k) v_0(x_k)), \\ \Phi_{i+1}(x_k) = x_k^\top Q x_k + v_{i-1}^\top(x_k) R v_{i-1}(x_k) + \Phi_i(f(x_k) + g(x_k) v_{i-1}(x_k)), \quad i \geq 1, \end{cases}$$

where  $\Phi_0(x_k) = V_0(x_k) = x_k^\top P_0 x_k$  and  $\{v_i\}$  are obtained by (2.2.10) and (2.2.12).

Now, we use the mathematical induction to demonstrate

$$\Phi_{i+1}(x_k) \leq V_i(x_k), \quad \forall i \geq 0.$$

Noticing  $\Phi_1(x_k) = V_1(x_k)$ , it is clear that  $\Phi_1(x_k) \leq V_0(x_k)$ . Then, we assume that it holds for  $i - 1$ , i.e.,  $\Phi_i(x_k) \leq V_{i-1}(x_k)$ ,  $\forall i \geq 1, \forall x_k$ . According to

$$V_i(x_k) = x_k^\top Q x_k + v_{i-1}^\top(x_k) R v_{i-1}(x_k) + V_{i-1}(x_{k+1}), \quad i \geq 1,$$

and

$$\Phi_{i+1}(x_k) = x_k^\top Q x_k + v_{i-1}^\top(x_k) R v_{i-1}(x_k) + \Phi_i(x_{k+1}), \quad i \geq 1,$$

we have

$$V_i(x_k) - \Phi_{i+1}(x_k) = V_{i-1}(x_{k+1}) - \Phi_i(x_{k+1}) \geq 0, \quad i \geq 1,$$

which implies  $\Phi_{i+1}(x_k) \leq V_i(x_k)$ ,  $i \geq 1$ . Considering  $\Phi_1(x_k) \leq V_0(x_k)$ , we have  $\Phi_{i+1}(x_k) \leq V_i(x_k)$ ,  $i \geq 0$ . According to Lemma 2.2.1, it is clear that  $V_{i+1}(x_k) \leq \Phi_{i+1}(x_k)$ ,  $\forall i \geq 0$ . Therefore,

$$V_{i+1}(x_k) \leq V_i(x_k), \quad \forall i \geq 0, \forall x_k.$$

Thus, we complete the first part of the proof by mathematical induction.

Next, suppose that  $V_0(x_k) \leq V_1(x_k)$  holds for any  $x_k$ . Define a new sequence  $\{\Gamma_i\}$ , which is updated according to

$$\Gamma_{i+1}(x_k) = x_k^\top Q x_k + v_{i+1}^\top(x_k) R v_{i+1}(x_k) + \Gamma_i(x_{k+1}), \quad i \geq 0,$$

with  $\Gamma_0(x_k) = V_0(x_k) = x_k^\top P_0 x_k$ .

Similarly, we use the mathematical induction to demonstrate

$$\Gamma_i(x_k) \leq V_{i+1}(x_k), \quad \forall i \geq 0.$$

First, it is easy to see  $\Gamma_0(x_k) = V_0(x_k) \leq V_1(x_k)$ . Then, we assume that it holds for  $i - 1$ , i.e.,  $\Gamma_{i-1}(x_k) \leq V_i(x_k)$ ,  $\forall i \geq 1, \forall x_k$ .

According to

$$\Gamma_i(x_k) = x_k^\top Q x_k + v_i^\top(x_k) R v_i(x_k) + \Gamma_{i-1}(x_{k+1}), \quad i \geq 1,$$

and

$$V_{i+1}(x_k) = x_k^\top Q x_k + v_i^\top(x_k) R v_i(x_k) + V_i(x_{k+1}), \quad i \geq 1,$$

we have

$$V_{i+1}(x_k) - \Gamma_i(x_k) = V_i(x_{k+1}) - \Gamma_{i-1}(x_{k+1}) \geq 0, \quad i \geq 1,$$

which implies  $\Gamma_i(x_k) \leq V_{i+1}(x_k)$ ,  $i \geq 1$ . Considering  $\Gamma_0(x_k) \leq V_1(x_k)$ , we have  $\Gamma_i(x_k) \leq V_{i+1}(x_k)$ ,  $i \geq 0$ . According to Lemma 2.2.1, it is easy to find  $V_i(x_k) \leq \Gamma_i(x_k)$ ,  $\forall i \geq 0$ . Therefore,

$$V_i(x_k) \leq V_{i+1}(x_k), \quad \forall i \geq 0, \forall x_k.$$

Thus, we complete the second part of the proof by mathematical induction.

*Remark 2.2.1* From Theorem 2.2.1, we can see that the monotonicity property of the value function  $V_i$  is determined by the relationship between  $V_0$  and  $V_1$ , i.e.,  $V_0(x_k) \geq V_1(x_k)$  or  $V_0(x_k) \leq V_1(x_k)$ ,  $\forall x_k$ . In the traditional VI algorithm, the initial value function is selected as  $V_0(\cdot) = 0$ . We can easily find that this is just a special case of our general scheme, i.e.,  $V_0(x_k) \leq V_1(x_k)$ , which leads to a nondecreasing value function sequence. Furthermore, the monotonicity property is still valid starting from  $p$  if we can find that  $V_p(x_k) \geq V_{p+1}(x_k)$  or  $V_p(x_k) \leq V_{p+1}(x_k)$  for all  $x_k$  and some  $p$ . For example,

$$V_p(x_k) \geq V_{p+1}(x_k) \text{ for all } x_k \text{ and some } p \geq 0 \Rightarrow V_i(x_k) \geq V_{i+1}(x_k), \forall x_k, \forall i \geq p.$$

Next, we will demonstrate the uniform convergence of value function using the technique of [27, 35], and we will show that the control sequence converges to the



optimal control law by a corollary. The following theorem is due to Rantzer and his coworkers [27, 35].

**Theorem 2.2.2** *Suppose the condition*

$$0 \leq J^*(f(x_k) + g(x_k)u_k) \leq \gamma U(x_k, u_k)$$

*holds uniformly for some  $0 < \gamma < \infty$  and that  $0 \leq \alpha J^* \leq V_0 \leq \beta J^*$ ,  $0 \leq \alpha \leq 1$ , and  $1 \leq \beta < \infty$ . The value function sequence  $\{V_i\}$  and the control law sequence  $\{v_i\}$  are iteratively updated by (2.2.10)–(2.2.12). Then, the value function  $V_i$  approaches  $J^*$  according to the following inequalities:*

$$\left[1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^i}\right] J^*(x_k) \leq V_i(x_k) \leq \left[1 + \frac{\beta - 1}{(1 + \gamma^{-1})^i}\right] J^*(x_k). \quad (2.2.13)$$

*Moreover, the value function  $V_i(x_k)$  converges to  $J^*(x_k)$  uniformly on  $\Omega$ .*

*Proof* First, we demonstrate that the system defined in this section satisfies the conditions of Theorem 2.2.2. According to Assumption 2.2.2, the system state cannot jump to infinity by any one step of finite control input, i.e.,  $f(x_k) + g(x_k)u_k$  is finite. Because  $U(x_k, u_k)$  is a positive-definite function, there exists some  $0 < \gamma < \infty$  such that  $0 \leq J^*(f(x_k) + g(x_k)u_k) \leq \gamma U(x_k, u_k)$  holds uniformly. For any finite positive-definite initial value function  $V_0$ , there exist  $\alpha$  and  $\beta$  such that  $0 \leq \alpha J^* \leq V_0 \leq \beta J^*$  is satisfied, where  $0 \leq \alpha \leq 1$  and  $1 \leq \beta < \infty$ . Next, we will demonstrate the lower bound of the inequality (2.2.13) by mathematical induction, i.e.,

$$\left[1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^i}\right] J^*(x_k) \leq V_i(x_k). \quad (2.2.14)$$

When  $i = 1$ , since

$$\frac{\alpha - 1}{1 + \gamma} (\gamma U(x_k, u_k) - J^*(x_{k+1})) \leq 0, \quad 0 \leq \alpha \leq 1,$$

and  $\alpha J^* \leq V_0$ ,  $\forall x_k$ , we have

$$\begin{aligned} V_1(x_k) &= \min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\} \\ &\geq \min_{u_k} \{U(x_k, u_k) + \alpha J^*(x_{k+1})\} \\ &\geq \min_{u_k} \left\{ \left(1 + \gamma \frac{\alpha - 1}{1 + \gamma}\right) U(x_k, u_k) + \left(\alpha - \frac{\alpha - 1}{1 + \gamma}\right) J^*(x_{k+1}) \right\} \\ &= \left[1 + \frac{\alpha - 1}{(1 + \gamma^{-1})}\right] \min_{u_k} \{U(x_k, u_k) + J^*(x_{k+1})\} \\ &= \left[1 + \frac{\alpha - 1}{(1 + \gamma^{-1})}\right] J^*(x_k). \end{aligned}$$

$$\geq \min_{u_k} \left\{ U(x_k, u_k) + \left[ 1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^{i-1}} \right] J^*(x_{k+1}) \right\}$$

Now, assume that the inequality (2.2.14) holds for  $i - 1$ . Then, we have

$$\begin{aligned} V_i(x_k) &= \min_{u_k} \{ U(x_k, u_k) + V_{i-1}(x_{k+1}) \} \\ &\geq \min_{u_k} \left\{ U(x_k, u_k) + \left[ 1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^{i-1}} \right] J^*(x_{k+1}) \right\} \\ &\geq \min_{u_k} \left\{ \left[ 1 + \frac{(\alpha - 1)\gamma^i}{(\gamma + 1)^i} \right] U(x_k, u_k) \right. \\ &\quad \left. + \left[ 1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^{i-1}} - \frac{(\alpha - 1)\gamma^{i-1}}{(\gamma + 1)^i} \right] J^*(x_{k+1}) \right\} \\ &= \left[ 1 + \frac{(\alpha - 1)\gamma^i}{(\gamma + 1)^i} \right] \min_{u_k} \{ U(x_k, u_k) + J^*(x_{k+1}) \} \\ &= \left[ 1 + \frac{(\alpha - 1)}{(1 + \gamma^{-1})^i} \right] J^*(x_k). \end{aligned}$$

Thus, the lower bound of (2.2.13) is proved. The upper bound of (2.2.13) can be shown by the same procedure.

Lastly, we demonstrate the uniform convergence of value function as the iteration index  $i$  goes to  $\infty$ . When  $i \rightarrow \infty$ , for  $0 < \gamma < \infty$ , we have

$$\lim_{i \rightarrow \infty} \left[ 1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^i} \right] J^*(x_k) = J^*(x_k),$$

and

$$\lim_{i \rightarrow \infty} \left[ 1 + \frac{\beta - 1}{(1 + \gamma^{-1})^i} \right] J^*(x_k) = J^*(x_k).$$

Define  $V_\infty(x_k) = \lim_{i \rightarrow \infty} V_i(x_k)$ . Then, we can get  $V_\infty(x_k) = J^*(x_k)$ . Hence,  $V_i(x_k)$  converges pointwise to  $J^*(x_k)$ . Because  $\Omega$  is compact, we can get the uniform convergence of value function immediately from Dini's theorem [6]. The proof is complete.

From Theorem 2.2.2, we can determine the upper and lower bounds for every iterative value function. As the iteration index  $i$  increases, the upper bound will exponentially approach the lower bound. When the iteration index  $i$  goes to  $\infty$ , the upper bound will be equal to the lower bound, which is just the optimal cost. Additionally, we can also analyze the convergence speed of the value function, which is not available using the approaches in [3–5, 24, 51, 52]. According to the inequality (2.2.13), smaller  $\gamma$  will lead to faster convergence speed of the value function. Moreover, it should be mentioned that conditions of Theorem 2.2.2 can be satisfied according to Assumptions 2.2.1–2.2.3, which are mild for general control problems.

Specially, when  $0 \leq V_0(x_k) \leq V_1(x_k)$ ,  $\forall x_k$ , according to Theorems 2.2.1 and 2.2.2, we can deduce that  $V_0(x_k) \leq J^*(x_k)$ . Thus, the constants  $\alpha$  and  $\beta$  satisfy  $0 < \alpha \leq 1$  and  $\beta = 1$ . Then, the corresponding inequality becomes

$$\left[1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^i}\right] J^*(x_k) \leq V_i(x_k) \leq J^*(x_k).$$

Note that larger  $\alpha$  will lead to faster convergence speed of the value function.

When  $V_0(x_k) \geq V_1(x_k)$ ,  $\forall x_k$ , according to Theorems 2.2.1 and 2.2.2, we can deduce that  $V_0(x_k) \geq J^*(x_k)$ . So, the constants  $\alpha$  and  $\beta$  satisfy  $\alpha = 1$  and  $\beta \geq 1$ . Then, the corresponding inequality becomes

$$J^*(x_k) \leq V_i(x_k) \leq \left[1 + \frac{\beta - 1}{(1 + \gamma^{-1})^i}\right] J^*(x_k).$$

Note that smaller  $\beta$  will lead to faster convergence speed of the value function.

According to the results of Theorem 2.2.2, we can derive the following corollary.

**Corollary 2.2.1** *Define the value function sequence  $\{V_i\}$  and the control law sequence  $\{v_i\}$  as in (2.2.10)–(2.2.12) with  $V_0(x_k) = x_k^\top P_0 x_k$ . If the system state  $x_k$  is controllable, then the control sequence  $\{v_i\}$  converges to the optimal control law  $u^*$  as  $i \rightarrow \infty$ , i.e.,  $\lim_{i \rightarrow \infty} v_i(x_k) = u^*(x_k)$ .*

*Proof* According to Theorem 2.2.2, we have proved that  $\lim_{i \rightarrow \infty} V_i(x_k) = V_\infty(x_k) = J^*(x_k)$ . Thus,

$$V_\infty(x_k) = \min_{u_k} \{x_k^\top Q x_k + u_k^\top R u_k + V_\infty(x_{k+1})\}.$$

That is to say that the value function sequence  $\{V_i\}$  converges to the optimal value function of the Bellman equation. Comparing (2.2.5)–(2.2.12), the corresponding control law  $\{v_i\}$  converges to the optimal control law  $u^*$  as  $i \rightarrow \infty$ . This completes the proof of the corollary.

Next, we will complete the stability analysis for nonlinear systems under the condition of control Lyapunov function.

**Theorem 2.2.3** *The value function sequence  $\{V_i\}$  and the control law sequence  $\{v_i\}$  are iteratively updated by (2.2.10)–(2.2.12). If  $V_0(x_k) = x_k^\top P_0 x_k \geq V_1(x_k)$  holds for any controllable  $x_k$ , then the value function  $V_i(x_k)$  is a Lyapunov function and the system using the control law  $v_i(x_k)$  is asymptotically stable.*

*Proof* First, according to  $V_0(x_k) \geq V_1(x_k)$  and Theorem 2.2.1, we have

$$V_i(x_k) \geq V_{i+1}(x_k) \geq U(x_k, v_i(x_k)), \forall i.$$

Because  $U(x_k, v_i(x_k))$  is a positive-definite function and  $V_i(0) = 0$ ,  $V_i(x_k)$  is also a positive-definite function.

Second, we have

$$V_i(x_{k+1}) - V_i(x_k) \leq V_i(x_{k+1}) - V_{i+1}(x_k) = -U(x_k, v_i(x_k)) \leq 0.$$

By the Lyapunov stability criteria (Lyapunov's extension theorem [26] or the Lagrange stability result [30]),  $V_i(x_k)$  is a Lyapunov function, and the system using the control law  $v_i(x_k)$  is asymptotically stable. This completes the proof of the theorem.

Note that  $v_0(x_k)$  satisfies the first-order necessary condition, which is given by the gradient of the right-hand side of (2.2.10) with respect to  $u_k$  as

$$\frac{\partial (x_k^T Q x_k + u_k^T R u_k)}{\partial u_k} + \left( \frac{\partial x_{k+1}}{\partial u_k} \right)^T \frac{\partial V_0(x_{k+1})}{\partial x_{k+1}} = 0.$$

That is,

$$2R u_k + 2g^T(x_k)P_0(f(x_k) + g(x_k)u_k) = 0.$$

Then, we can solve for  $v_0(x_k)$  as

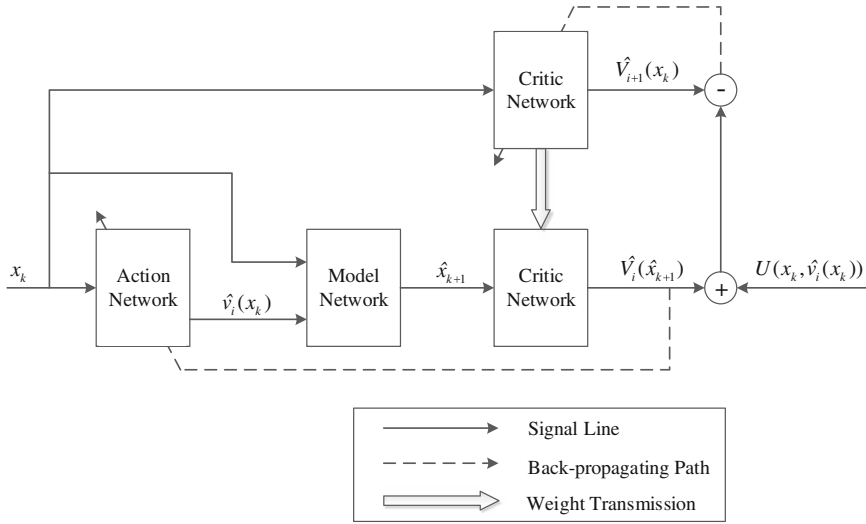
$$v_0(x_k) = -(g^T(x_k)P_0g(x_k) + R)^{-1}g^T(x_k)P_0f(x_k).$$

The control law  $v_0(x_k)$  exists since  $P_0$  and  $R$  are both positive-definite matrices.

*Remark 2.2.2* If the condition  $V_0(x_k) \geq V_1(x_k)$  holds,  $V_0(x_k) = x_k^T P_0 x_k$  is called control Lyapunov function if the associated feedback control law  $v_0(x_k)$  can guarantee the closed-loop system to be stable. Compared to PI algorithms, this condition  $V_0(x_k) \geq V_1(x_k)$  is easier to satisfy than an initial stabilizing control law. In particular, we can just choose  $P_0 = \kappa I_n$  and  $\kappa \geq 0$ , where  $I_n$  is the  $n \times n$  identity matrix. By choosing a large  $\kappa$ ,  $V_0(x_k) \geq V_1(x_k)$  is satisfied. Besides, similar to [12, 49], it should be mentioned that the condition  $V_0(x_k) \geq V_1(x_k)$  in Theorem 2.2.3 cannot be replaced by  $V_0(x_k) \geq J^*(x_k)$ , because the nonincreasing property of value function is guaranteed by  $V_0(x_k) \geq V_1(x_k)$ . However, if the condition  $V_0(x_k) \leq V_1(x_k)$  holds, we cannot derive that  $v_i(x_k)$  is a stable and admissible control for nonlinear systems. For linear discrete-time-invariant systems, Primbs and Nevistic [33] demonstrated that there exists a finite iteration index  $i^*$  and that the closed-loop system is asymptotically stable for all  $i \geq i^*$ .

## 2.2.2 Neural Network Implementation

We have demonstrated the convergence of value function in the above under the assumption that control laws and value functions can exactly be solved at each iteration. However, it is difficult to solve these equations for nonlinear systems.



**Fig. 2.2** The structural diagram of HDP algorithm

Fortunately, we can use NN to approximate  $v_i$  and  $V_i$  at each iteration. In this section, we will use heuristic dynamic programming (HDP, see definition in Chap. 1) to implement the GVI algorithm.

The structural diagram of HDP algorithm is given in Fig. 2.2. In the HDP algorithm, there are three NNs, which are model network, critic network, and action network, respectively. The model network is used to approximate the unknown nonlinear system by using available input–output data. The critic network approximates the relationship between state vector  $x_k$  and value function  $\hat{V}_i(x_k)$ , and the action network approximates the relationship between state vector  $x_k$  and control vector  $\hat{v}_i(x_k)$ .

We choose the popular backpropagation (BP) NN as our function approximation scheme, although any other function approximation structures would also suffice. The Levenberg–Marquardt (LM) algorithm is used to tune weights of NN, even though any standard NN training methods would suffice, including the gradient descent method. We find that LM algorithm can enormously improve the convergence speed and decrease the approximation error, which will lead ADP to better performance. LM algorithm, which combines steepest descent gradient and Gauss–Newton method, mainly includes three processes: calculating the Jacobian matrix, evaluating whether the parameters are getting closer to optimal ones or not, and updating the damping parameter. The details of LM algorithm used here can be found in [16].

The first step is to train the model network. The output of model network is denoted as

$$\hat{x}_{k+1} = W_m^T \sigma(\bar{\chi}_k) = W_m^T \sigma(Y_m^T \chi_k),$$

where  $\chi_k = [x_k^\top, \hat{v}_i^\top(x_k)]^\top$  is the input vector of model network and  $\bar{\chi}_k = Y_m^\top \chi_k$ . The input-to-hidden-layer weights  $Y_m$  are an  $(n + m) \times l$  matrix and the hidden-to-output-layer weights  $W_m$  are an  $l \times n$  matrix, where  $l$  is the number of hidden neurons,  $n$  is the dimension of state vector, and  $m$  is the dimension of control input vector. The activation function is chosen as  $\sigma(z) = \tanh(z)$ , and its derivative is denoted as  $\dot{\sigma}(z) = \frac{d\sigma(z)}{dz} \in \mathbb{R}^{l \times l}$  for  $z \in \mathbb{R}^l$ .

The stopping criterion is that the performance function is within a prespecified threshold, or the training step reaches the maximum value. When the weights of model network converge, they are kept unchanged. Then, the estimated value of the control coefficient matrix  $\hat{g}(x_k)$  is given by

$$\hat{g}(x_k) = \frac{\partial(W_m^\top(k)\sigma(\bar{\chi}_k))}{\partial \hat{v}_i} = W_m^\top(k)\dot{\sigma}(\bar{\chi}_k)Y_m^\top(k)\frac{\partial \chi_k}{\partial \hat{v}_i},$$

where  $\frac{\partial \chi_k}{\partial \hat{v}_i} = \begin{bmatrix} 0_{n \times m} \\ I_m \end{bmatrix}$  and  $I_m$  is the  $m \times m$  identity matrix.

Similarly, we use LM algorithm to train critic network and action network. The output of the critic network is denoted as

$$\hat{V}_i(x_k) = W_c^{(i)\top} \sigma(Y_c^{(i)\top} x_k).$$

Note that  $\hat{V}_i(x_k)$  is the estimated value function of the iterative algorithm (2.2.10)–(2.2.12) from the  $i$ th iteration, whereas  $W_c^{(i)}$  and  $Y_c^{(i)}$  are the critic NN weights to be obtained from NN training during the  $i$ th iteration. The target function for critic NN training is given by

$$V_i(x_k) = x_k^\top Q x_k + \hat{v}_{i-1}^\top(x_k) R \hat{v}_{i-1}(x_k) + \hat{V}_{i-1}(\hat{x}_{k+1}), \quad (2.2.15)$$

where  $\hat{V}_{i-1}(\hat{x}_{k+1}) = W_c^{(i-1)\top} \sigma(Y_c^{(i-1)\top} \hat{x}_{k+1})$ . Then, the error function for training critic network is defined by  $e_{c(i)}(x_k) = V_i(x_k) - \hat{V}_i(x_k)$ , and the performance function to be minimized is defined by

$$E_{c(i)}(x_k) = \frac{1}{2} e_{c(i)}^2(x_k).$$

The weight tuning algorithm of critic network is the same as model network.

In the action network, the state  $x_k$  is used as input to obtain the optimal control. The output can be formulated as  $\hat{v}_i(x_k) = W_a^{i\top} \sigma(Y_a^{i\top} x_k)$ , whereas  $W_a^i$  and  $Y_a^i$  are the action NN weights to be obtained from NN training during the  $i$ th iteration of the ADP algorithm (2.2.10)–(2.2.12). The target of action NN training is given by

$$v_i(x_k) = -\frac{1}{2} R^{-1} \hat{g}^\top(x_k) \frac{\partial \hat{V}_i(\hat{x}_{k+1})}{\partial \hat{x}_{k+1}}, \quad (2.2.16)$$

where  $\hat{x}_{k+1} = W_m^T \sigma(Y_m^T [x_k^T, \hat{v}_i^T]^T)$ . The convergence of action network weights is shown in [13]. The error function of the action network can be defined as  $e_{a(i)}(x_k) = v_i(x_k) - \hat{v}_i(x_k)$ . The weights of the action network are updated to minimize the following performance function:

$$E_{a(i)}(x_k) = \frac{1}{2} e_{a(i)}^T(x_k) e_{a(i)}(x_k).$$

The LM algorithm ensures that  $E_{a(i)}(x_k)$  will decrease every time when the parameters of action network update.

At last, a summary of the present general value iteration adaptive dynamic programming algorithm for optimal control is given in Algorithm 2.2.1.

---

**Algorithm 2.2.1** General value iteration adaptive dynamic programming algorithm

---

Step 1. Initialize the weights of critic and action neural networks and the parameters  $j_{\max}^m, j_{\max}^a, j_{\max}^c, \varepsilon_m, \varepsilon_a, \varepsilon_c, i_{\max}, \xi, Q, R$ .

Step 2. Construct the model network  $\hat{x}_{k+1} = W_m^T \sigma(Y_m^T \chi_k)$ . Obtain the training data, and train the model network until the given accuracy  $\varepsilon_m$  or the maximum number of iterations  $j_{\max}^m$  is reached.

Step 3. Set the iteration index  $i = 0$  and  $P_0 = \kappa I_n$ .

Step 4. Choose randomly an array of  $p$  state vector  $\{x_k^1, x_k^2, \dots, x_k^p\}$ . Compute the output of the action network  $\{\hat{v}_i(x_k^1), \hat{v}_i(x_k^2), \dots, \hat{v}_i(x_k^p)\}$ . Compute the output of the model network  $\{\hat{x}_{k+1}^1, \hat{x}_{k+1}^2, \dots, \hat{x}_{k+1}^p\}$  and the output of the critic network  $\{\hat{V}_i(\hat{x}_{k+1}^1), \hat{V}_i(\hat{x}_{k+1}^2), \dots, \hat{V}_i(\hat{x}_{k+1}^p)\}$ .

Step 5. Set the iteration index  $i = i + 1$ . Then, compute the target of the critic network training

$$\{V_i(x_k^1), V_i(x_k^2), \dots, V_i(x_k^p)\}$$

by (2.2.15). Train the critic network until the given accuracy  $\varepsilon_c$  or the maximum number of iterations  $j_{\max}^c$  is reached.

Step 6. If  $i > 1$ , then go to Step 7. Elseif  $V_0 > V_1$  is true for all  $x_k$ , go to Step 7; otherwise, increase  $\kappa$  and go to Step 3.

Step 7. Compute the target of action network training

$$\{v_i(x_k^1), v_i(x_k^2), \dots, v_i(x_k^p)\}$$

by (2.2.16), and train the action network until the given accuracy  $\varepsilon_a$  or the maximum number of iterations  $j_{\max}^a$  is reached.

Step 8. If  $i > i_{\max}$  or

$$|V_i(x_k^s) - V_{i-1}(x_k^s)| \leq \xi, \quad s = 1, 2, \dots, p,$$

go to Step 9; otherwise, go to Step 4.

Step 9. Compute the output of the action network  $\{\hat{v}_i(x_k^1), \hat{v}_i(x_k^2), \dots, \hat{v}_i(x_k^p)\}$ . Obtain the final near optimal control law

$$u^*(\cdot) = \hat{v}_i(\cdot),$$

and stop the algorithm.

---

### 2.2.3 Generalization to Optimal Tracking Control

The above GVI-based ADP approach can be employed to solve the optimal tracking control problem [45]. Consider the nonaffine nonlinear system (2.2.1), for infinite-time optimal tracking problem, the objective is to design an optimal control  $u^*(x_k)$ , such that the state  $x_k$  tracks the specified desired trajectory  $\xi_k \in \mathbb{R}^n$ ,  $k = 0, 1, \dots$ . In this section, we assume that there exists a feedback control  $u_{e,k}$ , which satisfies the following equation:

$$\xi_{k+1} = F(\xi_k, u_{e,k}), \quad (2.2.17)$$

where  $u_{e,k}$  is called the desired control.

*Remark 2.2.3* It should be pointed out that for a large class of nonlinear systems, there exists a feedback control  $u_{e,k}$  that satisfies (2.2.17). For example, for all the affine nonlinear systems (2.2.2) with invertible  $g(x_k)$ , the desired control  $u_{e,k}$  can be expressed as

$$u_{e,k} = g^{-1}(\xi_k)(\xi_{k+1} - f(\xi_k)),$$

where  $g(\xi_k)g^{-1}(\xi_k) = I_m$  and  $I_m$  is the  $m \times m$  identity matrix.

Define the tracking error as  $z_k = \xi_k - x_k$ . The utility function is quadratic and is given by

$$U(z_k, \mu_k) = z_k^\top Q z_k + \mu_k^\top R \mu_k,$$

where  $\mu_k = u_k - u_{e,k}$  and  $u_{e,k}$  is the desired control that satisfies (2.2.17). The quadratic cost function is

$$J(z_0, \underline{\mu}_0) = \sum_{k=0}^{\infty} U(z_k, \mu_k) = \sum_{k=0}^{\infty} \{z_k^\top Q z_k + (u_k - u_{e,k})^\top R (u_k - u_{e,k})\}, \quad (2.2.18)$$

where  $\underline{\mu}_0 = (\mu_0, \mu_1, \dots)$ .

For system (2.2.1), our goal is to find an optimal tracking control scheme which tracks the desired trajectory  $\xi_k$  and simultaneously minimizes the cost function (2.2.18). The optimal cost function is defined as

$$J^*(z_k) = \inf_{\underline{\mu}_k} \{J(z_k, \underline{\mu}_k)\},$$

where  $\underline{\mu}_k = (\mu_k, \mu_{k+1}, \dots)$ . According to Bellman's principle of optimality,  $J^*(z_k)$  satisfies the Bellman equation

$$\begin{aligned} J^*(z_k) &= \min_{\mu_k} \{U(z_k, \mu_k) + J^*(z_{k+1})\} \\ &= \min_{\mu_k} \{U(z_k, \mu_k) + J^*(F(z_k, \mu_k))\}. \end{aligned} \quad (2.2.19)$$



Then, the optimal control law is expressed as

$$\mu^*(z_k) = \arg \min_{\mu_k} \{U(z_k, \mu_k) + J^*(F(z_k, \mu_k))\}.$$

Hence, the Bellman equation (2.2.19) can be written as

$$J^*(z_k) = U(z_k, \mu^*(z_k)) + J^*(F(z_k, \mu^*(z_k))). \quad (2.2.20)$$

Generally speaking,  $J^*(z_k)$  is a high nonlinear and nonanalytic function, which cannot be obtained by directly solving the Bellman equation (2.2.20). Similar to Sect. 2.2.1, a GVI-based ADP method can be developed to obtain  $J^*(z_k)$  iteratively. Then, the optimal tracking control can be obtained.

Let  $\Psi(z_k)$  be an arbitrary positive-semidefinite function for  $z_k \in \mathbb{R}^n$ . Then, let the initial value function be

$$V_0(z_k) = \Psi(z_k). \quad (2.2.21)$$

The control law  $v_0(z_k)$  can be computed as follows:

$$\begin{aligned} v_0(z_k) &= \arg \min_{\mu_k} \{U(z_k, \mu_k) + V_0(z_{k+1})\} \\ &= \arg \min_{\mu_k} \{U(z_k, \mu_k) + V_0(F(z_k, \mu_k))\}, \end{aligned} \quad (2.2.22)$$

where  $V_0(z_{k+1}) = \Psi(z_{k+1})$ . For  $i = 1, 2, \dots$ , the iterative ADP algorithm will iterate between value function update

$$\begin{aligned} V_i(z_k) &= \min_{\mu_k} \{U(z_k, \mu_k) + V_{i-1}(z_{k+1})\} \\ &= U(z_k, v_{i-1}(z_k)) + V_{i-1}(F(z_k, v_{i-1}(z_k))), \end{aligned} \quad (2.2.23)$$

and policy improvement

$$v_i(z_k) = \arg \min_{\mu_k} \{U(z_k, \mu_k) + V_i(z_{k+1})\} = \arg \min_{\mu_k} \{U(z_k, \mu_k) + V_i(F(z_k, \mu_k))\}. \quad (2.2.24)$$

Note that the ADP algorithm described above in (2.2.22)–(2.2.24) (for nonaffine nonlinear systems) is essentially the same as that in (2.2.10)–(2.2.12) (for affine nonlinear systems). The only difference between the two is the choice of initial value function (see (2.2.7) and (2.2.21)).

Additional properties of the GVI-based ADP algorithm are given as follows.

**Theorem 2.2.4** For  $i = 0, 1, \dots$ , let  $V_i(z_k)$  and  $v_i(z_k)$  be obtained by (2.2.21)–(2.2.24). Let  $\rho$ ,  $\gamma$ ,  $\alpha$ , and  $\beta$  be constants such that

$$0 < \rho \leq \gamma < \infty, \quad (2.2.25)$$

and  $0 \leq \alpha \leq \beta < 1$ , respectively. If  $\forall z_k$ , the following conditions

$$\rho U(z_k, v_k) \leq J^*(F(z_k, v_k)) \leq \gamma U(z_k, v_k) \quad (2.2.26)$$

and

$$\alpha J^*(z_k) \leq V_0(z_k) \leq \beta J^*(z_k) \quad (2.2.27)$$

are satisfied uniformly, then the iterative value function  $V_i(z_k)$  satisfies

$$\left(1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^i}\right) J^*(z_k) \leq V_i(z_k) \leq \left(1 + \frac{\beta - 1}{(1 + \rho^{-1})^i}\right) J^*(z_k). \quad (2.2.28)$$

*Proof* The theorem can be proved in two steps.

(1) Prove the lower bound of (2.2.28).

Mathematical induction is employed to prove the conclusion. Let  $i = 1$ . From (2.2.26) and (2.2.27), we have

$$\begin{aligned} V_1(z_k) &= \min_{v_k} \{U(z_k, v_k) + V_0(z_{k+1})\} \\ &\geq \min_{v_k} \{U(z_k, v_k) + \alpha J^*(z_{k+1})\} \\ &\geq \min_{v_k} \left\{ \left(1 + \gamma \frac{\alpha - 1}{1 + \gamma}\right) U(z_k, v_k) + \left(\alpha - \frac{\alpha - 1}{1 + \gamma}\right) J^*(z_{k+1}) \right\} \\ &= \left(1 + \frac{\alpha - 1}{1 + \gamma^{-1}}\right) \min_{v_k} \{U(z_k, v_k) + J^*(z_{k+1})\} \\ &= \left(1 + \frac{\alpha - 1}{1 + \gamma^{-1}}\right) J^*(z_k). \end{aligned}$$

Assume the conclusion holds for  $i = l - 1$ ,  $l = 1, 2, \dots$ . Then, for  $i = l$ , we have

$$\begin{aligned} V_l(z_k) &= \min_{v_k} \{U(z_k, v_k) + V_{l-1}(z_{k+1})\} \\ &\geq \min_{v_k} \left\{ U(z_k, v_k) + \left(1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^{l-1}}\right) J^*(z_{k+1}) \right\} \\ &\geq \min_{v_k} \left\{ \left(1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^l}\right) U(z_k, v_k) \right. \\ &\quad \left. + \left(1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^{l-1}} - \gamma^{-1} \frac{\alpha - 1}{(1 + \gamma^{-1})^l}\right) J^*(z_{k+1}) \right\} \end{aligned}$$

$$\begin{aligned}
&= \left(1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^l}\right) \min_{v_k} \{U(z_k, v_k) + J^*(z_{k+1})\} \\
&= \left(1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^l}\right) J^*(z_k).
\end{aligned}$$

(2) Prove the upper bound of (2.2.28).

We also use mathematical induction to prove the conclusion. Let  $i = 1$ . We have

$$\begin{aligned}
V_1(z_k) &= \min_{v_k} \{U(z_k, v_k) + V_0(z_{k+1})\} \\
&\leq \min_{v_k} \{U(z_k, v_k) + \beta J^*(z_{k+1})\} \\
&\leq \min_{v_k} \left\{ U(z_k, v_k) + \beta J^*(z_{k+1}) + \frac{\beta - 1}{(1 + \rho)} (J^*(z_{k+1}) - \rho U(z_k, v_k)) \right\} \\
&= \left(1 + \frac{\beta - 1}{1 + \rho^{-1}}\right) \min_{v_k} \{U(z_k, v_k) + J^*(z_{k+1})\} \\
&= \left(1 + \frac{\beta - 1}{1 + \rho^{-1}}\right) J^*(z_k).
\end{aligned}$$

Assume that the conclusion holds for  $i = l - 1$ ,  $l = 1, 2, \dots$ . Then, for  $i = l$ , we have

$$\begin{aligned}
V_l(z_k) &= \min_{v_k} \{U(z_k, v_k) + V_{l-1}(z_{k+1})\} \\
&\leq \min_{v_k} \left\{ U(z_k, v_k) + \left(1 + \frac{\beta - 1}{(1 + \rho^{-1})^{l-1}}\right) J^*(z_{k+1}) \right\} \\
&\leq \left(1 + \frac{\beta - 1}{(1 + \rho^{-1})^l}\right) \min_{v_k} \{U(z_k, v_k) + J^*(z_{k+1})\} \\
&= \left(1 + \frac{\beta - 1}{(1 + \rho^{-1})^l}\right) J^*(z_k).
\end{aligned}$$

The proof is complete.

The following two results can readily be proved by following the same procedure.

**Theorem 2.2.5** For  $i = 0, 1, \dots$ , let  $v_i(z_k)$  and  $V_i(z_k)$  be obtained by (2.2.21)–(2.2.24). Let  $\rho, \gamma, \alpha$ , and  $\beta$  be constants that satisfy (2.2.25) and

$$1 \leq \alpha \leq \beta < \infty,$$

respectively. If  $\forall z_k$ , the inequalities (2.2.26) and (2.2.27) hold uniformly, then the iterative value function  $V_i(z_k)$  satisfies (2.2.28).

**Corollary 2.2.2** For  $i = 0, 1, \dots$ , let  $v_i(z_k)$  and  $V_i(z_k)$  be obtained by (2.2.21)–(2.2.24). Let  $\rho, \gamma, \alpha$ , and  $\beta$  be constants that satisfy (2.2.25) and

$$0 \leq \alpha \leq \beta < \infty, \quad (2.2.29)$$

respectively. If  $\forall z_k$ , the inequalities (2.2.26) and (2.2.27) hold uniformly, then the iterative value function  $V_i(z_k)$  converges to the optimal cost function  $J^*(z_k)$ , i.e.,

$$\lim_{i \rightarrow \infty} V_i(z_k) = J^*(z_k).$$

*Remark 2.2.4* When  $0 \leq \alpha \leq 1 \leq \beta < \infty$ , we can also obtain result similar to Theorem 2.2.2. Corollary 2.2.2 is obtained directly from Theorems 2.2.2, 2.2.4, and 2.2.5.

*Remark 2.2.5* Note that techniques employed in this section are extensions of that in [27, 35]. From Theorem 2.2.2, we can see that the iterative value function will converge to the optimum as  $i \rightarrow \infty$ , which is independent from the initial value function  $\Psi(z_k)$ . Furthermore, for arbitrary constants  $\rho$ ,  $\gamma$ ,  $\alpha$ , and  $\beta$  that satisfy (2.2.25) and (2.2.29), respectively, the iterative value function  $V_i(z_k)$  can be guaranteed to converge to the optimum as  $i \rightarrow \infty$ . Hence, the estimations of  $\rho$ ,  $\gamma$ ,  $\alpha$ , and  $\beta$  are not necessary.

## 2.2.4 Optimal Control of Systems with Constrained Inputs

The VI-based optimal control [5, 41], constrained optimal control [28, 51], and optimal tracking control [19, 52] methods are special cases of the results in Sect. 2.2.3, by noting that the initial value function is chosen as zero. Among them, input constraints are often confronted in practical problems, which results in a considerable difficulty in designing the optimal controller [17, 28, 51]. Therefore, in this section, we develop a VI-based constrained optimal control scheme via GDHP technique [28].

Consider the discrete-time nonaffine nonlinear systems (2.2.1), we define  $\bar{\Omega}_u = \{u_k : u_k = [u_{1k}, u_{2k}, \dots, u_{mk}]^T \in \mathbb{R}^m, |u_{lk}| \leq \bar{u}_l, l = 1, 2, \dots, m\}$ , where  $\bar{u}_l$  is the saturation bound for the  $l$ th actuator. Let  $\bar{U} = \text{diag}\{\bar{u}_1, \bar{u}_2, \dots, \bar{u}_m\}$  be a constant diagonal matrix.

In many literatures of optimal control [5, 13, 41, 42], the utility function is chosen as the quadratic form of (2.2.6). However, when dealing with constrained optimal control problems, it is not the case any more. Inspired by the work of [1, 29, 51], we can employ a generalized nonquadratic functional

$$\mathcal{Y}(u_k) = 2 \int_0^{u_k} \Phi^{-T} (\bar{U}^{-1}s) \bar{U}R \, ds \quad (2.2.30)$$

to substitute the quadratic term of  $u_k$  in (2.2.6). Note that in (2.2.30),

$$\Phi^{-1}(u_k) = [\phi^{-1}(u_{1k}), \phi^{-1}(u_{2k}), \dots, \phi^{-1}(u_{mk})]^T,$$

$R$  is positive-definite and assumed to be diagonal for simplicity of analysis,  $s \in \mathbb{R}^m$ ,  $\Phi \in \mathbb{R}^m$ ,  $\Phi^{-\top}$  denotes  $(\Phi^{-1})^\top$ , and  $\phi(\cdot)$  is a strictly monotonic odd function satisfying  $|\phi(\cdot)| < 1$  and belonging to  $\mathcal{C}^p$  ( $p \geq 1$ ) and  $\mathcal{L}_2(\Omega)$ . The well-known hyperbolic tangent function  $\phi(\cdot) = \tanh(\cdot)$  is one example of such functions. Besides, it is important to note that  $\mathcal{V}(u_k)$  is positive-definite since  $\phi^{-1}(\cdot)$  is a monotonic odd function and  $R$  is positive-definite.

In this sense, the utility function becomes  $U(x_k, u_k) = x_k^\top Q x_k + \mathcal{V}(u_k)$ . Accordingly, (2.2.4) and (2.2.5) become

$$J^*(x_k) = \min_{u_k} \left\{ x_k^\top Q x_k + 2 \int_0^{u_k} \Phi^{-\top} (\bar{U}^{-1} s) \bar{U} R ds + J^*(x_{k+1}) \right\}$$

and

$$u^*(x_k) = \arg \min_{u_k} \left\{ x_k^\top Q x_k + 2 \int_0^{u_k} \Phi^{-\top} (\bar{U}^{-1} s) \bar{U} R ds + J^*(x_{k+1}) \right\},$$

respectively.

The traditional VI-based iterative ADP algorithm is performed as follows. First, we start with the initial value function  $V_0(\cdot) = 0$  and solve  $V_i(x_k)$  and  $v_i(x_k)$  using the iterative algorithm described by (2.2.22)–(2.2.24).

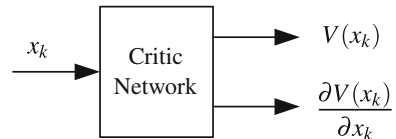
In this section, the GDHP technique is employed to implement the iterative ADP algorithm. In the iterative GDHP algorithm, there are three NNs, which are model network, critic network, and action network. Here, all the NNs are chosen as three-layer feedforward ones. It is important to note that the critic network of GDHP outputs both the value function  $V(x_k)$  and its derivative  $\partial V(x_k)/\partial x_k$  [34], which is schematically depicted in Fig. 2.3. It is a combination of HDP and dual heuristic dynamic programming (DHP).

The training of model network is complete after the system identification process, and its weights will be kept unchanged. As a result, we avoid the requirement of knowing  $F(x_k, u_k)$  during the implementation of the iterative GDHP algorithm. Next, the learned NN model will be used in the training process of critic network and action network.

We denote  $\lambda_i(x_k) = \partial V_i(x_k)/\partial x_k$  in our discussion. Hence, the critic network is used to approximate both  $V_i(x_k)$  and  $\lambda_i(x_k)$ . The output of critic network is expressed as

$$\begin{bmatrix} \hat{V}_i(x_k) \\ \hat{\lambda}_i(x_k) \end{bmatrix} = \begin{bmatrix} W_{c1}^{i\top} \\ W_{c2}^{i\top} \end{bmatrix} \sigma(Y_c^{i\top} x_k) = W_c^{i\top} \sigma(Y_c^{i\top} x_k),$$

**Fig. 2.3** The critic network of GDHP technique



where  $W_c^i = [W_{c1}^i, W_{c2}^i]$  and  $Y_c^i$  are critic NN weights to be obtained during the  $i$ th iteration of the ADP algorithm (2.2.22)–(2.2.24). Accordingly, we have  $\hat{V}_i(x_k) = W_{c1}^{i\top} \sigma(Y_c^{i\top} x_k)$  and  $\hat{\lambda}_i(x_k) = W_{c2}^{i\top} \sigma(Y_c^{i\top} x_k)$ . The target functions for critic NN training can be written as

$$V_i(x_k) = U(x_k, \hat{v}_{i-1}(x_k)) + \hat{V}_{i-1}(\hat{x}_{k+1})$$

and

$$\begin{aligned} \lambda_i(x_k) &= \frac{\partial U(x_k, \hat{v}_{i-1}(x_k))}{\partial x_k} + \frac{\partial \hat{V}_{i-1}(\hat{x}_{k+1})}{\partial x_k} \\ &= 2Qx_k + 2\left(\frac{\partial \hat{v}_{i-1}(x_k)}{\partial x_k}\right)^\top \bar{U}R\Phi^{-1}(\bar{U}^{-1}\hat{v}_{i-1}(x_k)) \\ &\quad + \left(\frac{\partial \hat{x}_{k+1}}{\partial x_k} + \frac{\partial \hat{x}_{k+1}}{\partial \hat{v}_{i-1}(x_k)} \frac{\partial \hat{v}_{i-1}(x_k)}{\partial x_k}\right)^\top \hat{\lambda}_{i-1}(\hat{x}_{k+1}). \end{aligned}$$

Then, we define the error function of critic network training as  $e_{cik}^v = V_i(x_k) - \hat{V}_i(x_k)$  and  $e_{cik}^\lambda = \lambda_i(x_k) - \hat{\lambda}_i(x_k)$ . The objective function to be minimized in the critic network is

$$E_{cik} = (1 - \tau)E_{cik}^v + \tau E_{cik}^\lambda,$$

where  $0 \leq \tau \leq 1$  is a parameter that adjusts how HDP and DHP are combined in GDHP,

$$E_{cik}^v = \frac{1}{2}(e_{cik}^v)^2$$

and

$$E_{cik}^\lambda = \frac{1}{2}e_{cik}^{\lambda\top} e_{cik}^\lambda.$$

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

$$\begin{aligned} W_c^i(p+1) &= W_c^i(p) - \alpha_c \left[ (1 - \tau) \frac{\partial E_{cik}^v}{\partial W_c^i(p)} + \tau \frac{\partial E_{cik}^\lambda}{\partial W_c^i(p)} \right], \\ Y_c^i(p+1) &= Y_c^i(p) - \alpha_c \left[ (1 - \tau) \frac{\partial E_{cik}^v}{\partial Y_c^i(p)} + \tau \frac{\partial E_{cik}^\lambda}{\partial Y_c^i(p)} \right], \end{aligned}$$

where  $\alpha_c > 0$  is the learning rate of critic network and  $p$  is the inner-loop iteration step for updating NN weight parameters. The detailed discussion on superiority of GDHP-based iterative ADP algorithm can be found in [42].

*Remark 2.2.6* The GDHP (globalized dual heuristic programming) is implemented using

$$E_{cik} = (1 - \tau)E_{cik}^v + \tau E_{cik}^\lambda.$$

When  $\tau = 0$ , the algorithm reduces to HDP (heuristic dynamic programming) where critic NN training is based on value function  $V$ . When  $\tau = 1$ , the algorithm becomes DHP (dual heuristic programming) where critic NN training is based on the derivative  $\lambda$  of the value function. In order to determine the minimum value of a function, we can either estimate the function itself or estimate its derivatives. However, when  $0 < \tau < 1$ , the algorithm will use the estimates of both the value function and its derivatives, which will usually lead to better results in optimization.

In the action network, the state  $x_k$  is used as input to obtain the approximate optimal control as output of the network, which is formulated as

$$\hat{v}_i(x_k) = W_a^{iT} \sigma(Y_a^{iT} x_k).$$

The target function of control action is given by

$$v_i(x_k) = \arg \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_i(\hat{x}_{k+1}) \right\}.$$

The error function of action network can be defined as

$$e_{a(i)k} = v_i(x_k) - \hat{v}_i(x_k).$$

The weights of action network are updated to minimize

$$E_{a(i)k} = \frac{1}{2} e_{a(i)k}^T e_{a(i)k}.$$

Similarly, the weight update algorithm is

$$\begin{aligned} W_a^i(p+1) &= W_a^i(p) - \alpha_a \left[ \frac{\partial E_{a(i)k}}{\partial W_a^i(p)} \right], \\ Y_a^i(p+1) &= Y_a^i(p) - \alpha_a \left[ \frac{\partial E_{a(i)k}}{\partial Y_a^i(p)} \right], \end{aligned}$$

where  $\alpha_a > 0$  is the learning rate of action network and  $p$  is the inner-loop iteration step for updating weight parameters.

### 2.2.5 Simulation Studies

In this section, several examples are provided to demonstrate the effectiveness of the present control methods.

*Example 2.2.1* Consider the linear system

$$x_{k+1} = \begin{bmatrix} 0 & 0.4 \\ 0.3 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k, \quad (2.2.31)$$

where  $x_k = [x_{1k}, x_{2k}]^\top$ . The weight matrices are chosen as

$$Q = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$$

and  $R = 1$ . Note that the open-loop poles are  $-0.1083$  and  $1.1083$ , which indicates that the system is unstable.

Algorithm 2.2.1 will be used here. To reduce the influence of the NN approximation errors, we choose three-layer BP NNs as model network, critic network and action network with the structures of 3–9–2, 2–8–1, and 2–8–1, respectively. The initial weights of NNs are chosen randomly in  $[-0.1, 0.1]$ .

Before implementing the GVI algorithm, we need to train the model network first. The operation region of system (2.2.31) is selected as  $-1 \leq x_1 \leq 1$  and  $-1 \leq x_2 \leq 1$ . Thousand samples are randomly chosen from this operation region as the training set, and the model network is trained until the given accuracy  $\varepsilon_m = 10^{-8}$  is reached with  $j_{\max}^m = 10000$ . The inner-loop iteration number of critic network and action network is  $j_{\max}^c = j_{\max}^a = 1000$ , and the given accuracy is  $\varepsilon_c = \varepsilon_a = 10^{-6}$ . The maximum outer-loop iteration is selected as  $i_{\max} = 10$ , and the prespecified accuracy is selected as  $\xi = 10^{-6}$ . The number of samples at each iteration is  $p = 2000$ .

Set  $P_0 = I_2$ . We find that  $V_0 \geq V_1$  holds for all states, which can be seen from Fig. 2.4. After implementing the outer-loop iteration for 10 times, the convergence of value function is observed. The 3-D plot of approximate value function at  $i = 0$  and  $i = 10$  is given in Fig. 2.5, and the 3-D plot of error between the optimal cost function  $J^*$  and the approximate optimal value function  $V_{10}$  is given in Fig. 2.6. We can see that the error between the optimal cost function and the approximate optimal value function is nearly within  $10^{-3}$  in the operational region from Fig. 2.6.

For the initial state  $x_0 = [1, -1]^\top$ , the convergence process of value function is given in Fig. 2.7. We apply the control law  $v_{10}$  to the system for 20 time steps. The corresponding state trajectories are given in Fig. 2.8, and the control input is shown in Fig. 2.9.

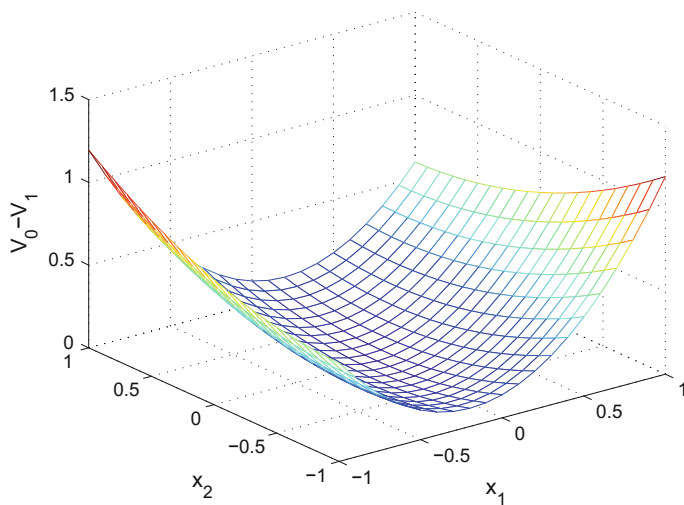
These simulation results indicate that our algorithm is effective in obtaining the optimal control law via learning in a timely manner.

*Example 2.2.2* Consider the nonlinear system

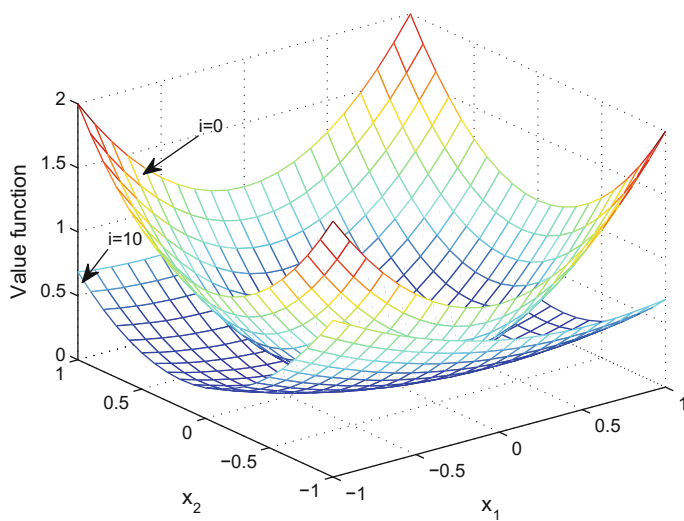
$$x_{k+1} = \begin{bmatrix} 0.2x_{1k} \exp(x_{2k}^2) \\ 0.3x_{2k}^3 \end{bmatrix} + \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix} u_k, \quad (2.2.32)$$

where  $x_k = [x_{1k}, x_{2k}]^\top$  and  $u_k = [u_{1k}, u_{2k}]^\top$ . The desired trajectory is set to

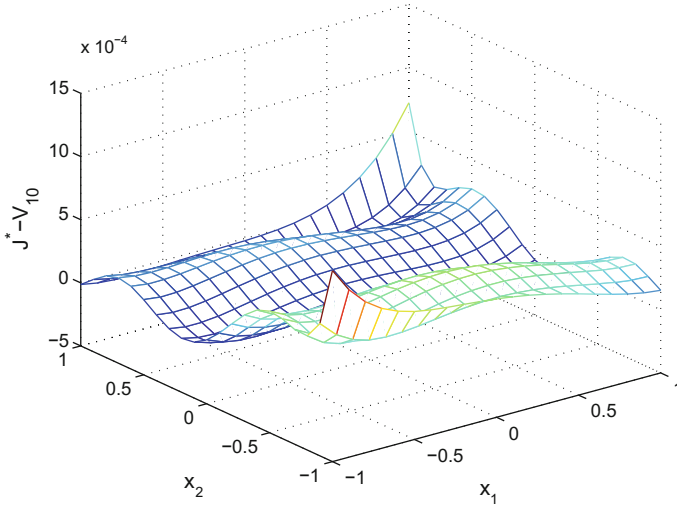




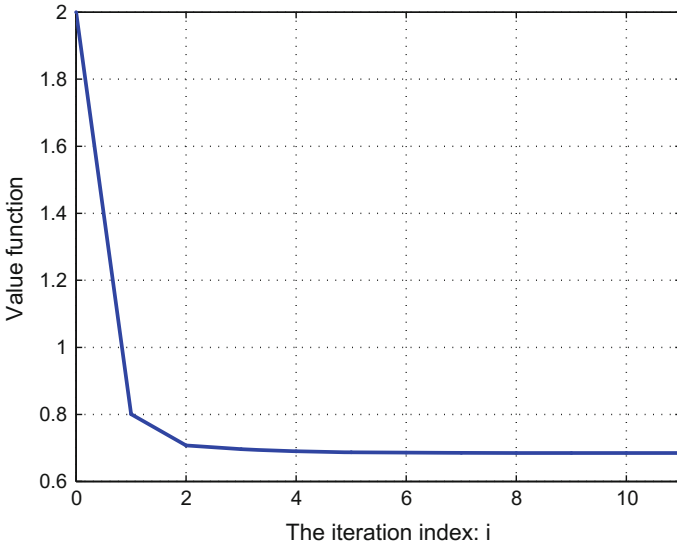
**Fig. 2.4** 3-D plot of  $V_0 - V_1$  in the operation region



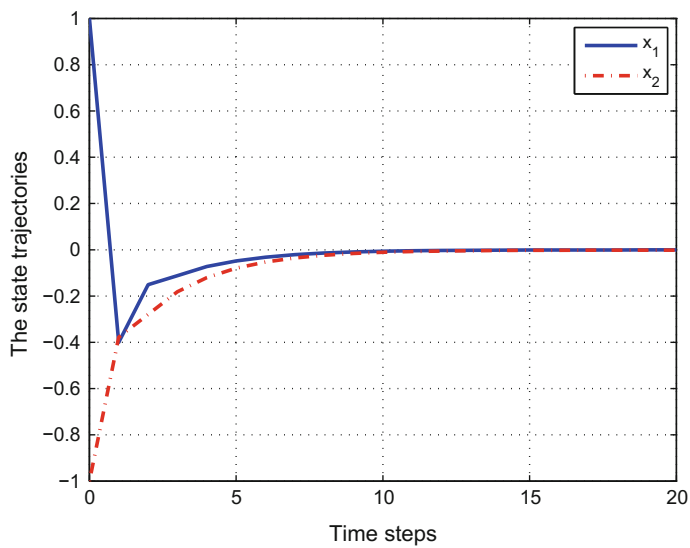
**Fig. 2.5** 3-D plot of approximate value function at  $i = 0, 10$



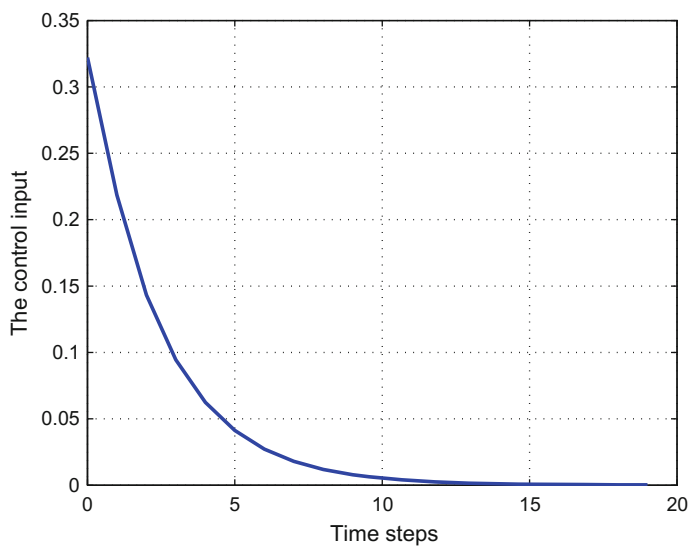
**Fig. 2.6** Error between the optimal cost function  $J^*$  and the approximate optimal value function  $V_{10}$



**Fig. 2.7** Convergence process of the value function at  $x = [1, -1]^T$



**Fig. 2.8** The state trajectories



**Fig. 2.9** The control input

$$\xi_k = \left[ \sin\left(k + \frac{\pi}{2}\right), 0.5 \cos(k) \right]^T. \quad (2.2.33)$$

According to (2.2.32) and (2.2.33), we can easily obtain the desired control

$$u_{e,k} = - \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \left( \xi_{k+1} - \begin{bmatrix} 0.2\xi_{1k} \exp(\xi_{2k}^2) \\ 0.3\xi_{2k}^3 \end{bmatrix} \right).$$

The value function is defined as in (2.2.18), where  $Q = R = I \in \mathbb{R}^{2 \times 2}$  and  $I$  denotes the identity matrix.

We use NNs to implement the GVI ADP algorithm. The structures of the critic and the action networks are chosen as 2–8–1 and 2–8–2, respectively. We choose a random array of state variable in  $[-1, 1]$  to train the NNs. For each iterative step, the critic network and the action network are trained for 2000 steps under the learning rate 0.005 so that the approximation error limit  $10^{-6}$  is reached. The GVI algorithm runs for 30 iterations to guarantee the convergence of the iterative value function. To illustrate the effectiveness of the algorithm, four different initial value functions are considered. Let the initial value functions be the quadratic form which are expressed by  $\Psi^j(z_k) = z_k^T P_j z_k$ ,  $j = 1, 2, 3, 4$ . Let  $P_1 = 0$ . Let  $P_2, P_3$ , and  $P_4$  be positive-definite matrices given by  $P_2 = [9.07, -0.26; -0.26, 11.62]$ ,  $P_3 = [10.48, 2.16; 2.16, 13.24]$ , and  $P_4 = [11.59, 0.61; 0.61, 13.40]$ , respectively.

According to Theorem 2.2.2, for an arbitrary positive-semidefinite function, the iterative value function will converge to the optimum. The curve of the iterative value functions under the four different initial value functions  $\Psi^j(z_k)$ ,  $j = 1, 2, 3, 4$ , is displayed in Fig. 2.10, which justifies the convergence property of our algorithm. The tracking error trajectories are shown in Fig. 2.11. These results show good convergence results as well as good tracking control performance.

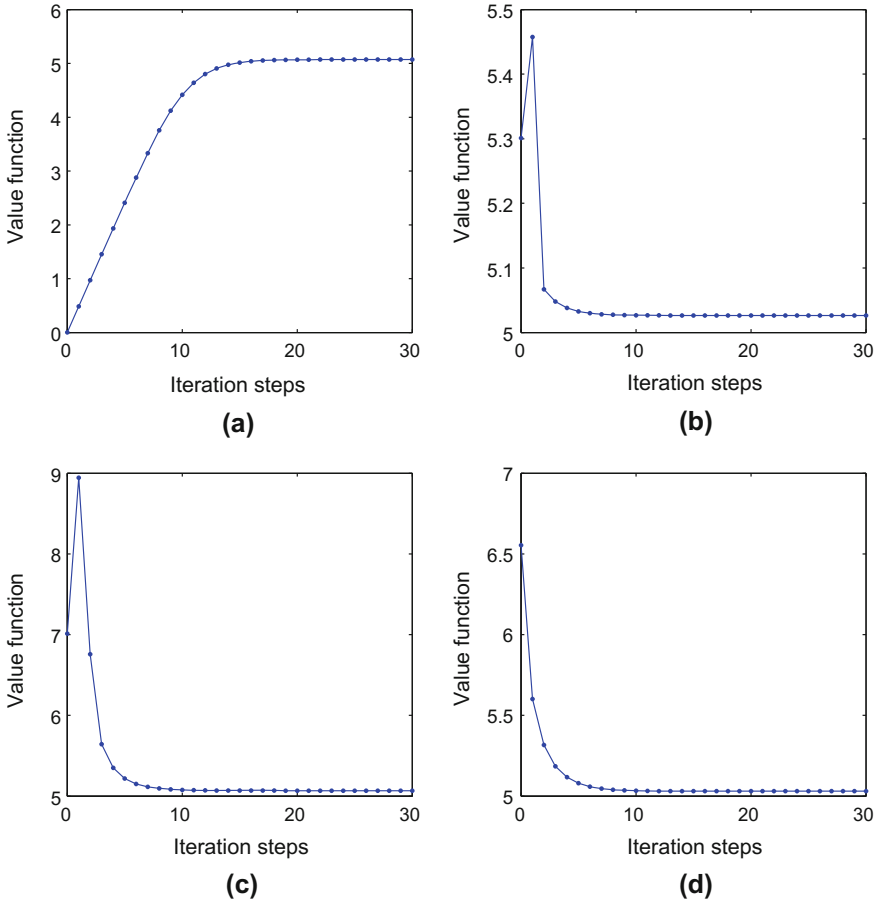
*Example 2.2.3* The following nonlinear system is a modification of the example in [21]:

$$x_{k+1} = \begin{bmatrix} x_{1k} + \sin(4u_k - 2x_{2k}) \\ x_{2k} - 2u_k \end{bmatrix}, \quad (2.2.34)$$

where  $x_k = [x_{1k}, x_{2k}]^T \in \mathbb{R}^2$ ,  $u_k \in \mathbb{R}$ ,  $k = 1, 2, \dots$ . We can see that  $x_k = [0, 0]^T$  is an equilibrium state of system (2.2.34). However, the system (2.2.34) is marginally stable at this equilibrium, since the eigenvalues of

$$\left. \frac{\partial x_{k+1}}{\partial x_k} \right|_{(0,0)} = \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}$$

are all 1. It is desired to control the system with control constraint of  $|u| \leq 0.5$ . The cost function is chosen as

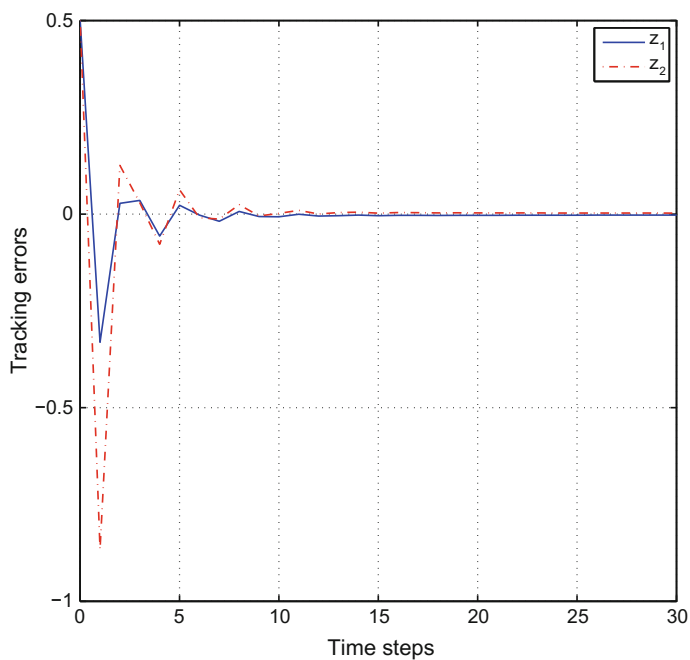


**Fig. 2.10** The trajectories of the iterative value functions with initial value function given by  $\Psi^j(z_k)$ ,  $j = 1, 2, 3, 4$ . **a**  $\Psi^1(z_k)$ . **b**  $\Psi^2(z_k)$ . **c**  $\Psi^3(z_k)$ . **d**  $\Psi^4(z_k)$

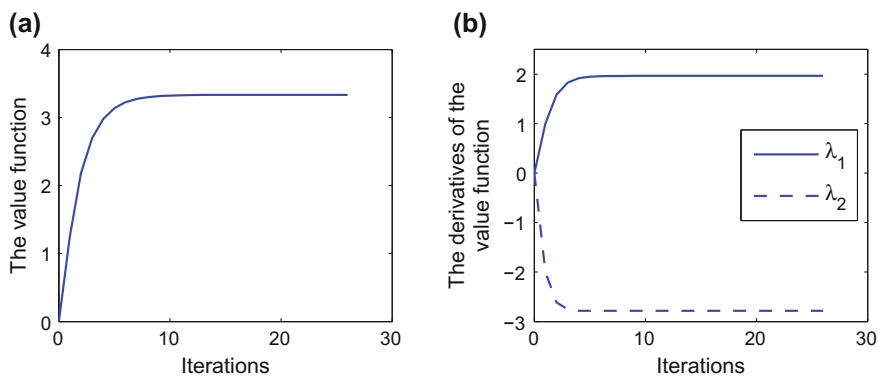
$$J(x_0) = \sum_{k=0}^{\infty} \left\{ x_k^T Q x_k + 2 \int_0^{u_k} \tanh^{-T}(\bar{U}^{-1}s) \bar{U} R ds \right\},$$

where  $Q$  and  $R$  are identity matrices with suitable dimensions and  $\bar{U} = 0.5$ .

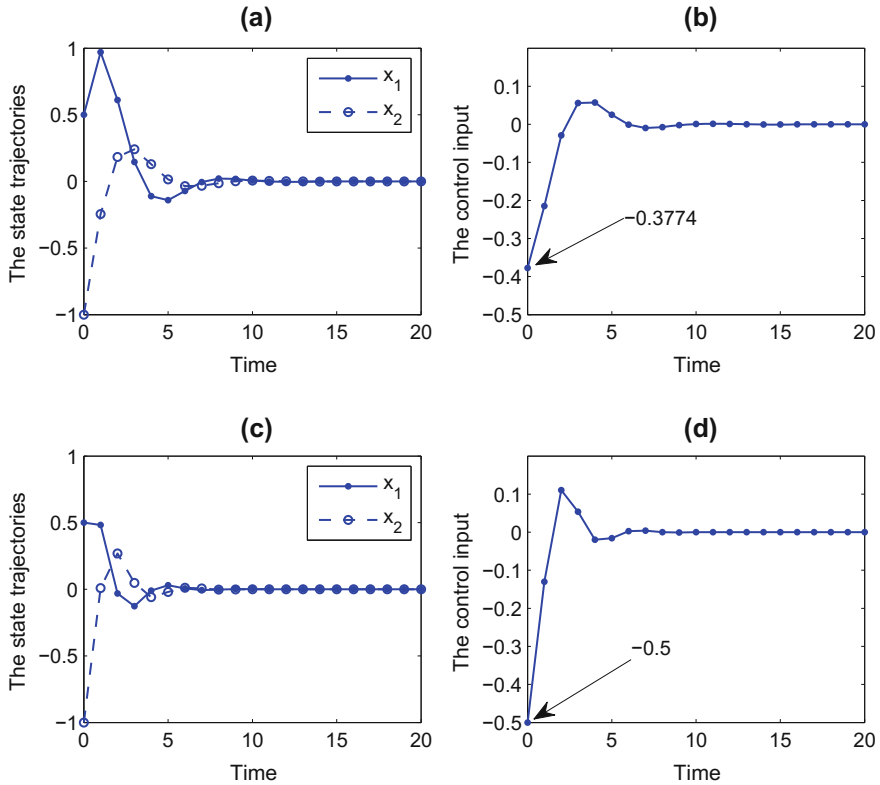
In this example, the three NNs are chosen with structures of 3–8–2, 2–8–3, and 2–8–1, respectively. Here, the initial weights of the critic network and action network are all set to be random in  $[-0.1, 0.1]$ . Then, letting the parameter  $\tau = 0.5$  and the learning rate  $\alpha_c = \alpha_a = 0.05$ , we train the critic network and action network for 26 iterations. When  $k = 0$ , the convergence process of the value function and its derivatives is depicted in Fig. 2.12.



**Fig. 2.11** The tracking error



**Fig. 2.12** **a** The convergence process of the value function. **b** The convergence process of the derivatives of the value function



**Fig. 2.13** Simulation results of Example 2.2.3. **a** The state trajectory  $x$ . **b** The control input  $u$ . **c** The state trajectory  $x$  without considering the control constraint. **d** The control input  $u$  without considering the control constraint

Next, for given initial state  $x_0 = [0.5, -1]^T$ , we apply the optimal control laws designed by the iterative GDHP algorithm, with and without considering the control constraints, to system (2.2.34) for 20 time steps, respectively. The simulation results are shown in Fig. 2.13, which also exhibits excellent control results of the iterative GDHP algorithm.

### 2.3 Iterative $\theta$ -Adaptive Dynamic Programming Algorithm for Nonlinear Systems

In this section, we present an iterative  $\theta$ -ADP algorithm for optimal control of discrete-time nonlinear systems [44]. Consider the deterministic discrete-time systems

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \dots, \quad (2.3.1)$$

where  $x_k \in \mathbb{R}^n$  is the  $n$ -dimensional state vector and  $u_k \in \mathbb{R}^m$  is the  $m$ -dimensional control vector. Let  $x_0$  be the initial state and  $F(x_k, u_k)$  be the system function.

Let  $\underline{u}_k = (u_k, u_{k+1}, \dots)$  be an arbitrary sequence of controls from  $k$  to  $\infty$ . The cost function for state  $x_0$  under the control sequence  $\underline{u}_0 = (u_0, u_1, \dots)$  is defined as

$$J(x_0, \underline{u}_0) = \sum_{k=0}^{\infty} U(x_k, u_k),$$

where  $U(x_k, u_k) > 0, \forall x_k, u_k \neq 0$ , is the utility function.

For convenience of analysis, results of this section are based on Assumptions 2.2.1–2.2.3 and the following assumption.

**Assumption 2.3.1** The utility function  $U(x_k, u_k)$  is a continuous positive-definite function of  $x_k$  and  $u_k$ .

As system (2.3.1) is controllable, there exists a stable control sequence  $\underline{u}_k = (u_k, u_{k+1}, \dots)$  that moves  $x_k$  to zero. Let  $\underline{\mathcal{U}}_k$  denote the set which contains all the stable control sequences, and let  $\mathcal{U}_k$  be the set of the stable control laws. Then, the optimal cost function can be defined as

$$J^*(x_k) = \inf_{\underline{u}_k} \{J(x_k, \underline{u}_k) : \underline{u}_k \in \underline{\mathcal{U}}_k\}. \quad (2.3.2)$$

According to the Bellman's principle of optimality,  $J^*(x_k)$  satisfies the Bellman equation

$$J^*(x_k) = \min_{u_k} \{U(x_k, u_k) + J^*(F(x_k, u_k))\}. \quad (2.3.3)$$

The corresponding optimal control law is given by

$$u^*(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + J^*(F(x_k, u_k))\}.$$

Hence, the Bellman equation (2.3.3) can be written as

$$J^*(x_k) = U(x_k, u^*(x_k)) + J^*(F(x_k, u^*(x_k))). \quad (2.3.4)$$

We can see that if we want to obtain the optimal control law  $u^*(x_k)$ , we must obtain the optimal value function  $J^*(x_k)$ . Generally speaking,  $J^*(x_k)$  is unknown before all the controls  $u_k \in \mathbb{R}^m$  are considered. If we adopt the traditional dynamic programming method to obtain the optimal value function one step at a time, then we have to face the “curse of dimensionality.” In [5, 43], iterative algorithms of ADP were used to obtain the solution of Bellman equation indirectly. However, we pointed out that the stability of the system cannot be guaranteed in [5] and an admissible control sequence



is necessary to initialize the algorithm in [43]. To overcome these difficulties, a new iterative ADP algorithm will be developed in this section.

### 2.3.1 Convergence Analysis

In the present iterative  $\theta$ -ADP algorithm, the value function and control law are updated with the iteration index  $i$  increasing from 0 to  $\infty$ . The following definition is necessary to begin the algorithm.

**Definition 2.3.1** For  $x_k \in \mathbb{R}^n$ , let

$$\tilde{\Psi}_{x_k} = \{\Psi(x_k) : \Psi(x_k) > 0, \text{ and } \exists \bar{v}(x_k) \in \mathfrak{A}_k, \text{ s.t. } \Psi(F(x_k, \bar{v}(x_k))) < \Psi(x_k)\} \quad (2.3.5)$$

be the set of initial positive-definite functions.

Let  $\Psi(x_k)$  be an arbitrary function such that  $\Psi(x_k) \in \tilde{\Psi}_{x_k}, \forall x_k \in \mathbb{R}^n$ . The existence and properties of  $\tilde{\Psi}_{x_k}$  will be discussed later. Let the initial value function

$$V_0(x_k) = \theta \Psi(x_k) \quad (2.3.6)$$

$\forall x_k \in \mathbb{R}^n$ , where  $\theta > 0$  is a finite positive constant. The iterative control law  $v_0(x_k)$  can be computed as follows:

$$\begin{aligned} v_0(x_k) &= \arg \min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\} \\ &= \arg \min_{u_k} \{U(x_k, u_k) + V_0(F(x_k, u_k))\}, \end{aligned} \quad (2.3.7)$$

where  $V_0(x_{k+1}) = \theta \Psi(x_{k+1})$ . For  $i = 1, 2, \dots$ , the iterative  $\theta$ -ADP algorithm will iterate between

$$\begin{aligned} V_i(x_k) &= \min_{u_k} \{U(x_k, u_k) + V_{i-1}(x_{k+1})\} \\ &= U(x_k, v_{i-1}(x_k)) + V_{i-1}(F(x_k, v_{i-1}(x_k))) \end{aligned} \quad (2.3.8)$$

and

$$\begin{aligned} v_i(x_k) &= \arg \min_{u_k} \{U(x_k, u_k) + V_i(x_{k+1})\} \\ &= \arg \min_{u_k} \{U(x_k, u_k) + V_i(F(x_k, u_k))\}. \end{aligned} \quad (2.3.9)$$

We note that the ADP algorithm (2.3.7)–(2.3.9) described above is essentially the same as those in (2.2.10)–(2.2.12) and (2.2.22)–(2.2.24). The only difference is the

choice of initial value function and the choice of utility function. Here, the utility function may be nonquadratic.

*Remark 2.3.1* Equations (2.3.7)–(2.3.9) in the iterative  $\theta$ -ADP algorithm are similar to the Bellman equation (2.3.4), but they are not the same. There are at least three obvious differences.

- (1) The Bellman equation (2.3.4) possesses a unique optimal cost function, i.e.,  $J^*(x_k)$ ,  $\forall x_k$ , while in the iterative ADP equations (2.3.7)–(2.3.9), the value functions are different for different iteration index  $i$ , i.e.,  $V_i(x_k) \neq V_j(x_k)$ ,  $\forall i \neq j$ .
- (2) The control law obtained by Bellman equation (2.3.4) is the optimal control law, i.e.,  $u^*(x_k)$ ,  $\forall x_k$ , while the control laws from the iterative ADP equations (2.3.7)–(2.3.9) are different for each iteration index  $i$ , i.e.,  $v_i(x_k) \neq v_j(x_k)$ ,  $\forall i \neq j$ , which are not optimal in general.
- (3) For any finite  $i$ , the iterative value function  $V_i(x_k)$  is a sum of finite sequence with a terminal constraint term and the property of  $V_i(x_k)$  can be seen in the following lemma (Lemma 2.3.1). But the optimal cost function  $J^*(x_k)$  in (2.3.4) is a sum of an infinite sequence. So, in general,  $V_i(x_k) \neq J^*(x_k)$ .

**Lemma 2.3.1** *Let  $x_k$  be an arbitrary state vector. If the iterative value function  $V_i(x_k)$  and the control law  $v_i(x_k)$  are obtained by (2.3.7)–(2.3.9), then  $V_i(x_k)$  can be expressed as*

$$V_i(x_k) = \sum_{j=0}^i U(x_{k+j}, v_{i-j}(x_{k+j})) + \theta \Psi(x_{k+i+1}).$$

*Proof* According to (2.3.8), we have

$$\begin{aligned} V_i(x_k) = \min_{u_k} \left\{ U(x_k, u_k) + \min_{u_{k+1}} \left\{ U(x_{k+1}, u_{k+1}) \right. \right. \\ \left. \left. + \cdots + \min_{u_{k+i-1}} \{ U(x_{k+i-1}, u_{k+i-1}) + V_1(x_{k+i}) \} \cdots \right\} \right\}, \end{aligned} \quad (2.3.10)$$

where

$$V_1(x_{k+i}) = \min_{u_{k+i}} \{ U(x_{k+i}, u_{k+i}) + \theta \Psi(x_{k+i+1}) \}.$$

Define

$$\underline{u}_k^N = (u_k, u_{k+1}, \dots, u_N)$$

as a finite sequence of controls from  $k$  to  $N$ , where  $N \geq k$  is an arbitrary positive integer. Then, (2.3.10) can be written as

$$\begin{aligned}
V_i(x_k) &= \min_{\underline{u}_k^{k+i}} \{U(x_k, u_k) + U(x_{k+1}, u_{k+1}) + \cdots \\
&\quad + U(x_{k+i}, u_{k+i}) + \theta \Psi(x_{k+i+1})\} \\
&= \sum_{j=0}^i U(x_{k+j}, v_{i-j}(x_{k+j})) + \theta \Psi(x_{k+i+1}).
\end{aligned}$$

The proof is complete.

In the above, we can see that the optimal value function  $J^*(x_k)$  is replaced by a sequence of iterative value functions  $V_i(x_k)$  and the optimal control law  $u^*(x_k)$  is replaced by a sequence of iterative control laws  $v_i(x_k)$ , where  $i \geq 0$  is the iteration index. As (2.3.8) is not a Bellman equation, generally speaking, the iterative value function  $V_i(x_k)$  is not optimal. However, we can prove that  $J^*(x_k)$  is the limit of  $V_i(x_k)$  as  $i \rightarrow \infty$ . Next, the convergence properties will be analyzed.

**Lemma 2.3.2** *Let  $\mu(x_k) \in \mathfrak{A}_k$  be an arbitrary control law, and let  $V_i(x_k)$  and  $v_i(x_k)$  be expressed as in (2.3.7)–(2.3.9), respectively. Define a new value function  $P_i(x_k)$  as*

$$P_{i+1}(x_k) = U(x_k, \mu(x_k)) + P_i(x_{k+1}), \quad (2.3.11)$$

with  $P_0(x_k) = V_0(x_k) = \theta \Psi(x_k)$ ,  $\forall x_k$ , then  $V_i(x_k) \leq P_i(x_k)$ .

From the definition given in (2.3.11), if we let  $\mu(x_k) = u^*(x_k)$ , then

$$\lim_{i \rightarrow \infty} P_i(x_k) = J^*(x_k).$$

In general, we have

$$P_i(x_k) \geq J^*(x_k), \quad \forall i, x_k.$$

**Theorem 2.3.1** *Let  $x_k$  be an arbitrary state vector. The iterative control law  $v_i(x_k)$  and the iterative value function  $V_i(x_k)$  are obtained by (2.3.7)–(2.3.9). If Assumptions 2.2.1–2.2.3 and 2.3.1 hold, then for any finite  $i = 0, 1, \dots$ , there exists a finite  $\theta > 0$  such that the iterative value function  $V_i(x_k)$  is a monotonically nonincreasing sequence for  $i = 0, 1, \dots$ , i.e.,*

$$V_{i+1}(x_k) \leq V_i(x_k), \quad \forall i. \quad (2.3.12)$$

*Proof* To obtain the conclusion, we will show that for an arbitrary finite  $i < \infty$ , there exists a finite  $\theta_i > 0$  such that (2.3.12) holds. We prove this by mathematical induction.

First, we let  $i = 0$ . Let  $\mu(x_k) \in \mathfrak{A}_k$  be an arbitrary stable control law. Define the value function  $P_i(x_k)$  as in (2.3.11). For  $i = 0$ , we have

$$\begin{aligned}
P_1(x_k) &= U(x_k, \mu(x_k)) + P_0(x_{k+1}) \\
&= U(x_k, \mu(x_k)) + \theta \Psi(F(x_k, \mu(x_k))).
\end{aligned}$$

According to Definition 2.3.1, there exists a stable control law  $\bar{v}_k = \bar{v}(x_k)$  such that

$$\Psi(x_k) - \Psi(F(x_k, \bar{v}(x_k))) > 0.$$

As  $\bar{v}(x_k)$  is a stable control law, the utility function  $U(x_k, \bar{v}(x_k))$  is finite. Then, there exists a finite  $\theta_0 > 0$  such that

$$\theta_0[\Psi(x_k) - \Psi(F(x_k, \bar{v}(x_k)))] \geq U(x_k, \bar{v}(x_k)).$$

As  $\mu(x_k) \in \mathfrak{A}_k$  is arbitrary, we can let  $\mu(x_k) = \bar{v}(x_k)$ . Let  $\theta = \theta_0$  and

$$P_0(x_k) = V_0(x_k) = \theta_0 \Psi(x_k). \quad (2.3.13)$$

We can get

$$\theta_0 \Psi(x_k) \geq U(x_k, \bar{v}(x_k)) + \theta_0 \Psi(F(x_k, \bar{v}(x_k))) = P_1(x_k).$$

According to Lemma 2.3.2, we have

$$\begin{aligned}
V_1(x_k) &= \min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\} \\
&= \min_{u_k} \{U(x_k, u_k) + \theta_0 \Psi(x_{k+1})\} \\
&\leq U(x_k, \bar{v}(x_k)) + \theta_0 \Psi(F(x_k, \bar{v}(x_k))) \\
&= P_1(x_k).
\end{aligned} \quad (2.3.14)$$

According to (2.3.13) and (2.3.14), we can obtain

$$V_0(x_k) = \theta_0 \Psi(x_k) \geq P_1(x_k) \geq V_1(x_k),$$

which proves  $V_0(x_k) \geq V_1(x_k)$ .

Hence, the conclusion holds for  $i = 0$ . Assume that for  $i = l - 1, l = 1, 2, \dots$ , there exists a finite  $\theta_{l-1}$  such that (2.3.12) holds. Now, we consider the situation for  $i = l$ . According to Lemma 2.3.1, for all  $\tilde{\theta}_l > 0$ , the iterative value function  $V_l(x_k)$  can be expressed as

$$V_l(x_k) = \sum_{j=0}^{l-1} U(x_{k+j}, v_{l-j-1}(x_{k+j})) + \tilde{\theta}_l \Psi(x_{k+l}), \quad (2.3.15)$$

where  $v_l(x_k)$  is the iterative control law satisfying (2.3.9), and

$$V_0(x_k) = P_0(x_k) = \tilde{\theta}_l \Psi(x_k).$$

Let

$$u_k = v_{l-1}(x_k), u_{k+1} = v_{l-2}(x_{k+1}), \dots, u_{k+l-1} = v_0(x_{k+l-1}).$$

Then, the iterative value function  $P_{l+1}(x_k)$  can be derived as

$$\begin{aligned} P_{l+1}(x_k) &= U(x_k, v_{l-1}(x_k)) + U(x_{k+1}, v_{l-2}(x_{k+1})) \\ &\quad + \dots + U(x_{k+l-1}, v_0(x_{k+l-1})) \\ &\quad + U(x_{k+l}, \mu(x_{k+l})) + P_0(x_{k+l+1}) \\ &= \sum_{j=0}^{l-1} U(x_{k+j}, v_{l-j-1}(x_{k+j})) + U(x_{k+l}, \mu(x_{k+l})) + \tilde{\theta}_l \Psi(x_{k+l+1}), \end{aligned} \quad (2.3.16)$$

where  $\mu(x_{k+l}) \in \mathfrak{A}_{k+l}$ . According to Definition 2.3.1, there exists a stable control law  $\bar{v}(x_{k+l}) \in \mathfrak{A}_{k+l}$  such that

$$\Psi(x_{k+l}) - \Psi(F(x_{k+l}, \bar{v}(x_{k+l}))) > 0.$$

Thus, there exists a finite  $\theta_l$  satisfying

$$\theta_l [\Psi(x_{k+l}) - \Psi(F(x_{k+l}, \bar{v}(x_{k+l})))] \geq U(x_{k+l}, \bar{v}(x_{k+l})). \quad (2.3.17)$$

Let  $\mu(x_{k+l}) = \bar{v}(x_{k+l})$ , and  $\tilde{\theta}_l = \theta_l$ . Then, according to (2.3.15)–(2.3.17), we can get

$$\begin{aligned} V_l(x_k) &= \sum_{j=0}^{l-1} U(x_{k+j}, v_{l-j-1}(x_{k+j})) + \theta_l \Psi(x_{k+l}) \\ &\geq \sum_{j=0}^{l-1} U(x_{k+j}, v_{l-j-1}(x_{k+j})) + U(x_{k+l}, \bar{v}(x_{k+l})) + \theta_l \Psi(x_{k+l+1}) \\ &= P_{l+1}(x_k). \end{aligned}$$

According to Lemma 2.3.2, we have  $V_{l+1}(x_k) \leq P_{l+1}(x_k)$ . Therefore, we obtain

$$V_{l+1}(x_k) \leq V_l(x_k).$$

The mathematical induction is complete. On the other hand, as  $i$  is finite, if we let  $\tilde{\theta} = \max\{\theta_0, \theta_1, \dots, \theta_i\}$ , then we can choose an arbitrary finite  $\theta$  that satisfies  $\theta \geq \tilde{\theta}$  such that (2.3.12) holds. The proof is complete.

*Remark 2.3.2* In (2.3.17), for all  $i = 1, 2, \dots$ , if we choose a  $\theta_i$  such that

$$\theta_i[\Psi(x_{k+i}) - \Psi(F(x_{k+i}, \bar{v}(x_{k+i})))] > U(x_{k+i}, \bar{v}_{k+i})$$

holds, then we can obtain (2.3.12). In this situation, the iterative value function  $V_i(x_k)$  is a monotonically decreasing sequence for  $i = 0, 1, \dots$

**Theorem 2.3.2** *Let  $x_k$  be an arbitrary state vector. If Assumptions 2.2.1–2.2.3 and 2.3.1 hold and there exists a control law  $\bar{v}(x_k) \in \mathfrak{A}_k$  which satisfies (2.3.5) such that the following limit*

$$\lim_{x_k \rightarrow 0} \frac{U(x_k, \bar{v}(x_k))}{\Psi(x_k) - \Psi(F(x_k, \bar{v}(x_k)))} \quad (2.3.18)$$

*exists, then there exists a finite  $\theta > 0$  such that (2.3.12) is true.*

*Proof* According to (2.3.17) in Theorem 2.3.1, we can see that for any finite  $i < \infty$ , the parameter  $\theta_i$  should satisfy

$$\theta_i \geq \frac{U(x_{k+i}, \bar{v}(x_{k+i}))}{\Psi(x_{k+i}) - \Psi(F(x_{k+i}, \bar{v}(x_{k+i})))}$$

in order for (2.3.12) to be true. Let  $i \rightarrow \infty$ . We have

$$\lim_{i \rightarrow \infty} \theta_i \geq \lim_{i \rightarrow \infty} \frac{U(x_{k+i}, \bar{v}(x_{k+i}))}{\Psi(x_{k+i}) - \Psi(F(x_{k+i}, \bar{v}(x_{k+i})))}. \quad (2.3.19)$$

We can see that if the limit of the right-hand side of (2.3.19) exists, then  $\theta_\infty = \lim_{i \rightarrow \infty} \theta_i$  can be defined. Therefore, if we define

$$\bar{\theta} = \sup\{\theta_0, \theta_1, \dots, \theta_\infty\}, \quad (2.3.20)$$

then  $\bar{\theta}$  can be well defined. Hence, we can choose an arbitrary finite  $\theta$  which satisfies

$$\theta \geq \bar{\theta}, \quad (2.3.21)$$

such that (2.3.12) is true.

On the other hand,  $\bar{v}(x_k) \in \mathfrak{A}_k$  is a stable control law. We have  $x_k \rightarrow 0$  as  $k \rightarrow \infty$  under the stable control sequence  $(\bar{v}_k, \bar{v}_{k+1}, \dots)$ , where  $\bar{v}_k = \bar{v}(x_k)$  for all  $k = 0, 1, \dots$ . If (2.3.18) is finite, then according to (2.3.19) and (2.3.20), there exists a finite  $\theta$  such that

$$\theta \geq \lim_{x_k \rightarrow 0} \frac{U(x_k, \bar{v}(x_k))}{\Psi(x_k) - \Psi(F(x_k, \bar{v}(x_k)))}.$$

The proof is complete.

**Remark 2.3.3** In this section, we expect that the iterative value function  $V_i(x_k) \rightarrow J^*(x_k)$  and the iterative control law  $v_i(x_k) \rightarrow u^*(x_k)$ . It is obvious that  $u^*(x_k) \in \mathfrak{A}_k$ .

If we put  $u^*(x_k)$  into (2.3.11), then for  $i \rightarrow \infty$ ,  $\lim_{i \rightarrow \infty} P_i(x_k) = J^*(x_k)$  holds for any finite  $\theta$ .

From Theorems 2.3.1 and 2.3.2, we can see that if there exists a finite  $\theta$  such that (2.3.12) holds, then  $V_i(x_k) \geq 0$  and it is a nonincreasing sequence with lower bound for iteration index  $i = 0, 1, \dots$ . We can derive the following theorem.

**Theorem 2.3.3** *Let  $x_k$  be an arbitrary state vector. Define the value function  $V_\infty(x_k)$  as the limit of the iterative value function  $V_i(x_k)$ , i.e.,*

$$V_\infty(x_k) = \lim_{i \rightarrow \infty} V_i(x_k).$$

Then,

$$V_\infty(x_k) = \min_{u_k} \{U(x_k, u_k) + V_\infty(x_{k+1})\}. \quad (2.3.22)$$

*Proof* Let  $\mu(x_k)$  be an arbitrary stable control law. According to Theorem 2.3.1,  $\forall i = 0, 1, \dots$ , we have

$$V_\infty(x_k) \leq V_{i+1}(x_k) \leq U(x_k, \mu(x_k)) + V_i(x_{k+1}).$$

Let  $i \rightarrow \infty$ . We then have

$$V_\infty(x_k) \leq U(x_k, \mu(x_k)) + V_\infty(x_{k+1}).$$

So,

$$V_\infty(x_k) \leq \min_{u_k} \{U(x_k, u(x_k)) + V_\infty(x_{k+1})\}. \quad (2.3.23)$$

Let  $\varepsilon > 0$  be an arbitrary positive number. Since  $V_i(x_k)$  is nonincreasing for all  $i$  and  $\lim_{i \rightarrow \infty} V_i(x_k) = V_\infty(x_k)$ , there exists a positive integer  $p$  such that

$$V_p(x_k) - \varepsilon \leq V_\infty(x_k) \leq V_p(x_k).$$

Then, we let

$$V_p(x_k) = \min_{u_k} \{U(x_k, u_k) + V_{p-1}(x_{k+1})\} = U(x_k, v_{p-1}(x_k)) + V_{p-1}(x_{k+1}).$$

Hence,

$$\begin{aligned} V_\infty(x_k) &\geq V_p(x_k) - \varepsilon \\ &\geq U(x_k, v_{p-1}(x_k)) + V_{p-1}(x_{k+1}) - \varepsilon \\ &\geq U(x_k, v_{p-1}(x_k)) + V_\infty(x_{k+1}) - \varepsilon \\ &\geq \min_{u_k} \{U(x_k, u_k) + V_\infty(x_{k+1})\} - \varepsilon. \end{aligned}$$

Since  $\varepsilon$  is arbitrary, we have

$$V_\infty(x_k) \geq \min_{u_k} \{U(x_k, u_k) + V_\infty(x_{k+1})\}. \quad (2.3.24)$$

Combining (2.3.23) and (2.3.24), we have (2.3.22) which proves the conclusion of this theorem.

**Remark 2.3.4** Two important properties we must point out. First, from the iterative  $\theta$ -ADP algorithm (2.3.7)–(2.3.9), we see that the initial function  $\Psi(x_k)$  is arbitrarily chosen in the set  $\bar{\Psi}(x_k)$ . The parameter  $\theta$  is also arbitrarily chosen if it satisfies (2.3.21). Actually, it is not necessary to find all  $\theta_i$  to construct the set in (2.3.20). What we should do is to choose a  $\theta$  large enough to run the iterative  $\theta$ -ADP algorithm (2.3.7)–(2.3.9) and guarantee the iterative value function to be convergent. This allows for very convenient implementation of the present algorithm. Second, for different initial value  $\theta$  and different initial function  $\Psi(x_k)$ , the iterative value function of the iterative  $\theta$ -ADP algorithm will converge to the same value function. We will show this property after two necessary lemmas.

**Lemma 2.3.3** *Let  $\bar{v}(x_k) \in \mathfrak{A}_k$  be an arbitrary stable control law, and let the value function  $P_i(x_k)$  be defined in (2.3.11) with  $u = \bar{v}$ . Define a new value function  $P'_i(x_k)$  as*

$$P'_{i+1}(x_k) = U(x_k, \bar{v}(x_k)) + P'_i(x_{k+1}),$$

*with  $P'_0(x_k) = \theta' \Psi(x_k)$ ,  $\forall x_k$ . Let  $\theta$  and  $\theta'$  be two different finite constants which satisfy (2.3.21), i.e., let  $\theta \geq \bar{\theta}$  and  $\theta' \geq \bar{\theta}$  such that (2.3.12) is true. Then,  $P_\infty(x_k) = P'_\infty(x_k) = \Gamma_\infty(x_k)$ , where  $\Gamma_\infty(x_k)$  is defined as*

$$\Gamma_\infty(x_k) = \lim_{i \rightarrow \infty} \left\{ \sum_{j=0}^i U(x_{k+j}, \bar{v}(x_{k+j})) \right\}.$$

*Proof* According to the definitions of  $P_i(x_k)$  and  $P'_i(x_k)$ , we have

$$\begin{aligned} P_i(x_k) &= \sum_{j=0}^i U(x_{k+j}, \bar{v}(x_{k+j})) + \theta \Psi(x_{k+i+1}), \\ P'_i(x_k) &= \sum_{j=0}^i U(x_{k+j}, \bar{v}(x_{k+j})) + \theta' \Psi(x_{k+i+1}), \end{aligned}$$

where  $\theta$  and  $\theta'$  both satisfy (2.3.21), and  $\theta \neq \theta'$ . As  $\bar{v}(x_k)$  is a stable control law, we have  $x_{k+i} \rightarrow 0$  as  $i \rightarrow \infty$ . Then,  $\lim_{k \rightarrow \infty} \theta \Psi(x_k) = \lim_{k \rightarrow \infty} \theta' \Psi(x_k) = 0$  since  $x_k \rightarrow 0$ .

So, we can get



$$P_\infty(x_k) = P'_\infty(x_k) = \lim_{i \rightarrow \infty} \left\{ \sum_{j=0}^i U(x_{k+j}, \bar{v}(x_{k+j})) \right\} = \Gamma_\infty(x_k).$$

The proof is complete.

Next, we will prove that the iterative value function  $V_i(x_k)$  converges to the optimal value function  $J^*(x_k)$  as  $i \rightarrow \infty$ . Before we give the optimality theorem, the following lemma is necessary.

### 2.3.2 Optimality Analysis

The following lemma is needed for the optimality analysis.

**Lemma 2.3.4** *Let  $V_i(x_k)$  be defined in (2.3.7)–(2.3.9) and  $P_i(x_k)$  be defined in (2.3.11), for  $i = 0, 1, \dots$ . Let  $\theta$  satisfy (2.3.21). Then, there exists a finite positive integer  $q$  such that*

$$P_q(x_k) \leq J^*(x_k) + \varepsilon, \quad \forall \varepsilon.$$

*Proof* According to Lemma 2.3.1, we have

$$V_\infty(x_k) = \lim_{i \rightarrow \infty} \left\{ \sum_{j=0}^i U(x_{k+j}, v_{i-j}(x_{k+j})) \right\}.$$

According to the definition of  $J^*(x_k)$ , we have  $V_\infty(x_k) \geq J^*(x_k)$ . Let  $q$  be an arbitrary finite positive integer. According to Theorems 2.3.1 and 2.3.3, we have  $V_q(x_k) \geq V_\infty(x_k)$ . According to Lemma 2.3.2, we have

$$P_q(x_k) \geq V_q(x_k) \geq V_\infty(x_k) \geq J^*(x_k).$$

Next, let

$$\underline{\mu}_k^{k+q-1} = \underline{u}_k^{*(k+q-1)}.$$

We can obtain

$$P_q(x_k) - J^*(x_k) = \theta \Psi(x_{k+q}) - \sum_{j=q}^{\infty} U(x_{k+j}, u_{k+j}^*) \geq 0.$$

From (2.3.11), as  $\mu(x_k)$  is a stable control law, the control sequence  $\underline{\mu}_k = (\mu_k, \mu_{k+1}, \dots)$  under the stable control law  $\mu(x_k)$  is a stable control sequence. Hence, we can get  $\theta \Psi(x_{k+q}) \rightarrow 0$  as  $q \rightarrow \infty$ . Then, from the fact that

$$\lim_{q \rightarrow \infty} \sum_{j=q}^{\infty} U(x_{k+j}, u_{k+j}^*) = 0,$$

we can obtain

$$\lim_{q \rightarrow \infty} \left\{ \theta \Psi(x_{k+q}) - \sum_{j=q}^{\infty} U(x_{k+j}, u_{k+j}^*) \right\} = 0.$$

Therefore,  $\forall \varepsilon > 0$ , there exists a finite  $q$  such that  $P_q(x_k) - J^*(x_k) \leq \varepsilon$  holds. This completes the proof of the lemma.

**Theorem 2.3.4** *Let  $V_i(x_k)$  be defined by (2.3.8) where  $\theta$  satisfies (2.3.21). If the system state  $x_k$  is controllable, then  $V_i(x_k)$  converges to the optimal cost function  $J^*(x_k)$  as  $i \rightarrow \infty$ , i.e.,*

$$V_i(x_k) \rightarrow J^*(x_k), \text{ as } i \rightarrow \infty, \forall x_k.$$

*Proof* According to the definition of  $J^*(x_k)$  in (2.3.2), we have

$$J^*(x_k) \leq V_i(x_k).$$

Then, let  $i \rightarrow \infty$ . We have

$$J^*(x_k) \leq V_{\infty}(x_k). \quad (2.3.25)$$

Let  $\varepsilon > 0$  be an arbitrary positive number. According to Lemma 2.3.4, there exists a finite positive integer  $q$  such that

$$V_q(x_k) \leq P_q(x_k) \leq J^*(x_k) + \varepsilon. \quad (2.3.26)$$

On the other hand, according to Theorem 2.3.1, we have

$$V_{\infty}(x_k) \leq V_q(x_k). \quad (2.3.27)$$

Combining (2.3.26) and (2.3.27), we have  $V_{\infty}(x_k) \leq J^*(x_k) + \varepsilon$ . As  $\varepsilon$  is an arbitrary positive number, we have

$$V_{\infty}(x_k) \leq J^*(x_k). \quad (2.3.28)$$

According to (2.3.25) and (2.3.28), we have

$$V_{\infty}(x_k) = J^*(x_k).$$

The proof is complete.

We can now derive the following corollary.

**Corollary 2.3.1** *Let the value function  $V_i(x_k)$  be defined by (2.3.8). If the system state  $x_k$  is controllable and Theorem 2.3.4 holds, then the iterative control law  $v_i(x_k)$  converges to the optimal control law  $u^*(x_k)$ .*

As is known, the stability property of control systems is a most basic and necessary property for any control systems. So, we will give the stability analysis for system (2.3.1) under the iterative  $\theta$ -ADP algorithm (2.3.7)–(2.3.9).

**Theorem 2.3.5** *Let  $x_k$  be an arbitrary controllable state. For  $i = 0, 1, \dots$ , if Assumptions 2.2.1–2.2.3 and 2.3.1 hold and the iterative value function  $V_i(x_k)$  and iterative control law  $v_i(x_k)$  are defined by (2.3.7)–(2.3.9) where  $\theta$  satisfies (2.3.21), then  $v_i(x_k)$  is an asymptotically stable control law for system (2.3.1),  $\forall i = 0, 1, \dots$*

*Proof* The theorem will be proved in two steps.

(1) *Show that the iterative value function  $V_i(x_k)$  is a positive-definite function,  $\forall i = 0, 1, \dots$*

For the iterative  $\theta$ -ADP algorithm, we have  $V_0(x_k) = \theta \Psi(x_k)$ .

According to Assumption 2.3.1,  $V_i(x_k)$  is a positive-definite function for  $i = 0$ .

Assume that for  $i = l$ ,  $V_l(x_k)$  is a positive-definite function. Then, for  $i = l + 1$ , (2.3.9) holds. Let  $x_k = 0$ , and we can get

$$V_{l+1}(0) = U(0, v_l(0)) + V_l(F(0, v_l(0))).$$

According to Assumptions 2.2.1–2.2.3 and 2.3.1, we have  $v_l(0) = 0$ ,  $F(0, v_l(0)) = 0$ ,  $U(0, v_l(0)) = 0$ . As  $V_l(x_k)$  is a positive-definite function, we have  $V_l(0) = 0$ . Then, we have  $V_{l+1}(0) = 0$ . If  $x_k \neq 0$ , according to Assumption 2.3.1, we have  $V_{l+1}(x_k) > 0$ . On the other hand, let  $\|x_k\| \rightarrow \infty$ . As  $U(x_k, u_k)$  is a positive-definite function,  $V_{l+1}(x_k) \rightarrow \infty$ . So,  $V_{l+1}(x_k)$  is a positive-definite function. The mathematical induction is complete.

(2) *Show that  $v_i(x_k)$  is an asymptotically stable control law for system (2.3.1).*

As the iterative value function  $V_i(x_k)$  is a positive-definite function,  $\forall i = 0, 1, \dots$ , according to (2.3.8), we have

$$V_i(F(x_k, v_i(x_k))) - V_{i+1}(x_k) = -U(x_k, v_i(x_k)) \leq 0.$$

According to Theorem 2.3.1, we have  $V_{i+1}(x_k) \leq V_i(x_k)$ ,  $\forall i \geq 0$ . Then, for all  $x_k \neq 0$ , we can obtain

$$\begin{aligned} V_i(F(x_k, v_i(x_k))) - V_i(x_k) &\leq V_i(F(x_k, v_i(x_k))) - V_{i+1}(x_k) \\ &= -U(x_k, v_i(x_k)) < 0. \end{aligned}$$

For  $i = 0, 1, \dots$ , the iterative value function  $V_i(x_k)$  is a Lyapunov function [20, 26, 30]. Therefore, the conclusion is proved.

Next, we will prove that the optimal control law  $u^*(x_k)$  is an admissible control law for system (2.3.1).

**Theorem 2.3.6** *Let  $x_k$  be an arbitrary controllable state. For  $i = 0, 1, \dots$ , if Assumptions 2.2.1–2.2.3 and 2.3.1 hold and the iterative value function  $V_i(x_k)$  and iterative control law  $v_i(x_k)$  are defined by (2.3.7)–(2.3.9) where  $\theta$  satisfies (2.3.21), then the optimal control law  $u^*(x_k)$  is an admissible control law for system (2.3.1).*

The proof of this theorem can be done by considering the fact that  $J^*(x_k)$  is finite. Therefore, we omit the details here.

**Remark 2.3.5** From the above analysis, we can see that the present iterative  $\theta$ -ADP algorithm is different from VI algorithms in [5, 52]. The main differences can be summarized as follows.

- (1) The initial conditions are different. In [5, 52], VI algorithms are initialized by zero, i.e.,  $V_0(x_k) \equiv 0, \forall x_k$ . In this section, the iterative  $\theta$ -ADP algorithm is initialized by  $\theta\psi(x_k) \neq 0$ .
- (2) The convergence properties are different. For VI algorithms in [5, 52], the iterative value function  $V_i(x_k)$  is monotonically nondecreasing and converges to the optimum. In this section, the iterative value function  $V_i(x_k)$  in the  $\theta$ -ADP algorithm is monotonically nonincreasing and converges to the optimal one.
- (3) We emphasize that the properties of the iterative control laws are different. For the VI algorithms in [5, 52], the stability of iterative control laws cannot be guaranteed, which means the VI algorithm can only be implemented off-line. In this section, it is proved that for all  $i = 0, 1, \dots$ , the iterative control law  $v_i(x_k)$  is a stable control law. This means that the present iterative  $\theta$ -ADP algorithm is feasible for implementations both online and off-line. This is an obvious merit of the present iterative  $\theta$ -ADP algorithm. In the simulation study, we will provide simulation comparisons between the VI algorithms in [5, 52] and the present iterative  $\theta$ -ADP algorithm. This conclusion echoes the observation in Remark 2.2.2.

### 2.3.3 Summary of Iterative $\theta$ -ADP Algorithm

In the previous development, we can see that an initial positive-definite function  $\Psi(x_k) \in \tilde{\Psi}_{x_k}$  is needed to start the iterative  $\theta$ -ADP algorithm. So, the existence of the set  $\tilde{\Psi}_{x_k}$  is important for the algorithm. Next, we will show the  $\tilde{\Psi}_{x_k} \neq \emptyset$ , where  $\emptyset$  is the empty set.

**Theorem 2.3.7** *Let  $x_k$  be an arbitrary controllable state, and let  $J^*(x_k)$  be the optimal cost function expressed by (2.3.2). If Assumptions 2.2.1–2.2.3 and 2.3.1 hold, then*

$$J^*(x_k) \in \bar{\Psi}_{x_k}.$$

*Proof* By Assumption 2.3.1 and the definition of  $J^*(x_k)$  in (2.3.2) and (2.3.4), we can see that

$$J^*(x_k) = \lim_{N \rightarrow \infty} \left\{ \sum_{k=0}^N U(x_{k+j}, u^*(x_{k+j})) \right\}$$

is a positive-definite function of  $x_k$ . From (2.3.4), we can also obtain

$$J^*(F(x_k, u^*(x_k))) \leq J^*(F(x_k)), \quad \forall x_k.$$

This completes the proof of the theorem.

According to Theorem 2.3.7,  $\bar{\Psi}_{x_k}$  is not an empty set. While generally, the optimal value is difficult to obtain before the algorithm is complete. So, some other methods are established to obtain  $\Psi(x_k)$ .

**Corollary 2.3.2** *Let  $x_k$  be an arbitrary state vector. If  $\Psi(x_k)$  is a Lyapunov function of system (2.3.1), then  $\Psi(x_k) \in \bar{\Psi}_{x_k}$ .*

**Remark 2.3.6** According to the definition of admissible control law, we can see that  $\Psi_{x_k} \in \bar{\Psi}_{x_k}$  is equivalent to that  $\Psi_{x_k}$  is a Lyapunov function. There are two properties we should point out. First, the general purpose of choosing a Lyapunov function  $\Psi(x_k)$  is to find a control  $\bar{v}(x_k)$  to stabilize the system. In this section, however, the purpose of choosing the initial function  $\theta\Psi(x_k)$  is to obtain the optimal control of the system (not only to stabilize the system but also to minimize the value function). Second, if we adopt  $V_0(x_k) = \Psi(x_k)$  to initialize the system, then the initial iterative control law  $v'_0(x_k)$  can be obtained by

$$v'_0(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + \Psi(x_{k+1})\}.$$

We should point out that  $v'_0(x_k)$  may not be a stable control law for the system, although the algorithm is initialized by a Lyapunov function. Using the present iterative  $\theta$ -ADP algorithm (2.3.7)–(2.3.9) in this section, we can prove that all the iterative controls  $v_i(x_k)$  for  $i = 0, 1, \dots$ , are stable and simultaneously guarantee the iterative value function to converge to the optimum. Hence, our present algorithm is effective to obtain the optimal control law both online and off-line.

From Corollary 2.3.2, we can see that if we get a Lyapunov function of system (2.3.1), then  $\Psi(x_k)$  can be obtained. As Lyapunov function is also difficult to obtain, we will give some simple methods to choose the function  $\Psi(x_k)$ .

First, it is recommended to use the utility function  $U(x_k, 0)$  to start the iterative  $\theta$ -ADP algorithm, where we set  $V_0(x_k) = \theta U(x_k, 0)$  with a large  $\theta$ . If we get a  $V_1(x_k)$  such that  $V_1(x_k) < V_0(x_k)$ , then  $U(x_k, 0) \in \tilde{\Psi}_{x_k}$ .

Second, we can use NN structures of ADP to generate an initial function  $\Psi(x_k)$ . We first randomly initialize the weights of the action NN. Give an arbitrary positive-definite function  $G(x_k) > 0$  and train the critic NN to satisfy the equation

$$\hat{\Psi}(x_k) = G(x_k) + \hat{\Psi}(F(x_k, \hat{v}(x_k))),$$

where  $\hat{\Psi}(x_k)$  and  $\hat{v}(x_k)$  are outputs of critic and action networks, respectively. The NN structure and the training rule can be seen in the next section. If the critic network training is convergent, then let  $\Psi(x_k) = \hat{\Psi}(x_k)$  and the initial value function is determined.

*Remark 2.3.7* For many nonlinear systems and utility functions, such as [2, 52], we can obtain  $U(x_k, 0) \in \tilde{\Psi}_{x_k}$ . In this situation, we only need to set a large  $\theta$  for the initial condition and run the iterative  $\theta$ -ADP algorithm (2.3.7)–(2.3.9). This can reduce the amount of computation very much. If there does not exist a stable control law such that (2.3.18) is finite, then there may not exist a finite  $\theta$  such that (2.3.12) is true. In this case, we can find an initial admissible control law  $\eta(x_k)$  such that  $x_{k+N} = 0$ , where  $N \geq 1$  is an arbitrary positive integer. Let

$$V_0(x_k) = \sum_{\tau=0}^N U(x_{k+\tau}, \eta(x_{k+\tau})).$$

Then, using the algorithm (2.3.7)–(2.3.9), we can also obtain  $V_i(x_k) \leq V_{i+1}(x_k)$ . The details of proof are available in [43].

*Remark 2.3.8* The iterative  $\theta$ -ADP algorithm is different from the policy iteration algorithm in [1, 31]. For the policy iteration algorithm, an admissible control sequence is necessary to initialize the algorithm, while for the iterative  $\theta$ -ADP algorithm developed in this section, the initial admissible control sequence is avoided. Instead, we only need an arbitrary function  $\Psi(x_k) \in \tilde{\Psi}_{x_k}$  to start the algorithm. Generally speaking, for nonlinear systems, admissible control sequences are difficult to obtain, while the function  $\Psi(x_k)$  can easily be obtained (for many cases,  $U(x_k, 0) \in \tilde{\Psi}_{x_k}$ ). Second, for PI algorithms in [1, 31], during every iteration step, we need to solve a generalized Bellman equation to update the iterative control law, while in the present iterative  $\theta$ -ADP algorithm, the generalized Bellman equation is unnecessary. Therefore, the iterative  $\theta$ -ADP algorithm has more advantages than the PI algorithm.

Now, we summarize the iterative  $\theta$ -ADP algorithm as follows.

---

**Algorithm 2.3.1** Iterative  $\theta$ -adaptive dynamic programming algorithm

---

- Step 1. Choose randomly an array of initial states  $x_k$  and choose a computation precision  $\varepsilon$ . Choose an arbitrary positive definite function  $\Psi(x_k) \in \Psi_{x_k}$ . Choose a constant  $\theta > 0$ .
- Step 2. Let  $i = 0$ . Let the initial value function  $V_0(x_k) = \theta\Psi(x_k)$ .
- Step 3. Compute  $v_0(x_k)$  by (2.3.7) and obtain  $V_1(x_k)$  by (2.3.8).
- Step 4. If  $V_1(x_k) \leq V_0(x_k)$ , then go to next step. Otherwise,  $\theta$  is not large enough, and choose a larger  $\theta' > \theta$ . Let  $\theta = \theta'$  and go to Step 2.
- Step 5. Let  $i = i + 1$ . Compute  $V_i(x_k)$  by (2.3.8) and  $v_i(x_k)$  by (2.3.9).
- Step 6. If  $V_i(x_k) \leq V_{i-1}(x_k)$ , go to Step 7; else, choose a larger  $\theta' > \theta$ . Let  $\theta = \theta'$  and go to Step 2.
- Step 7. If  $|V_i(x_k) - V_{i-1}(x_k)| \leq \varepsilon$ , then go to next step; else go to Step 5.
- Step 8. Stop.
- 

*Remark 2.3.9* Generally speaking, NNs are used to implement the present iterative  $\theta$ -ADP algorithm. In order to approximate the functions  $V_i(x_k)$  and  $v_i(x_k)$ , a large number of  $x_k$  in the state space is required to train NNs. In this situation, as we have declared in Step 1, we should choose randomly an array of initial states  $x_k$  in the state space to initialize the algorithm. For all  $i = 0, 1, \dots$ , according to the array of states  $x_k$ , we can obtain the iterative value function  $V_i(x_k)$  and the iterative control law  $v_i(x_k)$  by training NNs, respectively. To the best of our knowledge, all the NN implementations for ADP require a large number of  $x_k$  in state space to approximate the iterative control laws and the iterative value functions, such as [36, 48]. The detailed NN implementation for the present iterative  $\theta$ -ADP algorithm can be found in [44].

### 2.3.4 Simulation Studies

To evaluate the performance of our iterative  $\theta$ -ADP algorithm, we choose two examples with quadratic utility functions for numerical experiments.

*Example 2.3.1* This example is chosen from [43, 52] with modifications. Consider

$$x_{k+1} = \begin{bmatrix} 0.8x_{1k} \exp(x_{2k}^2) \\ 0.9x_{2k}^3 \end{bmatrix} + \begin{bmatrix} -0.2 & 0 \\ 0 & -0.2 \end{bmatrix} u_k,$$

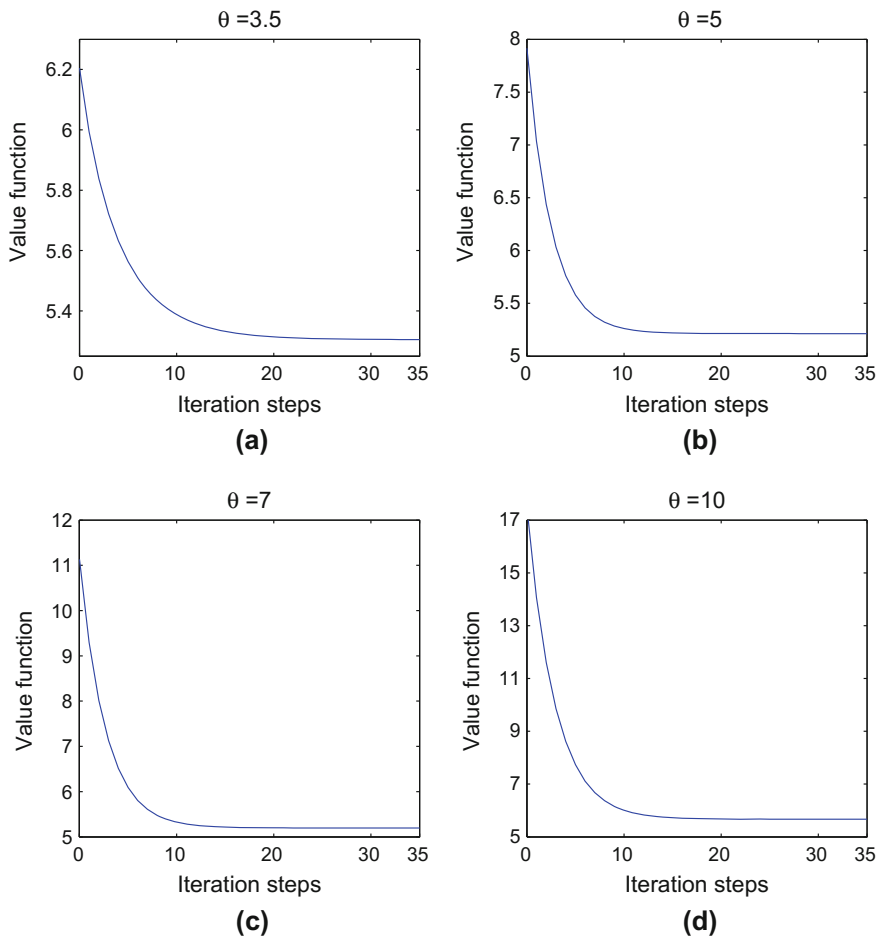
where  $x_k = [x_{1k}, x_{2k}]^T$  and  $u_k = [u_{1k}, u_{2k}]^T$  are the state and control variables, respectively. The initial state is  $x_0 = [1, -1]^T$ . The cost function is the quadratic form expressed as

$$J(x_0, \underline{u}_0^\infty) = \sum_{k=0}^{\infty} (x_k^T Q x_k + u_k^T R u_k)$$

where the matrices  $Q$  and  $R$  given as identity matrix with suitable dimensions.

Two NNs are used to implement the iterative  $\theta$ -ADP algorithm. The critic and action networks are both chosen as three-layer BP NNs with the structures of 2–8–1 and 2–8–2, respectively. For each iteration step, the critic and action networks are trained for 200 steps using the learning rate of 0.02 so that the NN training errors become less than  $10^{-6}$ . To show the effectiveness of the iterative  $\theta$ -ADP algorithm, we choose four  $\theta$ 's (including  $\theta = 3.5, 5, 7, 10$ ) to initialize the algorithm. Let the algorithm run for 35 iteration steps for different  $\theta$ 's to obtain the optimal value function. The convergence curves of the value functions are shown in Fig. 2.14.

From Fig. 2.14, we can see that all the convergence curves of value functions are monotonically nonincreasing. For convenience of analysis, we let



**Fig. 2.14** The convergence of value functions for Example 2.3.1. **a**  $\theta = 3.5$ . **b**  $\theta = 5$ . **c**  $\theta = 7$ . **d**  $\theta = 10$



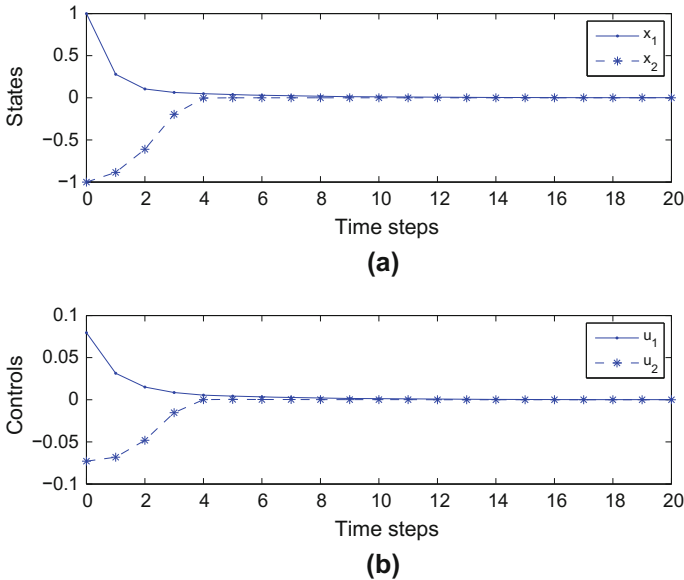
$$\theta_0 = \lim_{x_k \rightarrow 0} \frac{U(x_k, v_0(x_k))}{U(x_k, 0) - U(F(x_k, v_0(x_k)), 0)}$$

where  $v_0(x_k)$  is obtained in (2.3.7). Then, for  $\theta = 3.5$ , we have  $\theta_0 = 1.9015$ . For  $\theta = 5$ , we have  $\theta_0 = 1.90984$ . For  $\theta = 7$ , we have  $\theta_0 = 2.04469$ . For  $\theta = 10$ , we have  $\theta_0 = 2.16256$ . We can also see that if the iterative value function is convergent, then the iterative value function can converge to the optimum and the optimal value function is independent from the parameter  $\theta$ . We apply the optimal control law to the system for  $T_f = 20$  time steps and obtain the following results. The optimal state and control trajectories are shown in Fig. 2.15a and b, respectively.

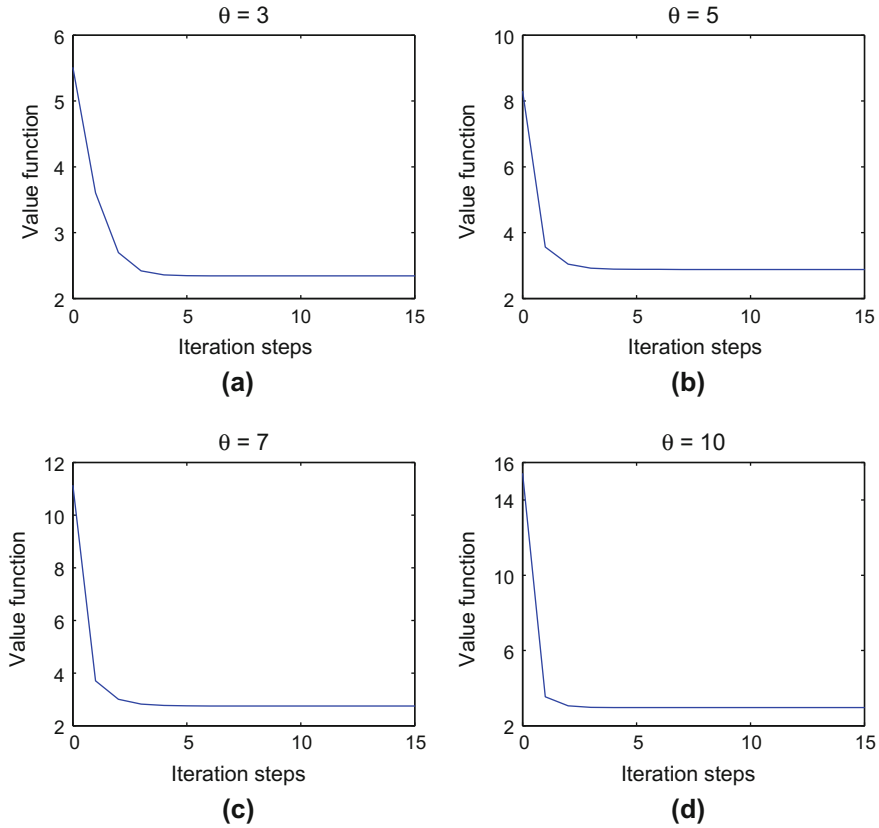
From the above simulation results, we can see that if we choose  $\theta$  large enough to initialize the iterative  $\theta$ -ADP algorithm, the iterative value function  $V_i(x_k)$  will be monotonically nonincreasing and converge to the optimum, which verifies the effectiveness of the present algorithm. Next, we enhance the complexity of the system. We will consider the situation where the autonomous system is unstable, and we will show that the present iterative  $\theta$ -ADP is still effective.

*Example 2.3.2* This example is chosen from [32, 37]. Consider

$$x_{k+1} = \begin{bmatrix} (x_{1k}^2 + x_{2k}^2 + u_k) \cos(x_{2k}) \\ (x_{1k}^2 + x_{2k}^2 + u_k) \sin(x_{2k}) \end{bmatrix}, \quad (2.3.29)$$



**Fig. 2.15** The optimal trajectories. **a** Optimal state trajectories. **b** Optimal control trajectories

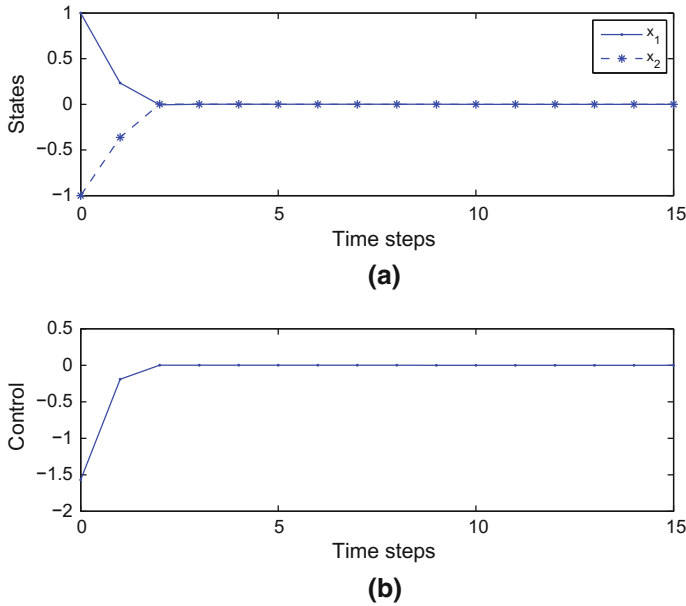


**Fig. 2.16** The convergence of value functions for Example 2.3.2. **a**  $\theta = 3$ . **b**  $\theta = 5$ . **c**  $\theta = 7$ . **d**  $\theta = 10$

where  $x_k = [x_{1k}, x_{2k}]^T$  denotes the system state vector and  $u_k$  denotes the control. The value function is the same as the one in Example 2.3.1.

The initial state is  $x_0 = [1, -1]^T$ . From system (2.3.29), we can see that  $x_k = 0$  is an equilibrium state and the autonomous system  $F(x_k, 0)$  is unstable. We also use NNs to implement the iterative ADP algorithm where four  $\theta$ 's (including  $\theta = 3, 5, 7, 10$ ) are chosen to initialize the algorithm and the convergence curves of the value functions are shown in Fig. 2.16.

Applying the optimal control law to the system for  $T_f = 15$  time steps, the optimal state and control trajectories are shown in Fig. 2.17a and b, respectively.



**Fig. 2.17** The optimal trajectories. **a** Optimal state trajectories. **b** Optimal control trajectory

## 2.4 Conclusions

In this chapter, we developed several VI-based ADP methods for optimal control problems of discrete-time nonlinear systems. First, a GVI-based ADP scheme is established to obtain optimal control for discrete-time affine nonlinear systems. Then, the GVI ADP algorithm is used to solve the optimal tracking control problem for discrete-time nonlinear systems as a generalization. Furthermore, as a case study, the VI-based ADP approach is developed to derive optimal control for discrete-time nonlinear systems with unknown dynamics and input constraints. It is emphasized that using the ADP approach, affine and nonaffine nonlinear systems can be treated uniformly. Next, an iterative  $\theta$ -ADP technique is presented to solve the optimal control problem of discrete-time nonlinear systems. Convergence analysis and optimality analysis results are established for the iterative  $\theta$ -ADP algorithm. Simulation results are provided to show the effectiveness of the present algorithm.

## References

1. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791

2. Abu-Khalaf M, Lewis FL, Huang J (2008) Neurodynamic programming and zero-sum games for constrained control systems. *IEEE Trans Neural Netw* 19(7):1243–1252
3. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2007) Adaptive critic designs for discrete-time zero-sum games with application to  $H_\infty$  control. *IEEE Trans Syst Man Cybern.-Part B: Cybern* 37(1):240–247
4. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2007) Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. *Automatica* 43(3):473–481
5. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2008) Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans Syst Man Cybern.-Part B: Cybern* 38(4):943–949
6. Apostol TM (1974) *Mathematical analysis: A modern approach to advanced calculus*. Addison-Wesley, Boston, MA
7. Athans M, Falb PL (1966) *Optimal control: an introduction to the theory and its applications*. McGraw-Hill, New York
8. Beard R, Saridis G, Wen J (1997) Galerkin approximations of the generalized Hamilton–Jacobi–Bellman equation. *Automatica* 33(12):2158–2177
9. Bellman RE (1957) *Dynamic programming*. Princeton University Press, Princeton, NJ
10. Berkovitz LD, Medhin NG (2013) *Nonlinear optimal control theory*. CRC Press, Boca Raton, FL
11. Bertsekas DP (2005) *Dynamic programming and optimal control*. Athena Scientific, Belmont, MA
12. Bitmead RR, Gever M, Petersen IR (1985) Monotonicity and stabilizability properties of solutions of the Riccati difference equation: Propositions, lemmas, theorems, fallacious conjectures and counterexamples. *Syst Control Lett* 5:309–315
13. Dierks T, Thumati BT, Jagannathan S (2009) Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence. *Neural Netw* 22(5):851–860
14. Dreyfus SE, Law AM (1977) *The art and theory of dynamic programming*. Academic Press, New York
15. Fu J, He H, Zhou X (2011) Adaptive learning and control for MIMO system based on adaptive dynamic programming. *IEEE Trans Neural Netw* 22(7):1133–1148
16. Hagan MT, Menhaj MB (1994) Training feedforward networks with the Marquardt algorithm. *IEEE Trans Neural Netw* 5(6):989–993
17. Heydari A, Balakrishnan SN (2013) Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics. *IEEE Trans Neural Netw Learn Syst* 24(1):145–157
18. Howard RA (1960) *Dynamic programming and Markov processes*. MIT Press, Cambridge, MA
19. Huang Y, Liu D (2014) Neural-network-based optimal tracking control scheme for a class of unknown discrete-time nonlinear systems using iterative ADP algorithm. *Neurocomputing* 125:46–56
20. Koppel LB (1968) *Introduction to control theory with applications to process control*. Prentice-Hall, Englewood Cliffs, NJ
21. Levin AU, Narendra KS (1993) Control of nonlinear dynamical systems using neural networks: controllability and stabilization. *IEEE Trans Neural Netw* 4(2):192–206
22. Lewis FL, Liu D (2012) *Reinforcement learning and approximate dynamic programming for feedback control*. Wiley, Hoboken, NJ
23. Lewis FL, Syrmos VL (1995) *Optimal control*. Wiley, New York
24. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst Mag* 9(3):32–50
25. Li H, Liu D (2012) Optimal control for discrete-time affine non-linear systems using general value iteration. *IET Control Theory Appl* 6(18):2725–2736
26. Liao X, Wang L, Yu P (2007) *Stability of dynamical systems*. Elsevier, Amsterdam, Netherlands

27. Lincoln B, Rantzer A (2006) Relaxing dynamic programming. *IEEE Trans Autom Control* 51(8):1249–1260
28. Liu D, Wang D, Yang X (2013) An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs. *Inf Sci* 220:331–342
29. Lyshevski SE (1998) Optimal control of nonlinear continuous-time systems: design of bounded controllers via generalized nonquadratic functionals. In: *Proceedings of the American control conference*, pp 205–209
30. Michel AN, Hou L, Liu D (2015) *Stability of dynamical systems: On the role of monotonic and non-monotonic Lyapunov functions*. Birkhäuser, Boston, MA
31. Murray JJ, Cox CJ, Lendaris GG, Sacks R (2002) Adaptive dynamic programming. *IEEE Trans Syst Man Cybern-Part C: Appl Rev* 32(2):140–153
32. Navarro-Lopez EM (2007) Local feedback passivation of nonlinear discrete-time systems through the speed-gradient algorithm. *Automatica* 43(7):1302–1306
33. Primbs JA, Nevistic V (2000) Feasibility and stability of constrained finite receding horizon control. *Automatica* 36(7):965–971
34. Prokhorov DV, Wunsch DC (1997) Adaptive critic designs. *IEEE Trans Neural Netw* 8(5):997–1007
35. Rantzer A (2006) Relaxed dynamic programming in switching systems. *IEE Proc-Control Theory Appl* 153(5):567–574
36. Si J, Wang YT (2001) On-line learning control by association and reinforcement. *IEEE Trans Neural Netw* 12(2):264–276
37. Sira-Ramirez H (1991) Non-linear discrete variable structure systems in quasi-sliding mode. *Int J Control* 54(5):1171–1187
38. Sutton RS, Barto AG (1998) *Reinforcement learning: an introduction*. MIT Press, Cambridge, MA
39. Vincent TL, Grantham WJ (1997) *Nonlinear and optimal control systems*. Wiley, New York
40. Vrabie D, Vamvoudakis KG, Lewis FL (2013) *Optimal adaptive control and differential games by reinforcement learning principles*. IET, London
41. Wang D, Liu D (2013) Neuro-optimal control for a class of unknown nonlinear dynamic systems using SN-DHP technique. *Neurocomputing* 121:218–225
42. Wang D, Liu D, Wei Q, Zhao D, Jin N (2012) Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. *Automatica* 48(8):1825–1832
43. Wang FY, Jin N, Liu D, Wei Q (2011) Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with  $\varepsilon$ -error bound. *IEEE Trans Neural Netw* 22(1):24–36
44. Wei Q, Liu D (2014) A novel iterative  $\theta$ -adaptive dynamic programming for discrete-time nonlinear systems. *IEEE Trans Autom Sci Eng* 11(4):1176–1190
45. Wei Q, Liu D, Xu Y (2014) Neuro-optimal tracking control for a class of discrete-time nonlinear systems via generalized value iteration adaptive dynamic programming. *Soft Comput* 20(2):697–706
46. Werbos PJ (1977) Advanced forecasting methods for global crisis warning and models of intelligence. *Gen Syst Yearbook* 22:25–38
47. Werbos PJ (1992) Approximate dynamic programming for real-time control and neural modeling. In: White DA, Sofge DA (eds) *Handbook of intelligent control: neural, fuzzy, and adaptive approaches* (Chapter 13). Van Nostrand Reinhold, New York
48. Yang Q, Jagannathan S (2012) Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators. *IEEE Trans Syst Man Cybern-Part B: Cybern* 42(2):377–390
49. Zhang H, Huang J, Lewis FL (2009) An improved method in receding horizon control with updating of terminal cost function. In: Valavanis KP (ed) *Applications of intelligent control to engineering systems*. Springer, New York, pp 365–393
50. Zhang H, Liu D, Luo Y, Wang D (2013) *Adaptive dynamic programming for control: algorithms and stability*. Springer, London

51. Zhang H, Luo Y, Liu D (2009) Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints. *IEEE Trans Neural Netw* 20(9):1490–1503
52. Zhang H, Wei Q, Luo Y (2008) A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm. *IEEE Trans Syst Man Cybern-Part B: Cybern* 38(4):937–942

Adaptive Dynamic Programming with Applications in  
Optimal Control

Liu, D.; Wei, Q.; Wang, D.; Yang, X.; Li, H.

2017, XXX, 594 p. 203 illus., 175 illus. in color.,

Hardcover

ISBN: 978-3-319-50813-9