Series on Deep Learning Neural Networks – Volume 2

ITERATIVE ADAPTIVE DYNAMIC PROGRAMMING FOR SELF-LEARNING OPTIMAL CONTROL

Qinglai Wei
Ruizhuo Song
Hongyang Li

World Scientific

ITERATIVE ADAPTIVE DYNAMIC PROGRAMMING FOR SELF-LEARNING OPTIMAL CONTROL

Series On Deep Learning Neural Networks

Series Editors: Daniel Graupe *(University of Illinois at Chicago, USA)* &
Derong Liu *(University of Illinois at Chicago, USA)*

Published

Vol. 2 *Iterative Adaptive Dynamic Programming for Self-Learning Optimal Control*
by Qinglai Wei, Ruizhuo Song & Hongyang Li

Vol. 1 *Intelligent Analysis of Fundus Images: Methods and Applications*
by Yuanyuan Chen, Yi Zhang & Jie Zhong

Series on Deep Learning Neural Networks – Volume 2

ITERATIVE ADAPTIVE DYNAMIC PROGRAMMING FOR SELF-LEARNING OPTIMAL CONTROL

Qinglai Wei
Chinese Academy of Sciences, China
Ruizhuo Song
University of Science and Technology Beijing, China
Hongyang Li
Chinese Academy of Sciences, China

World Scientific

NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI • TOKYO

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

Library of Congress Control Number: 2025013920

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

Series on Deep Learning Neural Networks — Vol. 2
ITERATIVE ADAPTIVE DYNAMIC PROGRAMMING FOR SELF-LEARNING OPTIMAL CONTROL

ISBN 978-981-98-1120-5 (hardcover)
ISBN 978-981-98-1121-2 (ebook for institutions)
ISBN 978-981-98-1122-9 (ebook for individuals)

For any available supplementary material, please visit
https://www.worldscientific.com/worldscibooks/10.1142/14258#t=suppl

Desk Editors: Nambirajan Karuppiah/Steven Patt

Typeset by Stallion Press
Email: enquiries@stallionpress.com

Printed in Singapore

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grants 2021YFE0206100, 2024YFB4709100; in part by the National Natural Science Foundation of China under Grants 62425310, 62273036, 62073321, 62477045; in part by the National Defense Basic Scientific Research Program JCKY2019203C029; in part by the Beijing Natural Science Foundation under Grants 4254106, L241007; in part by Science and technology Innovation Project of China Academy of Chinese Medical Sciences CI2023C005YG; in part by the Science and Technology Development Fund, Macau SAR under Grants FDCT-22-009-MISE, 0060/2021/A2, 0015/2020/AMJ; in part by the financial support from the National Defense Basic Scientific Research Project JCKY2020130C025.

This page intentionally left blank

About the Authors

Qinglai Wei Ph.D. is a Professor at The State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences.

He received his B.S. degree in automation and Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2002 and 2009, respectively. From 2009 to 2011, he was a Post-Doctoral Fellow with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. He is currently a Professor at the institute. He has authored three books and published over 80 international journal papers. His research interests include adaptive dynamic programming, neural network-based control, computational intelligence, optimal control, nonlinear systems, and their industrial applications.

Prof. Wei was a recipient of the IEEE System, Man, and Cybernetics Society Andrew P. Sage Best Transactions Paper Award, the IEEE Transactions on Neural Networks and Learning Systems Outstanding Paper Award, the IEEE/CAA Journal of Automatica Sinica Best Paper Award, the Outstanding Paper Award of Acta Automatica Sinica, and the Zhang Siying Outstanding Paper Award of the Chinese Control and Decision Conference. He was also a recipient of the Shuang-Chuang Talents in Jiangsu Province, China, and the

Young Researcher Award of the Asia-Pacific Neural Network Society. He has been the Vice President of the IEEE Computational Intelligence Society's Beijing Chapter since 2022. He was the General Co-Chair of the 2023 Symposium of Parallel Intelligence, the Invited Session Chair of the IEEE 8th Data Driven Control and Learning Systems Conference, the Special Sessions Chair of the 25th International Conference on Neural Information Processing, the Program Co-Chair of the 24th International Conference on Neural Information Processing, and the Registration Chair of the 12th World Congress on Intelligent Control and Automation and the 2014 IEEE World Congress on Computational Intelligence. He was a Guest Editor for several international journals. He is an Associate Editor-in-Chief/the Deputy Editor-in-Chief/Senior Editor of the *IEEE/CAA Journal of Automatica Sinica*, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *Neurocomputing, and Acta Automatica Sinica*. He is also an Associate Editor of *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Automation Science and Engineering*, *IEEE Transactions on Consumer Electronics*, *IEEE Transactions on Cognitive and Developmental Systems*, and *Optimal Control Applications and Methods*.

Ruizhuo Song Ph.D. is a Professor at the School of Automation and Electrical Engineering, University of Science and Technology Beijing.

She received his Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2012. From 2013 to 2014, she was a Visiting Scholar with the Department of Electrical Engineering, University of Texas at Arlington, Arlington, TX, USA. From January 2018 to February 2018, she was a Visiting Scholar with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, USA. She was a Post-Doctoral Fellow with the University of Science and Technology Beijing, Beijing, China, where she is currently a Professor at the School of Automation and Electrical Engineering. She has authored or co-authored more than 100 journals and conference articles and coauthored 3 monographs. Her current research interests

include intelligent perception and computing, optimal decision-making, and gaming for virtual/real complex systems, and the results are applied to human body signal measurement and control, energy internet, etc.

Hongyang Li Ph.D. is an Assistant Professor at The State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences.

He received his Ph.D. degree in computer application technology from the University of Chinese Academy of Sciences, Beijing, China, in 2022. His research interests include adaptive dynamic programming, reinforcement learning, optimal control, and their industrial applications.

This page intentionally left blank

Contents

List of Symbols

x	state vector
u	control vector
v	iterative control law
F	system function
$\mathbb{R}$	real number set
Ω	state set
J, Φ, $\mathcal{P}$	performance index functions
$\underline{\widehat{x}}$	state sequence
$\underline{u}$, $\underline{\nu}$, $\underline{\mu}$	control sequences
U	utility function
$\mathfrak{A}$	set of finite-horizon admissible control sequences
J^*	optimal performance index function
u^*	law of optimal control
N	terminal time
V, P	iterative performance index function
K	approximate length of optimal control
$\mathcal{T}$	set of state vector
X	array of states
ℓ	number of hidden layer neurons
Y	weight matrix between the input layer and hidden layer
W	weight matrix between the hidden layer and output layer
Q, R	positive definite matrices
α, β	learning rate
γ	discount factor
$\underline{\mathfrak{U}}$	control sequence set

Ψ	positive semi-definite function
$\mathcal{X}$	array of system state data
$\mathcal{U}$	array of control data
ξ	bounded immensurable disturbance
H	Hamiltonian function
F_N	neural network function
σ	activation function
e_c	estimation error
E_c	squared residual error
V	Lyapunov function
E	received signal level

Chapter 1

Principle of Adaptive Dynamic Programming

1.1 Dynamic Programming

Optimal control of nonlinear systems has been the focus of control fields for many decades [5, 26, 38, 57]. Dynamic programming has been an effective method for solving optimization and optimal control problems for several decades [18, 19, 42]. In this section, it aims to present a brief introduction to dynamic programming. Dynamic programming was developed by R. E. Bellman in 1957 [7]. Dynamic programming is based on Bellman's principle of optimality: An optimal policy has the property that no matter what the previous decision (i.e., controls) have been, the remaining decisions must constitute an optimal policy with regards to the state resulting from those previous decisions.

1.1.1 *Discrete-time systems*

We can use mathematical formula to express the above process. Give the following discrete-time nonlinear systems

$$x_{k+1} = F(x_k, u_k), \tag{1.1}$$

where $x_k \in \mathbb{R}^n$ is the system state and $u_k \in \mathbb{R}^m$ is the control input. Let $\underline{u}_k = (u_k, u_{k+1}, \ldots)$ denote the control sequence from k to ∞. Let $U(x_k, u_k)$ be the utility function. The performance index function is

defined as

$$J(x_k, \underline{u}_k) = \sum_{i=k}^{\infty} U(x_i, u_i). \tag{1.2}$$

The goal of this section is to find an optimal control scheme which stabilizes the system (1.1) and simultaneously minimizes the performance index function (1.2). For convenience of analysis, results of this section are based on the following assumptions.

Assumption 1.1. System (1.1) is controllable on a compact set $\Omega_x \in \mathbb{R}^n$ and the function $F(x_k, u_k)$ is Lipschitz continuous for x_k, u_k.

Assumption 1.2. The system state $x_k = 0$ is an equilibrium state of system (1.1) under the control $u_k = 0$, i.e., $F(0, 0) = 0$.

Assumption 1.3. The feed-back control $u_k = u(x_k)$ satisfies $u_k = u(x_k) = 0$ for $x_k = 0$.

Assumption 1.4. The utility function $U(x_k, u_k)$ is a continuous positive definite function for any x_k and u_k.

According to the Bellman's principle, the optimal performance index function satisfies the following Bellman equation

$$J^*(x_k) = \min_{u_k}\{U(x_k, u_k) + J^*(x_{k+1})\}. \tag{1.3}$$

Then, the law of optimal single control can be expressed as

$$u^*(x_k) = \arg\min_{u_k}\{U(x_k, u_k) + J^*(F(x_k, u_k))\}. \tag{1.4}$$

Hence, the HJB equation (1.3) can be written as

$$J^*(x_k) = U(x_k, u^*(x_k)) + J^*(F(x_k, u^*(x_k))). \tag{1.5}$$

Equation (1.3) is the principle of the optimality for discrete-time systems. Its importance lies in the fact that it allows to optimize over only one control vector at a time by working backward. From (1.4) and (1.5), if we want to obtain the optimal control law $u^*(x_k)$, we must obtain the optimal performance index function $J^*(x_k)$.

However, $J^*(x_k)$ cannot be obtained before $J^*(x_{k+1})$ is known. 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 performance index function one by one time step, then we have to face the "curse of dimensionality". It is nearly impossible to obtain $J^*(x_k)$ for all $x_k \in \Omega_x$ by solving the Bellman equation directly.

1.1.2 *Continuous-time systems*

Now consider the following continuous-time systems

$$\dot{x}(t) = f(x(t), u(t)), \tag{1.6}$$

where $x(t) \in \mathbb{R}^n$ is the system state, $u(t) \in \mathbb{R}^m$ is the control input. The performance index function is defined as

$$J(x(t)) = \int_t^\infty L(x(\tau), u(\tau))\mathrm{d}\tau, \tag{1.7}$$

where $L(x(t), u(t))$ is the utility function. The goal of the optimal control is to find an optimal control law which minimizes to the performance index function (1.7). Let $U[t, \infty]$ denote the control set in the interval $[t, \infty)$. According to the Bellman's principle, the optimal performance index function can be expressed as

$$J^*(x(t)) = \min_{u(\cdot)\in U[t,\tau]} \left\{ \int_t^\tau L(x(s), u(s))\mathrm{d}s + J^*(x(\tau)) \right\}. \tag{1.8}$$

According to (1.8), we can derive the following Hamilton–Jacobi–Bellman(HJB) equation.

$$\begin{aligned} -\frac{\partial J^*(x(t))}{\partial t} &= \min_{u\in U} \left\{ L(x(t), u(t)) + \left(\frac{\partial J^*(x(t))}{\partial x(t)} \right)^{\mathsf{T}} f(x(t), u(t)) \right\} \\ &= L(x(t), u^*(t)) + \left(\frac{\partial J^*(x(t))}{\partial x(t)} \right)^{\mathsf{T}} f(x(t), u^*(t)). \end{aligned} \tag{1.9}$$

The HJB equation (1.9) provides the solution of the optimal control problem. However, it is generally impossible to solve the optimal performance index function analytically, especially for

nonlinear systems. In this situation, numerical methods, such as adaptive dynamic programming (ADP), are necessary to obtain the optimal performance index function and the optimal control law approximately.

1.2 Original Forms of Adaptive Dynamic Programming

Adaptive dynamic programming (ADP), proposed by Werbos [53, 55], is an effective adaptive learning control approach to solve optimal control problems forward-in-time. There are several synonyms used for ADP including "adaptive critic designs" [2, 34], "adaptive dynamic programming" [12, 13, 29], "approximate dynamic programming" [4, 33, 56], "neural dynamic programming" [10], "neurodynamic programming" [9], and "reinforcement learning" [27, 37, 40]. In Ref. [56], ADP approaches were classified into four main schemes: Heuristic Dynamic Programming (HDP), Dual Heuristic Programming (DHP), Action Dependent HDP (ADHDP) (also known as Q-learning [47]), and Action Dependent DHP (ADDHP). In Ref. [34], two more ADP that are Globalized DHP (GDHP) and ADGDHP were proposed. In Refs. [31, 32], new ADP structures, which are goal representation dual heuristic dynamic programming (GrDHP), were developed to obtain the optimal control in a more effective way. In recent years, iterative methods are widely used in ADP to obtain the solution of HJB equation indirectly [1, 24, 29, 48, 51, 52]

There are two main iterative ADP algorithms which are policy and value iteration algorithms, respectively [16]. Policy iteration algorithms for optimal control of continuous-time systems with continuous state and action spaces were given in Ref. [1]. In 2011, Wang *et al.* [46] studied finite-horizon optimal control problems for discrete-time nonlinear systems with unspecified terminal time using policy iteration algorithms. Value iteration algorithms for optimal control for discrete-time nonlinear systems were given in Ref. [6]. In Ref. [3], a value iteration algorithm, which is referred to as greedy HDP iteration algorithm, was proposed for finding the optimal control law and the convergence of the algorithm was also proven. In Ref. [49], an iterative θ-ADP algorithm was proposed that permits the ADP algorithm be implemented both on-line and off-line without the initial admissible control sequence. Based on policy and value iterations,

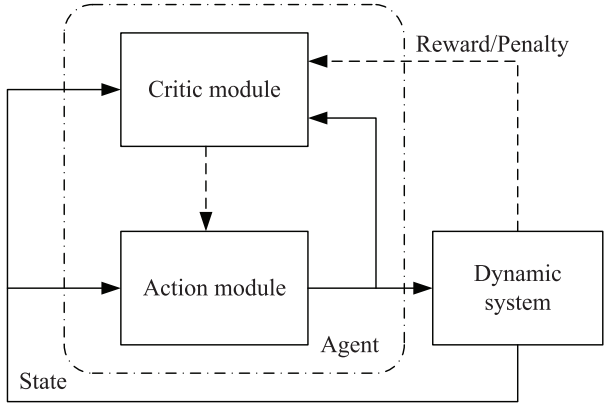

Fig. 1.1. The structure of adaptive dynamic programming.

more improvements in ADP methods have been discussed and have received lots of attention [11, 28, 39, 43, 50, 58–60, 62].

1.2.1 *Principle of adaptive dynamic programming*

Adaptive dynamic programming employs approximation structures, such as neural networks, to approximate the performance index function in dynamic programming [14, 30], such that the performance index function satisfies the HJB equation. There are generally three modules in ADP, which are model, critic and action modules, respectively. Every module can be realized by neural networks. The basic structure of ADP can be displayed by Fig. 1.1.

In Fig. 1.1, the dynamic system can be modeled by neural network. The action network is employed to approximate the optimal control law and the critic network is employed to approximate the optimal performance index function. The action and critic networks are combined as a agent, which creates a control input to the system. Then, a reward/punishing information is generated as the input of the critic network, which aims to update its weight by the HJB equation.

1.2.1.1 *Heuristic dynamic programming*

Heuristic dynamic programming (HDP) is a most basic and popular ADP method [6, 36, 54, 61]. The structure of HDP is shown in Fig. 1.2

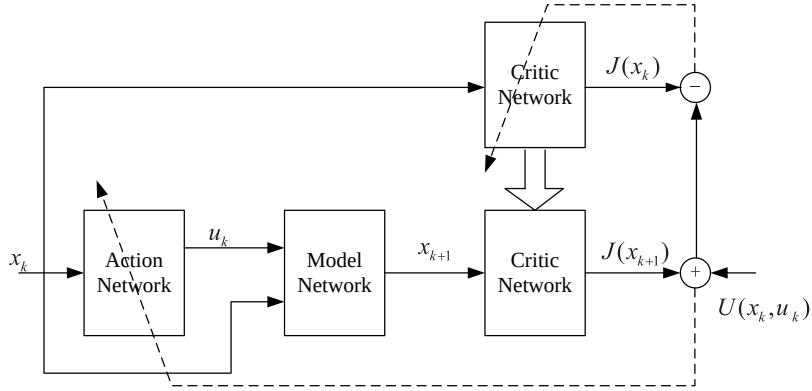

Fig. 1.2. The HDP structure diagram.

In HDP, the action network is used to approximate the control law. The model network is used to approximate the system dynamics. The critic network is used to approximate the performance index function. The performance index function in HDP is expressed as

$$J(x_k) = U(x_k, u(x_k)) + J(x_{k+1}), \tag{1.10}$$

where $u(x_k)$ is the feedback control law. The performance index functions $J(x_k)$ and $J(x_{k+1})$ are the output of the critic networks. If the weight of the critic network is defined as w, then the right-hand side of (1.10) can be written as

$$d(x_k, w) = U(x_k, u(x_k)) + J(x_{k+1}, w). \tag{1.11}$$

The weight of the critic network is regulated which aims to satisfy the following function

$$w^* = \arg\min_w \{|J(x_k, w) - d(x_k, w)|^2\}. \tag{1.12}$$

According to the principle of optimality, the optimal control law satisfies the following equation

$$\begin{aligned} \frac{\partial J^*(x_k)}{\partial u_k} &= \frac{\partial U(x_k, u_k)}{\partial u_k} + \frac{\partial J^*(x_{k+1})}{\partial u_k} \\ &= \frac{\partial U(x_k, u_k)}{\partial u_k} + \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}} \frac{\partial f(x_k, u_k)}{\partial u_k} \\ &= 0. \end{aligned} \tag{1.13}$$

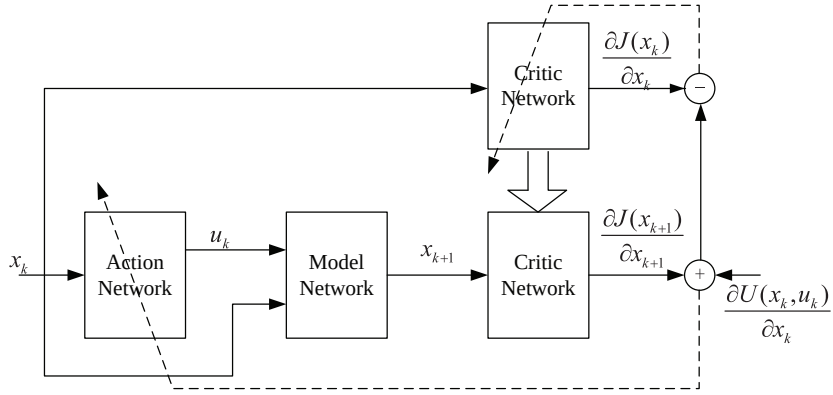

Fig. 1.3. The DHP structure diagram.

Then, we can obtain the optimal control law

$$u^* = \arg\min_{u_k} \left(\left| \frac{\partial J(x_k)}{\partial u_k} - \frac{\partial U(x_k, u_k)}{\partial u_k} - \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}} \frac{\partial f(x_k, u_k)}{\partial u_k} \right| \right), \tag{1.14}$$

where $\frac{\partial J^*(x_{k+1})}{\partial x_{k+1}}$ can be obtained by critic network.

1.2.1.2 *Dual heuristic dynamic programming*

The structure of dual heuristic dynamic programming (DHP) is shown in Fig. 1.3. There are also three neural networks in DHP structure, which are action network, model network, and critic network. The goal of action network in DHP is the same as the one in HDP. But the critic network in DHP is to approximate the derivative of the performance index function, i.e., $\frac{\partial J(x_k)}{\partial x_k}$, instead of approximating the performance index function itself [15, 44, 45]. In DHP, $\frac{\partial J(x_k)}{\partial x_k}$ is also called costate.

In DHP, the weight of the critic network is updated using the derivation of the performance index function, which is expressed as

$$\frac{\partial J(x_k)}{\partial x_k} = \frac{\partial U(x_k, u(x_k))}{\partial x_k} + \frac{\partial J(x_{k+1})}{\partial x_k}, \tag{1.15}$$

where $u(x_k)$ is the control law. The costate $\frac{\partial J(x_k)}{\partial x_k}$ and $\frac{\partial J(x_{k+1})}{\partial x_k}$ are the output of the critic network. If the weight of the critic is w, the

right-hand side of (1.15) can be expressed as

$$e(x_k, w) = \frac{\partial U(x_k, u(x_k))}{\partial x_k} + \frac{\partial J(x_{k+1}, w)}{\partial x_k}. \tag{1.16}$$

Regulate the weight of the critic network w, which minimizes the following error function

$$w^* = \arg\min_w \left\{ \left| \frac{\partial J(x_k, w)}{\partial x_k} - e(x_k, w) \right|^2 \right\} \tag{1.17}$$

to obtain the costate. According to the principle of optimality, the optimal control law satisfies

$$\begin{aligned} \frac{\partial J^*(x_k)}{\partial u_k} &= \frac{\partial U\left(x_k, u_k\right)}{\partial u_k} + \frac{\partial J^*(x_{k+1})}{\partial u_k} \\ &= \frac{\partial U(x_k, u_k)}{\partial u_k} + \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}} \frac{\partial f(x_k, u_k)}{\partial u_k}. \end{aligned} \tag{1.18}$$

Then, we can obtain the optimal control law as

$$u^*(x_k) = \arg\min_{u_k} \left(\left| \frac{\partial J(x_k)}{\partial u_k} - \frac{\partial U(x_k, u_k)}{\partial u_k} - \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}} \frac{\partial f(x_k, u_k)}{\partial u_k} \right| \right), \tag{1.19}$$

where $\frac{\partial J^*(x_{k+1})}{\partial x_{k+1}}$, i.e., the costate satisfies (1.17).

1.2.1.3 *Action dependent heuristic dynamic programming*

The principle of action dependent heuristic dynamic programming (ADHDP) is nearly the same as the HDP [22, 23, 37], where the structure of ADHDP is shown in Fig. 1.4. An obvious different between the HDP and ADHDP is the input of the critic network. In HDP, the input of the critic network is the state variable, while in ADHDP, the input of the critic network contains both the state and the control variables. In this situation, the output of the critic network is usually defined as a Q function and the ADHDP is also called as Q-learning algorithm [25, 41, 47]. The update rules of the weight of the critic and action networks are similar to the HDP and omitted here.

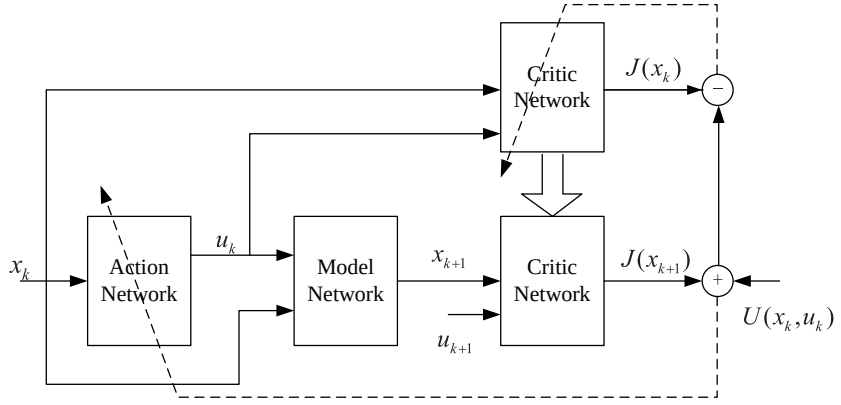

Fig. 1.4. The ADHDP structure diagram.

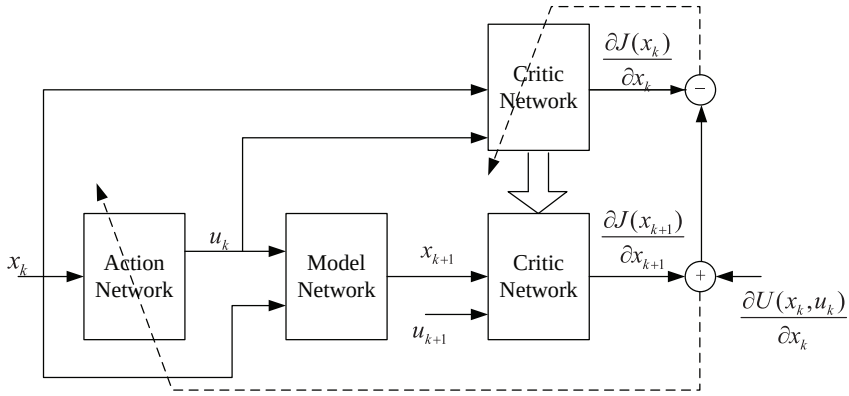

Fig. 1.5. The ADDHP structure diagram.

1.2.1.4 *Action dependent dual heuristic dynamic programming*

The principle of action dependent dual heuristic dynamic programming (ADDHP) is nearly the same as the DHP [34], where the structure of ADDHP is shown in Fig. 1.5. An obvious difference between DHP and ADDHP is the input of the critic networks. In DHP, the input of the critic network is the state variable, while in ADDHP, the input of the critic network contains both the state and the control variables. The update rules of the weight of the critic and action networks are similar to the DHP [34] and omitted here.

1.3 Iterative Forms of Adaptive Dynamic Programming

Traditional development of ADP focused on the improvement of the structure, such as HDP, DHP, GDHP, and so on. However, the properties of the ADP methods, such as the convergence of the iterative value function and the stability of the system under the control law which are obtained by the traditional ADP methods were scarcely analyzed. Iterative methods, which are convenient to analyze the properties of ADP, become more and more popular in ADP to obtain the solution of HJB equation indirectly. Value and policy iterations are two primary iterative ADP algorithms [8, 17, 40].

1.3.1 *Value iteration*

Value iteration algorithms generally start with an initial value iteration function and implement to obtain the optimal control law for discrete-time nonlinear systems. In Refs. [4, 20, 35], the iteration process of the value iteration algorithms were displayed, which is derived as follows. Starting with the zero initial value function, i.e., $V_0(x_k) \equiv 0$, $\forall x_k$, solve for the initial iterative control law $v_0(x_k)$ as

$$\begin{aligned} v_0(x_k) &= \arg\min_{u_k}\{U(x_k, u_k) + V_0(x_{k+1})\} \\ &= \arg\min_{u_k}\{U(x_k, u_k) + V_0(F(x_k, u_k))\}, \end{aligned} \tag{1.20}$$

where $V_0(x_{k+1}) = \Psi(x_{k+1})$. The value function can be updated as

$$V_1(x_k) = U(x_k, v_0(x_k)) + V_0(F(x_k, v_0(x_k))). \tag{1.21}$$

For $i = 1, 2, \ldots$, the iterative ADP algorithm will iterate between

$$\begin{aligned} v_i(x_k) &= \arg\min_{u_k}\{U(x_k, u_k) + V_i(x_{k+1})\} \\ &= \arg\min_{u_k}\{U(x_k, u_k) + V_i(F(x_k, u_k))\}, \end{aligned} \tag{1.22}$$

and

$$\begin{aligned} V_{i+1}(x_k) &= \min_{u_k}\{U(x_k, u_k) + V_i(x_{k+1})\} \\ &= U(x_k, v_i(x_k)) + V_i(F(x_k, v_i(x_k))). \end{aligned} \tag{1.23}$$

In Refs. [20, 35], it was proven that the iterative value function $V_i(x_k)$ is convergent to the optimal performance index function, as

$i \to \infty$. In Ref. [4], it was proven that the iterative value function $V_i(x_k)$ is monotonically non-decreasing and convergent to the optimum.

1.3.2 *Policy iteration*

Policy iteration for discrete-time nonlinear systems was introduced in Refs. [8, 17, 40]. In the discrete-time policy iteration algorithm, the iterative value function and control law are updated by iteration, with the iteration index i increasing from 0 to infinity. Let $v_0(x_k)$ be an arbitrary admissible control law. For $i = 0$, let $V_0(x_k)$ be the corresponding iterative value function constructed by $v_0(x_k)$, which satisfies the following generalized HJB (GHJB) equation

$$V_0(x_k) = U(x_k, v_0(x_k)) + V_0(x_{k+1}). \tag{1.24}$$

Then, the iterative control law is computed by

$$\begin{aligned} v_1(x_k) &= \arg\min_{u_k}\{U(x_k, u_k) + V_0(x_{k+1})\} \\ &= \arg\min_{u_k}\{U(x_k, u_k) + V_0(F(x_k, u_k))\}. \end{aligned} \tag{1.25}$$

For $i = 1, 2, \ldots$, the iterative value function can be constructed by $v_i(x_k)$, which satisfies the following GHJB equation

$$V_i(x_k) = U(x_k, v_i(x_k)) + V_i(F(x_k, v_i(x_k)), \tag{1.26}$$

and the iterative control law can be updated by

$$\begin{aligned} v_{i+1}(x_k) &= \arg\min_{u_k}\{U(x_k, u_k) + V_i(x_{k+1})\} \\ &= \arg\min_{u_k}\{U(x_k, u_k) + V_i(F(x_k, u_k))\}. \end{aligned} \tag{1.27}$$

In Ref. [21], the detailed process of discrete-time policy iteration was displayed and the properties of the iterative control laws and iterative value function was analyzed. It was proven that the iterative value function is monotonically non-increasing and convergent to the optimal performance index function as the iteration index i increases to the infinity.

For continuous-time systems, policy iteration algorithms are generally focused on the optimal control of affine nonlinear systems

with quadratic form utility functions [29]. Consider the following continuous-time affine nonlinear systems

$$\dot{x} = f(x) + g(x)u, \tag{1.28}$$

where f and g are system functions. The performance index function is defined as

$$\begin{aligned} J(x_0, u) &= \int_0^\infty L(x(\tau), u(\tau))\mathrm{d}\tau \\ &= \int_0^\infty (x^\mathsf{T}(\tau)\mathrm{Q}x(\tau) + u^\mathsf{T}(\tau)\mathrm{R}u(\tau))\mathrm{d}\tau, \end{aligned} \tag{1.29}$$

where Q is a positive semi-definite matrix and R is a positive definite matrix.

The iteration process of the continuous-time policy iteration algorithms is derived as follows. Let $v_0(x)$ be an arbitrary admissible control law. Choose the initial iterative value function satisfies

$$0 = x^\mathsf{T}\mathrm{Q}x + v_0^\mathsf{T}(x)\mathrm{R}v_0(x) + \nabla V_0^\mathsf{T}(x)(f(x) + g(x)v_0(x)), \tag{1.30}$$

where $\nabla V_0(x) = \frac{\mathrm{d}V_0(x)}{\mathrm{d}x}$. Then, for any $i = 1, 2, \ldots$, the iterative control law is updated by

$$v_i(x) = -\frac{1}{2}\mathrm{R}^{-1}g^\mathsf{T}(x)\nabla V_{i-1}(x) \tag{1.31}$$

and the iterative value function $V_i(x)$ is updated to satisfy the following equation

$$0 = x^\mathsf{T}\mathrm{Q}x + {v_i}^\mathsf{T}(x)\mathrm{R}v_i(x) + \nabla V_i^\mathsf{T}(x)(f(x) + g(x)v_i(x)). \tag{1.32}$$

In Ref. [29], it was proven that the iterative value function $V_i(x)$ is monotonically non-increasing and convergent to the optimal performance index function as the iteration index i increases to the infinity.

1.4 About This Book

This book presents the recent results of iterative adaptive dynamic programming methods. It is composed of 13 chapters which cover

most of the hot research areas of iterative adaptive dynamic programming and are divided into three parts. Part I concerns the systems with single controller, including four chapters from Chapters 2 to 5. Part II concerns the systems with multiple controllers, including four chapters from Chapters 6 to 9. Part III concerns applications in residential energy systems, including four chapters from Chapters 10 to 13.

In Chapter 1, the principle and the history of adaptive dynamic programming are introduced, including the basic forms of adaptive dynamic programming. The review begins with the origin of adaptive dynamic programming and describes the basic structures in detail.

In Chapter 2, a new local iterative adaptive dynamic programming algorithm is developed to solve infinite horizon optimal control problems for discrete-time nonlinear systems. The developed local iterative adaptive dynamic programming algorithm permits an arbitrary positive semi-definite function to initialize the algorithm. Employing a state-dependent learning rate function, for the first time, the iterative value function and iterative control law can be updated by a subset of the state space instead of the whole state space, which effectively relaxes the computation burden. A new analysis method of the convergence property is developed to prove that the iterative value functions will converge to the optimum under some mild constraints. Monotonicity of the local iterative adaptive dynamic programming algorithm is presented, which shows that under some special conditions of the initial value function and the learning rate function, the iterative value function can monotonically converge to the optimum. Finally, three simulation examples and comparisons are given to illustrate the performance of the developed algorithm.

In Chapter 3, a novel local iterative adaptive dynamic programming algorithm is presented to solve infinite horizon optimal control problems for discrete-time nonlinear systems. A state-dependent learning rate function is introduced into iterative adaptive dynamic programming algorithms, such that the iterative value functions and iterative control laws are updated in a given subset of the state space in each iteration, instead of the whole state space. Admissibility properties of iterative control laws are analyzed for the developed local iterative adaptive dynamic programming algorithm. New termination criteria including convergence and admissibility criteria

are established, which make the iterative local adaptive dynamic programming algorithm terminated with an admissible approximate optimal control law. Finally, simulation results are given to illustrate the performance of the developed algorithm.

In Chapter 4, a new policy iteration adaptive dynamic programming algorithm is developed to solve infinite horizon optimal control problems for discrete-time nonlinear systems. In each iteration of the developed discrete-time local policy iteration algorithm, the iterative value function and iterative control law can be updated in a subset of the state space, instead of updated in the whole state space. Convergence properties are presented to show that the iterative value function is monotonically non-increasing and converges to the optimum under some mild constraints. The admissibility of the iterative control law is proven, which shows that the control system can be stabilized under any of the iterative control laws, even if the iterative control law is updated in a subset of the state space. Finally, two simulation examples are given to illustrate the performance of the developed method.

In Chapter 5, a novel policy iteration algorithm, called "continuous-time time-varying policy iteration algorithm", is presented to obtain the optimal control laws for infinite horizon continuous-time time-varying nonlinear systems. The adaptive dynamic programming technique is utilized to get the iterative control laws for the optimization of the performance index function. The properties of the continuous-time time-varying policy iteration algorithm are analyzed. Monotonicity, convergence, and optimality of the iterative value function have been analyzed, and the iterative value function can be proven to monotonically converge to the optimal solution of the Hamilton–Jacobi–Bellman equation. Furthermore, the iterative control law is guaranteed to be admissible to stabilize the nonlinear systems. In the implementation of the presented continuous-time time-varying policy algorithm, the approximate iterative control laws and iterative value function are obtained by neural networks. Finally, numerical results are given to verify the effectiveness of the presented method.

In Chapter 6, a new adaptive dynamic programming algorithm, called "iterative zero-sum adaptive dynamic programming algorithm," is developed to solve infinite horizon optimal control problems for discrete-time two-player zero-sum games of nonlinear

systems. The present iterative zero-sum adaptive dynamic programming algorithm permits arbitrary positive semi-definite functions to initialize the upper and lower iterations. A novel convergence analysis is developed to guarantee that the upper and lower iterative value functions converge to the upper and lower optimums, respectively. When the saddle-point equilibrium exists, it is emphasized that both the upper and lower iterative value functions are proven to converge to the optimal solution of the zero-sum game, where the existence criteria of the saddle-point equilibrium are not required. If the saddle-point equilibrium does not exist, the upper and lower optimal performance index functions are obtained, respectively, where the upper and lower performance index functions are proven not to be equivalent. Finally, two simulation examples and comparisons are given to illustrate the performance of the present method.

In Chapter 7, an online adaptive optimal control algorithm based on adaptive dynamic programming is developed to solve the multi-player non-zero-sum game for discrete-time unknown nonlinear systems. First, a model-free coupled globalized dual heuristic dynamic programming structure is designed to solve the multi-player non-zero-sum game problem, in which there is no model network or identifier. Second, in order to relax the requirement of the systems dynamics, an online adaptive learning algorithm is developed to solve the Hamilton–Jacobi equation using the system states of two adjacent time-steps. Third, a series of critic networks and action networks are used to approximate the value functions and optimal policies for all players, respectively. All the neural network weights are updated online based on real-time system states. Fourth, the uniform ultimate boundedness analysis of the neural network approximation errors is proved based on Lyapunov approach. Finally, simulation results are given to demonstrate the effectiveness of the developed scheme.

In Chapter 8, a novel distributed policy iteration algorithm is established for the infinite horizon optimal control problems of continuous-time nonlinear systems. In each iteration of the developed distributed policy iteration algorithm, only one controller's control law is updated and other controllers remain unchanged. The main contribution of the present algorithm is to improve the iterative control law one by one, instead of updating all the control laws in each iteration of the traditional policy iteration algorithms, which effectively releases the computation burden in each

iteration. The properties of distributed policy iteration algorithm for continuous-time nonlinear systems are analyzed. Admissibility of the present methods has also been analyzed. Besides, monotonicity, convergence, and optimality have been discussed, which show that the iterative value function is non-increasingly convergent to the solution of the Hamilton–Jacobi–Bellman equation. Finally, numerical simulations are conducted to illustrate the effectiveness of the proposed method.

In Chapter 9, a novel data-based fault-tolerant control method is presented for multicontroller linear systems via distributed policy iteration. Traditional fault-tolerant control methods based on policy iteration update all the control laws in each iteration, which causes huge computational burden in the situation of high-dimension control laws. In order to solve this problem, a novel distributed policy iteration method is presented, where only one iterative control law is updated in each iteration, to realize the fault-tolerant control of multicontroller linear systems. First, the model-based fault-tolerant control via distributed policy iteration is provided. Based on the model-based method, a data-based fault-tolerant control method is presented. Finally, numerical experiments are given to show the performance of the presented method.

In Chapter 10, a novel iterative Q-learning method called "dual iterative Q-learning algorithm" is developed to solve the optimal battery management and control problem in smart residential environments. The main idea is to use adaptive dynamic programming technique to obtain the optimal battery management and control scheme iteratively for residential energy systems. In the developed dual iterative Q-learning algorithm, two iterations are introduced, which are respectively global and local iterations, where local iteration minimizes the total cost of power load in each period and the global iteration makes the iterative Q-function converge to the optimum. Based on the dual iterative Q-learning algorithm, the convergence property of iterative Q-learning method for the optimal battery management and control problem is proven for the first time, which guarantees that both the iterative Q-function and the iterative control law reach the optimum. Neural networks are introduced to implement the developed dual iterative Q-learning algorithm. Finally, numerical results are given to illustrate the performance of the developed algorithm.

In Chapter 11, a novel mixed iterative adaptive dynamic programming algorithm is developed to solve the optimal battery energy management and control problem in microgrid systems. Based on the data of the load and electricity rate, two iterations are constructed, which are P-iteration and V-iteration, respectively. The V-iteration is implemented based on value iteration, which aims to obtain the iterative control law sequence in each period. The P-iteration is implemented based on policy iteration, which updates the iterative value function according to the iterative control law sequence. Properties of the developed mixed iterative adaptive dynamic programming algorithm are analyzed. It is shown that the iterative value function is monotonically non-increasing and converges to the solution of the Hamilton–Jacobi–Bellman equation. In each iteration, it is proven that the performance index function is finite under the iterative control law sequence. Finally, numerical results and comparisons are given to illustrate the performance of the developed algorithm.

In Chapter 12, a novel error-tolerant iterative adaptive dynamic programming algorithm is developed to solve optimal battery control and management problems in smart home environments with renewable energy. The algorithm can be initialized by an arbitrary positive semi-definite function. Based on the quasi-periodic data of the electricity rate, the home load demand, and the renewable energy, the error-tolerant iterative adaptive dynamic programming algorithm can be implemented. Considering the differences of the data in different periods, the developed error-tolerant iterative adaptive dynamic programming algorithm can adaptively regulate the discount factor in each iteration to make the iterative value function converge. A new analysis method is developed to guarantee the iterative value function to converge to a finite neighborhood of the optimal performance index function, in spite of the differences of the electricity rate, the home load demand, and the renewable energy in different periods. Numerical results and comparisons are given to illustrate the performance of the developed algorithm.

In Chapter 13, a new generalized actor-critic learning optimal control method is presented. It aims at the optimal energy control and management for smart home systems, which is expected to minimize the consumption cost for home users. The main contribution of the developed method is to establish a common iteration structure for

both value and policy iterations in adaptive dynamic programming based on a control law sequence in each iteration for periodic time-varying systems, instead of a single control law. The monotonicity, convergence, and optimality of the iterative value function for the generalized actor-critic learning optimal control method are proven. Finally, numerical results and comparisons are displayed to show the superiority of the developed method.

References

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

[2] Al-Tamimi, A., Abu-Khalaf, M. and Lewis, F.L. (2007). Adaptive critic designs for discrete-time zero-sum games with application to H_∞ control. *IEEE Transactions Systems, Man, and Cybernetics-Part B: Cybernetics* 37, 240–247.

[3] Al-Tamimi, A. and Lewis, F.L. (2007). Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof. In: *Proceedings of the IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pp. 38–43.

[4] Al-tamimi, A., Lewis, F.L. and Abu-khalaf, M. (2008). Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 38, 943–949.

[5] Alessandri, A., Gaggero, M. and Zoppoli, R. (2012). Feedback optimal control of distributed parameter systems by using finite-dimensional approximation schemes. *IEEE Transactions on Neural Networks and Learning Systems* 23(6), 984–996.

[6] Beard, R. (1995). Improving the closed-loop performance of nonlinear systems. PhD Thesis, Rensselaer Polytechnic Institute, Troy, NY.

[7] Bellman, R.E. (1957). *Dynamic Programming*. Princeton University Press, Princeton, New Jersey.

[8] Bertsekas, D.P. (2007). Dynamic Programming and Optimal Control, (3rd edn.), Athena Scientific, Belmont, MA.

[9] Bertsekas, D.P. and Tsitsiklis, J.N. (1996). *Neuro-dynamic Programming*. Athena Scientific, Belmont, MA.

[10] Enns, R. and Si, J. (2003). Helicopter trimming and tracking control using direct neural dynamic programming. *IEEE Transactions on Neural Networks* 14(4), 929–939.

[11] Heydari, A. and Balakrishnan, S.N. (2014). Optimal switching and control of nonlinear switching systems using approximate dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems* 25(6), 1106–1117.

[12] Jiang, Y. and Jiang, Z.P. (2013). Robust adaptive dynamic programming with an application to power systems. *IEEE Transactions on Neural Networks and Learning Systems* 24(7), 1150– 1156.

[13] Jiang, Y. and Jiang, Z.P. (2012). Robust adaptive dynamic programming for large-scale systems with an application to multimachine power systems. *IEEE Transactions on Circuits and Systems II: Express Briefs* 59(10), 693–697.

[14] Lendaris, G.G. (2008). Higher level application of ADP: A next phase for the control field. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics* 38(4), 901–912.

[15] Lendaris, G.G. and Paintz, C. (1997). Training strategies for critic and action neural networks in dual heuristic programming method. In: *Proceedings of International Joint Conference on Neural Networks* pp. 712–717.

[16] Lewis, F.L. and Vrabie, D. (2009). Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine* 9(3), 32–50.

[17] Lewis, F.L., Vrabie, D. and Vamvoudakis, K.G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine* 32(6), 76–105.

[18] Li, X., Shou, B. and Ralescu, D. (2014). Train rescheduling with stochastic recovery time: A new truck-backup approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44(9), 1216–1233.

[19] Li, Z., Xiao, S., Ge, S.S. and Su, H. (2016). Constrained multilegged robot system modeling and fuzzy control with uncertain kinematics and dynamics incorporating foot force optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46(1), 1–15.

[20] Lincoln, B. and Rantzer, A. (2006). Relaxing dynamic programming. *IEEE Transactions on Automatic Control* 51(8), 1249–1260.

[21] Liu, D. and Wei, Q. (2014). Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems* 25(3), 621–634.

[22] Liu, D., Xiong, X. and Zhang, Y. (2001). Action-dependent adaptive critic designs. In: *Proceedings of the INNS-IEEE International Joint Conference on Neural Networks*, Washington DC, USA, pp. 990–995.

[23] Liu, D. and Zhang, H. (2005). A neural dynamic programming approach for learning control of failure avoidance problems. *International Journal of Intelligence Control and Systems* 10(1), 21–32.

[24] Liu, D., Zhang. Y. and Zhang, H. (2005). A self-learning call admission control scheme for CDMA cellular networks. *IEEE Transactions on Neural Networks* 16(5), 1219–1228.

[25] Liu, D.R. (2005). Approximate dynamic programming for self-learning control. *Acta Automatica Sinica* 31(1), 13–18.

[26] Luo, B. and Wu, H. (2012). Approximate optimal control design for nonlinear one-dimensional parabolic PDE systems using empirical eigenfunctions and neural network. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 42(6), 1538–1549.

[27] Modares, H., Lewis, F.L. and Naghibi-Sistani, M.B. (2014). Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. *Automatica* 50, 193–202.

[28] Modares, H., Lewis, F.L. and Naghibi-Sistani, M.B. (2013). Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 24(10), 1513–1525.

[29] Murray, J.J., Cox, C.J., Lendaris, G.G. and Saeks, R. (2002). Adaptive dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews* 32(2), 140–153.

[30] Narendra, K.S. and Lewis, F.L. (2001). Special issue on neural network feedback control. *Automatica* 37(8), 172–179.

[31] Ni, Z., He, H., Zhao, D., Xu, X. and Prokhorov, D.V. (2015). GrDHP: A general utility function representation for dual heuristic dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems* 26(3), 614–627.

[32] Ni, Z. and He, H. (2013). Heuristic dynamic programming with internal goal representation. *Soft Computing* 17(11), 2101–2108.

[33] Powell, W.B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* Wiley, Hoboken, NJ.

[34] Prokhorov, D.V. and Wunsch, D.C. (1997). Adaptive critic designs. *IEEE Transactions on Neural Networks* 8(5), 997–1007.

[35] Rantzer, A. (2006). Relaxed dynamic programming in switching systems. *IET Control Theory and Applications* 153(5), 567–574.

[36] Seiffertt, J., Sanyal, S. and Wunsch, D.C. (2008). Hamilton-Jacobi-Bellman equations and approximate dynamic programming on time scales. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 38(4), 918–923.

[37] Si, J. and Wang, Y.T. (2001). On-line learning control by association and reinforcement. *IEEE Transactions on Neural Networks* 12(3), 264–275.

[38] Song, Z. and Kusiak, A. (2010). Multiobjective optimization of temporal processes. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 40(3), 845–856.

[39] Song,R., Lewis, F.L., Wei, Q. and Zhang, H. (2016). Off-policy actor-critic structure for optimal control of unknown systems with disturbances. *IEEE Transactions on Cybernetics* 46(5), 1041–1050.

[40] Sutton, R.S. and Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA.

[41] Timmer, S. and Riedmiller, M. (2007). Fitted Q Iteration with CMACs. In: P*roceedings of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning* pp. 1–8.

[42] Vales-Alonso, J., Chaves-Dieguez, D., Lopez-Matencio, P., Alcaraz, J.J., Parrado-Garcia, F.J. and Gonzalez-Castano, F.J. (2015). SAETA: A smart coaching assistant for professional volleyball training. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45(8)1, 138–1150.

[43] Vamvoudakis, K.G., Lewis, F.L. and Hudas, G.R. (2012) Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality. *Automatica* 48(8), 1598–1611.

[44] Venayagamoorthy, G.K., Harley, R.G. and Wunsch, D.C. (2002). Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator. *IEEE Transactions on Neural Networks* 13(3), 764–773.

[45] Venayagamoorthy, G.K. and Wunsch, D.C. (2000). Adaptive critic based neurocontroller for turbogenerators with global dual heuristic programming. In: *Proceedings of the Power Engineering Society Winter Meeting*, pp. 291–294.

[46] Wang, F.Y., Jin, N., Liu, D. and Wei, Q. (2011). Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ϵ-error bound. *IEEE Transactions on Neural Networks* 22(1), 24–36.

[47] Watkins, C. (1989). *Learning from Delayed Rewards*. PhD Thesis, Cambridge University, Cambridge, England.

[48] Wei, Q. Lewis, F.L., Sun, Q., Yan, P. and Song, R. (2016). Discrete-time deterministic Q-learning: A novel convergence analysis. *IEEE Transactions on Cybernetics*, article in press. doi:10.1109/TCYB.2016.25429230.

[49] Wei, Q. and Liu, D. (2014). A novel iterative θ-adaptive dynamic programming for discrete-time nonlinear systems. *IEEE Transactions on Automation Science and Engineering* 11(4), 1176–1190.

[50] Wei, Q., Liu, D., Lin, Q. and Song, R. (2016). Discrete-time optimal control via local policy iteration adaptive dynamic programming. *IEEE Transactions on Cybernetics*, article in press. doi: 10.1109/TCYB.2016.2586082.

[51] Wei, Q., Song, R. and Yan, P. (2016). Data-driven zero-sum neuro-optimal control for a class of continuous-time unknown nonlinear systems with disturbance using ADP. *IEEE Transactions on Neural Networks and Learning Systems* 27(2), 444–458.

[52] Wei, Q., Zhang, H. and Dai, J. (2009). Model-free multiobjective approximate dynamic programming for discrete-time nonlinear systems with general performance index functions. *Neurocomputing* 72(7-9), 1839–1848.

[53] Werbos, P.J. (1977). Advanced forecasting methods for global crisis warning and models of intelligence. *General Systems Yearbook* 22, 25–38.

[54] Werbos, P.J. (1990). Consistency of HDP applied to a simple reinforcement learning problem. *Neural Networks* 3(2), 179–189.

[55] Werbos, P.J. (1991). A menu of designs for reinforcement learning over time. In: Miller, W.T., Sutton, R.S. and Werbos, P.J. (eds.), *Neural Networks for Control*, Cambridge, MIT Press, pp. 67–95.

[56] Werbos, P.J. (1992). Approximate dynamic programming for real-time control and neural modeling. In: White, D.A. and Sofge, D.A. (eds.), *Handbook of Intelligent Control*, Van Nostrand Reinhold, New York.

[57] Yang, C., Li, Z. and Li, J. (2013). Trajectory planning and optimized adaptive control for a class of wheeled inverted pendulum vehicle models. *IEEE Transactions on Cybernetics* 43(1), 24–36.

[58] Zhang, H., Zhang, J., Yang, G. and Luo, Y. (2015). Leader-based optimal coordination control for the consensus problem of multi-agent differential games via fuzzy adaptive dynamic programming. *IEEE Transactions on Fuzzy Systems* 23(1), 152–163.

[59] Zhang, H., Qing, C. and Luo, Y. (2014). Neural-network-based constrained optimal control scheme for discrete-time switched nonlinear system using dual heuristic programming. *IEEE Transactions on Automation Science and Engineering* 11(3), 839–849.

[60] Zhao, Q., Xu, H. and Jagannathan, S. (2015). Neural network-based finite-horizon optimal control of uncertain affine nonlinear discrete-time systems. *IEEE Transactions on Neural Networks and Learning Systems* 26(3), 486–499.

[61] Zhao, Y., Patek, S.D. and Beling, P.A. (2008). Decentralized Bayesian search using approximate dynamic programming methods. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics* 38(4), 970–975.

[62] Zhong, X., He, H., Zhang, H. and Wang, Z. (2014). Optimal control for unknown discrete-time nonlinear Markov jump systems using adaptive dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems* 25(12), 2141–2155.

This page intentionally left blank

Chapter 2

Discrete-Time Local Value Iterative Adaptive Dynamic Programming

2.1 Introduction

Dynamic programming often suffers from the curse of dimensionality, which is primarily due to the backward-in-time mechanism. To avoid the difficulty, based on function approximators, such as neural networks, adaptive dynamic programming is an effective brain-like method to solve optimal control problems forward-in-time [9]. Recently, the study on adaptive dynamic programming and related fields has gained much attention from various scholars [1, 8, 12, 17, 19]. Iterative methods are widely used in adaptive dynamic programming to obtain solutions of the Bellman equation indirectly. Policy and value iterations are two primary iterative adaptive dynamic programming algorithms. Policy iteration algorithms for optimal control of continuous-time systems with continuous states and action spaces were first given in Ref. [15]. In Ref. [21], the optimal control law for multiple actor-critic structures was effectively obtained using shunting inhibitory artificial neural network. In Ref. [29], the multiagent optimal control was obtained using fuzzy approximation structures. Value iteration algorithm for optimal control of discrete-time nonlinear systems was presented in Ref. [4]. In Refs. [2, 10], the value iteration algorithm for deterministic discrete-time affine nonlinear systems was studied. It was proven that the iterative value function is nondecreasing and bounded, and hence converges to the optimum as the iteration index increases to

infinity. Based on the framework of policy and value iteration algorithms, more investigations on iterative ADP algorithms have been developed [5, 11, 13, 20, 22–25, 27, 28].

Existing global iterative ADP algorithms require updating over the entire state space in each iteration [2, 3, 7, 9, 16], posing challenges for real-world control systems that operate within a subset of the state space during each iteration period.This limitation, compounded by the computational demands of approximation structures like neural networks, restricts the practical implementation of traditional global iterative ADP algorithms. To overcome these challenges, we propose a new approach, termed the "local iterative ADP algorithm", designed to update within a specified subset of the state space. This algorithm addresses the shortcomings of global methods and shows promise for practical applications.

In this chapter, the local iterative ADP algorithm is introduced to solve infinite horizon optimal control problems for discrete-time nonlinear systems. The algorithm allows initialization with any positive semi-definite function, and the use of a state-dependent learning rate enables updates in a specific state space subset. The convergence properties of the algorithm are proven, demonstrating its superiority over traditional global iterative ADP algorithms. Simulation results show the efficacy of the proposed method.

2.2 Preliminaries and Assumptions

In this chapter, we will study the following discrete-time deterministic nonlinear system

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \ldots, \tag{2.1}$$

where $x_k \in \mathbb{R}^n$ is the state vector and $u_k \in \mathbb{R}^m$ is the control vector. Let x_0 be the initial state and $F(x_k, u_k)$ be the system function. For convenience of analysis, results of this chapter are based on the following assumption.

Assumption 2.1. The system (2.1) is controllable on a compact set $\Omega_x \subset \mathbb{R}^n$ containing the origin, and the function $F(x_k, u_k)$ is Lipschitz continuous on Ω_x; the system state $x_k = 0$ is an equilibrium

state of system (2.1) under the control $u_k = 0$, i.e., $F(0,0) = 0$; the feedback control $u_k = u(x_k)$ satisfies $u_k = u(x_k) = 0$ for $x_k = 0$.

Let $\mathcal{U}$ denote control space and let $u \in \mathcal{U}$ be the control law. Then, the infinite-horizon performance index function for state x_0 can be defined as

$$J(x_0, u) = \sum_{k=0}^{\infty} U(x_k, u_k), \tag{2.2}$$

where the utility function $U(x_k, u_k)$ is a positive definite function for x_k and u_k.

Then, the optimal performance index function can be defined as

$$J^*(x_k) = \min_u \{J(x_k, u) \colon u \in \mathcal{U}\}. \tag{2.3}$$

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

$$J^*(x_k) = \min_{u_k}\{U(x_k, u_k) + J^*(F(x_k, u_k))\}. \tag{2.4}$$

Define the law of optimal control as

$$u^*(x_k) = \arg\min_{u_k} \{U(x_k, u_k) + J^*(F(x_k, u_k))\}. \tag{2.5}$$

Generally, the optimal performance index function $J^*(x_k)$ is a non-analytical nonlinear function. It is nearly impossible to obtain $J^*(x_k)$ for all $x_k \in \Omega_x$ by solving the HJB equation directly. In this chapter, a new local iterative ADP algorithm will be developed to obtain the optimal control law iteratively with new property analysis.

2.3 Discrete-Time Local Iterative ADP Algorithm

In this section, a new discrete-time local iterative ADP algorithm is developed to obtain the optimal control law for the nonlinear system (2.1). The convergence, optimality, and monotonicity properties of the developed local iterative ADP algorithm will be presented. A new termination criterion for the local iterative ADP algorithm will be established.

2.3.1 *Derivation of the discrete-time local iterative ADP algorithm*

In the developed discrete-time local iterative ADP algorithm, the value function and control law are updated by iteration, with the iteration index i increasing from 0 to infinity. For $i = 0, 1, \ldots$, define $\{\mathscr{L}_x^i\}$ as a sequence of state sets, which satisfies $\mathscr{L}_x^i \subseteq \Omega_x$. Thus, for $i = 0, 1, \ldots, \mathscr{L}_x^i$ is a subset of the state space Ω_x.

For all $x_k \in \Omega_x$, let the initial function $\Psi(x_k)$ be an arbitrary positive semi-definite function. Let the initial value function be expressed by

$$V_0(x_k) = \Psi(x_k). \tag{2.6}$$

For all $x_k \in \mathscr{L}_x^0$, the local iterative control law $v_0(x_k)$ is computed as

$$\begin{aligned} v_0(x_k) &= \arg\min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\} \\ &= \arg\min_{u_k} \{U(x_k, u_k) + \Psi(F(x_k, u_k))\}, \end{aligned} \tag{2.7}$$

and let $v_0(x_k) = 0$, for all $x_k \in \Omega_x \backslash \mathscr{L}_x^0$. For all $x_k \in \mathscr{L}_x^0$, the local iterative value function is updated as

$$\begin{aligned} \mathcal{V}_1(x_k) &= \min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\} \\ &= U(x_k, v_0(x_k)) + V_0(F(x_k, v_0(x_k))), \end{aligned} \tag{2.8}$$

and let $\mathcal{V}_1(x_k) = \Psi(x_k)$, for all $x_k \in \Omega_x \backslash \mathscr{L}_x^0$. Define $\alpha_0(x_k)$ as a state-dependent learning rate function. For all $x_k \in \Omega_x$, let the learning rate function $\alpha_0(x_k)$ be a scalar function, which satisfies

$$\begin{cases} 0 < \alpha_0(x_k) \leq 1, & \forall x_k \in \mathscr{L}_x^0, \\ \alpha_0(x_k) = 0, & \forall x_k \in \Omega_x \backslash \mathscr{L}_x^0. \end{cases} \tag{2.9}$$

Then, for all $x_k \in \Omega_x$, the global iterative value function is defined as

$$V_1(x_k) = (1 - \alpha_0(x_k)) V_0(x_k) + \alpha_0(x_k) \mathcal{V}_1(x_k). \tag{2.10}$$

For $i = 1, 2, \ldots$, for all $x_k \in \mathscr{L}_x^i$, the iterative control law $v_i(x_k)$ can be computed as

$$\begin{aligned} v_i(x_k) &= \arg\min_{u_k}\{U(x_k, u_k) + V_i(x_{k+1})\} \\ &= \arg\min_{u_k}\{U(x_k, u_k) + V_i(F(x_k, u_k))\}, \end{aligned} \tag{2.11}$$

and for all $x_k \in \Omega_x \backslash \mathscr{L}_x^i$, let $v_i(x_k) = v_{i-1}(x_k)$. For all $x_k \in \mathscr{L}_x^i$, the local iterative value function can be updated as

$$\begin{aligned} \mathcal{V}_{i+1}(x_k) &= \min_{u_k}\{U(x_k, u_k) + V_i(x_{k+1})\} \\ &= U(x_k, v_i(x_k)) + V_i(F(x_k, v_i(x_k))), \end{aligned} \tag{2.12}$$

and for all $x_k \in \Omega_x \backslash \mathscr{L}_x^i$, let $\mathcal{V}_{i+1}(x_k) = V_i(x_k)$. Define $\{\alpha_i(x_k)\}$, $i = 1, 2, \ldots$, as a sequence of learning rate functions. For $i = 1, 2, \ldots$ and $\forall x_k \in \Omega_x$, let the learning rate function $\alpha_i(x_k)$ be a scalar function, which satisfies

$$\begin{cases} 0 < \alpha_i(x_k) \leq 1, & \forall x_k \in \mathscr{L}_x^i, \\ \alpha_i(x_k) = 0, & \forall x_k \in \Omega_x \backslash \mathscr{L}_x^i. \end{cases} \tag{2.13}$$

Then, for all $x_k \in \Omega_x$, the global iterative value function is defined as

$$V_{i+1}(x_k) = (1 - \alpha_i(x_k))V_i(x_k) + \alpha_i(x_k)\mathcal{V}_{i+1}(x_k). \tag{2.14}$$

Remark 2.1. For the developed local iterative ADP algorithm (2.6)–(2.14), in each iteration, we can see that the iterative value function and iterative control law are only updated in the given state subset $\mathscr{L}_x^i$, $i = 0, 1, \ldots$, instead of the whole state space. We emphasize that this is an inherent difference from the traditional global iterative ADP algorithms [2, 3, 7, 9, 14, 16], where the learning rate function in traditional global iterative ADP algorithms is generally chosen as $\alpha_i(x_k) \equiv 1$, $i = 0, 1, \ldots, \forall x_k \in \Omega_x$. In global iterative ADP algorithms, the state data for the whole state space are required to update the iterative value function and iterative control law. If the state data for the whole state space are not known in a certain iteration, then the traditional global iterative ADP algorithms cannot continue implementing. On the other

hand, for nonlinear systems, especially high dimensional nonlinear systems, the neural network implementation of global iterative ADP algorithms is generally difficult. It lies in the huge data for high dimensional state space, which makes it difficult to train the neural networks. For the local iterative ADP algorithm (2.6)–(2.14), as the iterative value function and iterative control law are updated within a given state subset, the computation burden of the local iterative control law can be much relaxed and hence the developed local iterative ADP algorithm possesses more potential for applications.

2.3.2 *Property analysis*

In this subsection, the convergence and optimality properties of the local iterative ADP algorithm will be discussed.Before the main theorem, the following lemmas are necessary.

Lemma 2.1. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let iterative value function $V_i(x_k)$ and the iterative control law $v_i(x_k)$ be obtained by* (2.6)–(2.14). *We have*

(i) *for $i = 0, 1, \ldots, V_i(x_k)$ is positive semi-definite for x_k;*
(ii) *for $i = 0, 1, \ldots, V_i(x_k)$ is positive definite for x_k if $\Psi(x_k)$ is positive definite;*
(iii) *if $V_i(x_k)$, $i = 0, 1, \ldots,$ is positive definite for x_k, then $V_j(x_k)$ is positive definite for all $j \geq i$;*
(iv) *if $0 < \alpha_i(x_k) \leq 1$, $\forall x_k \neq 0$, and $\mathscr{L}_x^i = \Omega_x$, then for all $j = 0, 1, \ldots,$ which satisfies $j > i$, $V_j(x_k)$ is positive definite for x_k.*

Proof. For $i = 0$, $V_1(x_k)$ can be expressed by (2.10). Let $x_k = 0$. According to Assumption 2.1, we have $v_0(x_k) = 0$ and $x_{k+1} = 0$. As $V_0(x_k) = \Psi(x_k)$ is positive semi-definite for x_k and $U(x_k, u_k)$ is positive definite for x_k, u_k, we have $V_1(x_k) = 0$ for $x_k = 0$. On the other hand, for $x_k \neq 0$, we can get $V_1(x_k) \geq 0$, which means $V_1(x_k)$ is positive semi-definite. According to mathematical induction, we can obtain that for all $i = 0, 1, \ldots, V_i(x_k)$ is positive semi-definite. Lemma 2.1(i) is proven.

Lemma 2.1 (ii) and (iii) can be easily derived by the proof of Lemma 2.1(ii).

According to Lemma 2.1(i), for all $i = 0, 1, \ldots$, $V_i(x_k)$ is positive semi-definite. If for all $x_k \neq 0$, $\alpha_i(x_k) > 0$, then according to (2.14), we have $V_{i+1}(x_k) > 0$ for $x_k \neq 0$, as $U(x_k, u_k)$ is positive definite. According to mathematical induction, we can prove the for all $j > i$, $V_j(x_k)$ is positive definite, which proves Lemma 2.1(iv). □

Then, we can derive the convergence property of the local iterative ADP algorithm.

Theorem 2.1. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let the iterative control law $v_i(x_k)$ and iterative value function $V_i(x_k)$ be obtained by* (2.6)–(2.14). *If for any $i = 0, 1, \ldots$ and $\forall\, x_k \in \Omega_x$, the learning rate $\alpha_i(x_k)$ satisfies* (2.9), (2.13) *and*

$$\sum_{i=0}^{\infty} \alpha_i(x_k) = \infty, \quad \forall x_k \neq 0, \tag{2.15}$$

then the iterative value function $V_i(x_k)$ converges to the optimal performance index function $J^*(x_k)$, as $i \to \infty$.

Proof. The statement can be proven in three steps.

First, according to Assumption 2.1 and the definition of the optimal performance index function $J^*(x_k)$ in (2.3), we have $J^*(x_k)$ is a positive definite function for x_k. Let ϵ be an arbitrary nonnegative number. Define a new set Ω_ϵ as $\Omega_\epsilon = \{x_k | x_k \in \Omega_x \text{ and } \|x_k\| \leq \epsilon\}$. According to Lemma 2.1, for any $i = 0, 1, \ldots$, $V_i(x_k) = J^*(x_k) = 0$ for $\epsilon = 0$.

Second, for an arbitrary $\epsilon > 0$ and $\forall\, x_k \in \Omega_x \backslash \Omega_\epsilon$, let $\underline{\delta}$, $\overline{\delta}$, and ψ be constants, such that $0 \leq \underline{\delta} \leq 1 \leq \overline{\delta} < \infty$ and $0 < \psi < \infty$, which make the following inequalities $\underline{\delta} J^*(x_k) \leq V_0(x_k) \leq \bar{\delta} J^*(x_k)$, and $\psi U(x_k, u_k) \geq J^*(x_{k+1})$, hold uniformly. Then, we will prove that for $i = 0, 1, \ldots$, the iterative value function $V_i(x_k)$ satisfies

$$\begin{aligned} &\left(1 + \left(\prod_{\ell=0}^{i-1} \left(1 - \frac{\alpha_\ell(x_k)}{\psi + 1}\right)\right)(\underline{\delta} - 1)\right) J^*(x_k) \leq V_i(x_k) \\ &\quad \leq \left(1 + \left(\prod_{\ell=0}^{i-1} \left(1 - \frac{\alpha_\ell(x_k)}{\psi + 1}\right)\right)(\bar{\delta} - 1)\right) J^*(x_k), \end{aligned} \tag{2.16}$$

where we define $\prod_j^i(\cdot) = 1$ for $j > i$.

Obviously, (2.16) holds for $i = 0$. For $i = 1$, we have

$$\begin{aligned} V_1(x_k) &= (1-\alpha_0(x_k))V_0(x_k) + \alpha_0(x_k)\min_{u_k}(U(x_k,u_k)+V_0(x_{k+1})) \\ &\le (1-\alpha_0(x_k))V_0(x_k) \\ &\quad + \alpha_0(x_k)\min_{u_k}\left(\left(1+\frac{\psi(\bar{\delta}-1)}{\psi+1}\right)U(x_k,u_k)\right. \\ &\quad \left. + \left(\bar{\delta}-\frac{(\bar{\delta}-1)}{\psi+1}\right)J^*(x_{k+1})\right) \\ &\le \bar{\delta}(1-\alpha_0(x_k))J^*(x_k) + \alpha_0(x_k)\left(1+\frac{\psi(\bar{\delta}-1)}{\psi+1}\right) \\ &\quad \times \min_{u_k}\left(U(x_k,u_k)+J^*(x_{k+1})\right) \\ &= \left(1+(\bar{\delta}-1)\left(1-\frac{\alpha_0(x_k)}{\psi+1}\right)\right)J^*(x_k). \end{aligned} \tag{2.17}$$

Thus, the right side of inequality (2.16) holds for $i = 1$. On the other hand, we can get

$$\begin{aligned} V_1(x_k) &\ge (1-\alpha_0(x_k))V_0(x_k) \\ &\quad + \alpha_0(x_k)\min_{u_k}\left(\left(1+\frac{\psi(\underline{\delta}-1)}{\psi+1}\right)U(x_k,u_k)\right. \\ &\quad \left. + \left(\underline{\delta}-\frac{(\underline{\delta}-1)}{\psi+1}\right)J^*(x_{k+1})\right) \\ &= \left(1+(\underline{\delta}-1)\left(1-\frac{\alpha_0(x_k)}{\psi+1}\right)\right)J^*(x_k). \end{aligned} \tag{2.18}$$

Thus, the left side of inequality (2.16) holds for $i = 1$. According to mathematical induction, we can get that inequality (2.16) holds for any $i = 0, 1, \ldots$.

Third, for $i = 0, 1, \ldots$, as the learning rate $\alpha_i(x_k)$ satisfies (2.9) and (2.13), we know that

$$-\ln\left(\prod_{i=0}^{\infty}\left(1-\frac{\alpha_i(x_k)}{\psi+1}\right)\right) = -\sum_{i=0}^{\infty}\ln\left(1-\frac{\alpha_i(x_k)}{\psi+1}\right), \tag{2.19}$$

where $-\ln\left(1-\frac{\alpha_i(x_k)}{\psi+1}\right) \geq 0$, $\forall x_k \in \Omega_x \backslash \Omega_\epsilon$. Define a new function $\mathcal{F}(\alpha_i(x_k))$ as

$$\begin{aligned}\mathcal{F}(\alpha_i(x_k)) &= -\ln\left(1-\frac{\alpha_i(x_k)}{\psi+1}\right)-\frac{\alpha_i(x_k)}{\psi+1}\\ &= -\ln(\psi+1-\alpha_i(x_k))+\ln(\psi+1)-\frac{\alpha_i(x_k)}{\psi+1}. \end{aligned} \tag{2.20}$$

According to (2.20), for $0 \leq \alpha_i(x_k) \leq 1$, $i = 0, 1, \ldots$, we have

$$\frac{\mathrm{d}\mathcal{F}(\alpha_i(x_k))}{\mathrm{d}\alpha_i(x_k)} = \frac{1}{\psi+1-\alpha_i(x_k)} - \frac{1}{\psi+1} \geq 0. \tag{2.21}$$

As $\mathcal{F}(\alpha_i(x_k)) = 0$ for $\alpha_i(x_k) = 0$, we can get

$$\mathcal{F}(\alpha_i(x_k)) \geq \mathcal{F}(0) = 0. \tag{2.22}$$

Thus, for $0 \leq \alpha_i(x_k) \leq 1$, $\forall x_k \in \Omega_x \backslash \Omega_\epsilon$, we have $-\ln\left(1-\frac{\alpha_i(x_k)}{\psi+1}\right) \geq \frac{\alpha_i(x_k)}{\psi+1}$, which means

$$-\sum_{i=0}^{\infty} \ln\left(1-\frac{\alpha_i(x_k)}{\psi+1}\right) \geq \sum_{i=0}^{\infty} \frac{\alpha_i(x_k)}{\psi+1}. \tag{2.23}$$

As $\alpha_i(x_k)$ satisfies (2.15), letting $i \to \infty$, for all $x_k \in \Omega_x \backslash \Omega_\epsilon$, we have

$$\sum_{i=0}^{\infty} \frac{\alpha_i(x_k)}{\psi+1} = \frac{1}{\psi+1} \sum_{i=0}^{\infty} \alpha_i(x_k) \to \infty. \tag{2.24}$$

According to (2.23), we can obtain

$$\sum_{i=0}^{\infty} \ln\left(1-\frac{\alpha_i(x_k)}{\psi+1}\right) \to -\infty, \tag{2.25}$$

which means

$$\prod_{i=0}^{\infty} \left(1-\frac{\alpha_i(x_k)}{\psi+1}\right) = e^{-\infty} = 0. \tag{2.26}$$

According to (2.16), for all $x_k \in \Omega_x \backslash \Omega_\epsilon$, the iterative value function $V_i(x_k) \to J^*(x_k)$ as $i \to \infty$. □

For convenience of analysis, we define a new state set $\bar{\mathscr{L}}_x^i$ as

$$\bar{\mathscr{L}}_x^i = \Omega_x \backslash \mathscr{L}_x^i, i = 0, 1, \ldots. \tag{2.27}$$

According to (2.27), for any $i = 0, 1, \ldots$, we have $\mathscr{L}_x^i \cap \bar{\mathscr{L}}_x^i = \emptyset$ and $\Omega_x = \mathscr{L}_x^i \cup \bar{\mathscr{L}}_x^i$, where $\emptyset$ denotes an empty set. Based on the above preparations, the monotonicity of the iterative value function can be developed.

Theorem 2.2. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let the iterative control law $v_i(x_k)$ and iterative value function $V_i(x_k)$ be obtained by* (2.6)–(2.14). *Assume that $\mathscr{L}_x^0 = \Omega_x$ and the learning rate function satisfies*

$$0 < \alpha_0(x_k) \leq 1, \forall x_k \in \Omega_x, \tag{2.28}$$

and $\alpha_i(x_k)$ satisfies (2.13) *for $i = 1, 2, \ldots$. For all $x_k \in \Omega_x$, if the iterative value function satisfies*

$$V_1(x_k) \leq V_0(x_k), \tag{2.29}$$

then the iterative value function $V_i(x_k)$ is monotonically non-increasing and converges to the optimum, i.e.,

$$V_{i+1}(x_k) \leq V_i(x_k). \tag{2.30}$$

Proof. For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, we define a new iterative value function $\Gamma_{i+1}(x_k)$ as

$$\Gamma_{i+1}(x_k) = \min_{u_k} \big\{U(x_k, u_k) + V_i(x_{k+1})\big\}. \tag{2.31}$$

We can see that $\Gamma_{i+1}(x_k)$ is a global iterative value function. For $i = 0, 1, \ldots$, according to the learning rate function $\alpha_i(x_k)$ in *(2.9)* and *(2.13),* we can get

$$\begin{aligned} V_{i+1}(x_k) &= (1 - \alpha_i(x_k))V_i(x_k) + \alpha_i(x_k)\mathcal{V}_{i+1}(x_k) \\ &= (1 - \alpha_i(x_k))V_i(x_k) + \alpha_i(x_k)\Gamma_{i+1}(x_k) \end{aligned} \tag{2.32}$$

holds for all $x_k \in \Omega_x$. Now, we can prove the statement by mathematical induction. Let $i = 0$. We know (2.29) holds. According to

(2.8), (2.10), and (2.29), for all $x_k \in \Omega_x$, we have

$$\begin{aligned} V_1(x_k) &= (1-\alpha_0(x_k))V_0(x_k) + \alpha_0(x_k)\mathcal{V}_1(x_k) \\ &= (1-\alpha_0(x_k))V_0(x_k) + \alpha_0(x_k)\Gamma_1(x_k) \\ &\geq (1-\alpha_0(x_k))V_1(x_k) + \alpha_0(x_k)\Gamma_1(x_k). \end{aligned} \tag{2.33}$$

According to (2.28), as $0 < \alpha_0(x_k) \leq 1$ and $\mathscr{L}_x^0 = \Omega_x$, we can get $\Gamma_1(x_k) \leq V_1(x_k) \leq V_0(x_k)$, $\forall x_k \in \Omega_x$. Let $i = 1$. For all $x_k \in \Omega_x$, we have

$$\begin{aligned} \Gamma_2(x_k) &= \min_{u_k}\{U(x_k,u_k) + V_1(x_{k+1})\} \\ &\leq \min_{u_k}\{U(x_k,u_k) + V_0(x_{k+1})\} \\ &= \Gamma_1(x_k) \\ &\leq V_1(x_k). \end{aligned} \tag{2.34}$$

Then, for the learning rate function $\alpha_1(x_k)$ that satisfies $0 \leq \alpha_1(x_k) \leq 1$ for all $x_k \in \Omega_x$, we can obtain

$$\begin{aligned} V_2(x_k) &= (1-\alpha_1(x_k))V_1(x_k) + \alpha_1(x_k)\mathcal{V}_2(x_k) \\ &= (1-\alpha_1(x_k))V_1(x_k) + \alpha_1(x_k)\Gamma_2(x_k) \\ &\leq (1-\alpha_1(x_k))V_1(x_k) + \alpha_1(x_k)V_1(x_k) \\ &= V_1(x_k). \end{aligned} \tag{2.35}$$

We can prove that the statement (2.30) holds for $i = 1$.

Assume that the statement (2.30) holds for $i = \ell - 1$, $\ell = 1, 2, \ldots$. For all $x_k \in \Omega_x$, define a new set $\mathcal{I}(x_k)$ as

$$\mathcal{I}(x_k) = \{i \,|\, 0 < \alpha_i(x_k) \leq 1, i = 0, 1, \ldots, x_k \in \Omega_x\}. \tag{2.36}$$

Let $i_0, i_1, \ldots$ be the elements in $\mathcal{I}(x_k)$, which satisfies $i_0 < i_1 < \cdots$. As $0 < \alpha_0(x_k) \leq 1$, we have $i_0 = 0$, which means $\mathcal{I}(x_k) \neq \emptyset$, $\forall x_k \in \Omega_x$. For all $x_k \in \Omega_x$, choose a constant $i_\hbar$ that satisfies

$$\begin{aligned} i_\hbar &= \arg\min_{i_j \in \mathcal{I}}\{(\ell-1) - i_j\}, \\ &\text{s.t. } i_j < \ell - 1,\ j = 0, 1, \ldots. \end{aligned} \tag{2.37}$$

If the inequality (2.30) holds for $i = \ell - 1$, $\ell = 1, 2, \ldots$, then according to (2.14) and (2.30), we can get

$$\begin{aligned} V_\ell(x_k) &= (1 - \alpha_{\ell-1}(x_k))V_{\ell-1}(x_k) + \alpha_{\ell-1}(x_k)\mathcal{V}_\ell(x_k) \\ &= (1 - \alpha_{\ell-1}(x_k))V_{\ell-1}(x_k) + \alpha_{\ell-1}(x_k)\Gamma_\ell(x_k) \\ &\geq (1 - \alpha_{\ell-1}(x_k))V_\ell(x_k) + \alpha_{\ell-1}(x_k)\Gamma_\ell(x_k). \end{aligned} \tag{2.38}$$

For all $x_k \in \mathscr{L}_x^{\ell-1}$, we can get $\Gamma_\ell(x_k) \leq V_\ell(x_k) \leq V_{\ell-1}(x_k)$. For $i = \ell$, we have

$$\begin{aligned} \Gamma_{\ell+1}(x_k) &= \min_{u_k}\{U(x_k, u_k) + V_\ell(x_{k+1})\} \\ &\leq \min_{u_k}\{U(x_k, u_k) + V_{\ell-1}(x_{k+1})\} \\ &= \Gamma_\ell(x_k) \\ &\leq V_\ell(x_k). \end{aligned} \tag{2.39}$$

Thus, for all $x_k \in \mathscr{L}_x^{\ell-1}$, we can obtain

$$\begin{aligned} V_{\ell+1}(x_k) &= (1 - \alpha_\ell(x_k))V_\ell(x_k) + \alpha_\ell(x_k)\mathcal{V}_{\ell+1}(x_k) \\ &= (1 - \alpha_\ell(x_k))V_\ell(x_k) + \alpha_\ell(x_k)\Gamma_{\ell+1}(x_k) \\ &\leq (1 - \alpha_\ell(x_k))V_\ell(x_k) + \alpha_\ell(x_k)V_\ell(x_k) \\ &= V_\ell(x_k). \end{aligned} \tag{2.40}$$

For all $x_k \in \bar{\mathscr{L}}_x^{\ell-1}$, we have $V_\ell(x_k) = V_{\ell-1}(x_k)$. Let $i_\hbar$ satisfy (2.37). Let $\lambda = (\ell - 1) - i_\hbar$ and we can derive that $0 < \lambda \leq \ell - 1$. Then, for all $\tilde{i} \in \{i_\hbar + 1, i_\hbar + 2, \ldots, i_\hbar + \lambda - 1, \ell - 1\}$, we have $x_k \in \bar{\mathscr{L}}_x^{\tilde{i}}$. Thus, for all $x_k \in \bar{\mathscr{L}}_x^{\ell-1}$, we can get

$$V_\ell(x_k) = V_{i_\hbar+\lambda}(x_k) = \cdots = V_{i_\hbar+1}(x_k). \tag{2.41}$$

As the inequality (2.30) holds for $i = \ell - 1$, $\ell = 1, 2, \ldots$, we have $V_{i_\hbar+1}(x_k) \leq V_{i_\hbar}(x_k)$ and $x_k \in \mathscr{L}_x^{i_\hbar}$. According to (2.14), we have

$$\begin{aligned} V_{i_\hbar+1}(x_k) &= (1 - \alpha_{i_\hbar}(x_k))V_{i_\hbar}(x_k) + \alpha_{i_\hbar}(x_k)\mathcal{V}_{i_\hbar+1}(x_k) \\ &= (1 - \alpha_{i_\hbar}(x_k))V_{i_\hbar}(x_k) + \alpha_{i_\hbar}(x_k)\Gamma_{i_\hbar+1}(x_k) \\ &\geq (1 - \alpha_{i_\hbar}(x_k))V_{i_\hbar+1}(x_k) + \alpha_{i_\hbar}(x_k)\Gamma_{i_\hbar+1}(x_k). \end{aligned} \tag{2.42}$$

As $x_k \in \mathscr{L}_x^{i_\hbar}$, we can get

$$\Gamma_{i_\hbar+1}(x_k) \leq V_{i_\hbar+1}(x_k) \leq V_{i_\hbar}(x_k). \tag{2.43}$$

According to (2.31), we can obtain

$$\begin{aligned}
\Gamma_{i_\hbar+2}(x_k) &= \min_{u_k}\{U(x_k,u_k) + V_{i_\hbar+1}(x_{k+1})\} \\
&\leq \min_{u_k}\{U(x_k,u_k) + V_{i_\hbar}(x_{k+1})\} \\
&= \Gamma_{i_\hbar+1}(x_k) \\
&\leq V_{i_\hbar+1}(x_k).
\end{aligned} \tag{2.44}$$

For $i = \ell$, we can get the following inequality

$$\begin{aligned}
\Gamma_{\ell+1}(x_k) &= \min_{u_k}\{U(x_k,u_k) + V_\ell(x_{k+1})\} \\
&\leq \min_{u_k}\{U(x_k,u_k) + V_{\ell-1}(x_{k+1})\} \\
&\quad\vdots \\
&\leq \min_{u_k}\{U(x_k,u_k) + V_{i_\hbar+1}(x_{k+1})\} \\
&= \Gamma_{i_\hbar+2}(x_k).
\end{aligned} \tag{2.45}$$

Hence, for $i = \ell$ and $\forall x_k \in \bar{\mathscr{L}}_x^{\ell-1}$, according to (2.14), (2.41), (2.44), and (2.45), we can get

$$\begin{aligned}
V_{\ell+1}(x_k) &= (1-\alpha_\ell(x_k))V_\ell(x_k) + \alpha_\ell(x_k)\mathcal{V}_{\ell+1}(x_k) \\
&= (1-\alpha_\ell(x_k))V_\ell(x_k) + \alpha_\ell(x_k)\Gamma_{\ell+1}(x_k) \\
&\leq (1-\alpha_\ell(x_k))V_{i_\hbar+1}(x_k) + \alpha_\ell(x_k)\Gamma_{i_\hbar+2}(x_k) \\
&\leq (1-\alpha_\ell(x_k))V_{i_\hbar+1}(x_k) + \alpha_\ell(x_k)V_{i_\hbar+1}(x_k) \\
&= V_{i_\hbar+1}(x_k) \\
&= V_\ell(x_k).
\end{aligned} \tag{2.46}$$

According to (2.40) and (2.46), for $i = \ell$, we can obtain the inequality (2.30) holds for all $x_k \in \Omega_x$. The mathematical induction is complete. □

Remark 2.2. In Theorem 2.2, the monotonicity of the local iterative ADP algorithm has been analyzed. To guarantee the monotonicity, additional constraints are required for the learning rate function, especially for the learning rate function $\alpha_0(x_k)$, where the condition $\alpha_0(x_k) = 0$ is not permitted. If $\alpha_0(x_k) = 0$ is permitted, the monotonicity in Theorem 2.2 may not hold. For example, if we let $\alpha_0(x_k) \equiv 0$, $\forall\, x_k \in \Omega_x$, then we have $V_1(x_k) \equiv V_0(x_k)$, which satisfies (2.29). However, it obviously cannot guarantee the monotonicity in (2.30) for all $i = 0, 1, \ldots$. On the other hand, in Theorem 2.2, as it requires $\mathscr{L}_x^0 = \Omega_x$ which means the iterative value function and iterative control law are updated for all $x_k \in \Omega_x$ for their first iterations. In the following corollary, the constraints for the initial iteration can be relaxed.

Corollary 2.1. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let $v_i(x_k)$ and $V_i(x_k)$ be obtained by* (2.6)–(2.14)*. Let the learning rate function $\alpha_i(x_k)$ satisfies* (2.9), (2.13), *and* (2.15)*. Let $\theta = 0, 1, \ldots$ be an arbitrary non-negative integer. For a nonnegative integer j that satisfies $0 \le j \le \theta$, assume that the state subset $\mathscr{L}_x^j$ satisfies*

$$\mathscr{L}_x^0 \cup \mathscr{L}_x^1 \cup \cdots \cup \mathscr{L}_x^\theta = \bigcup_{j=0}^{\theta} \mathscr{L}_x^j = \Omega_x. \tag{2.47}$$

If for all $x_k \in \mathscr{L}_x^j$, $j = 0, 1, \ldots, \vartheta$, and the iterative value function $V_j(x_k)$ satisfies

$$V_{j+1}(x_k) \le V_j(x_k), \tag{2.48}$$

then for $i = 0, 1, \ldots$, the iterative value function $V_i(x_k)$ is monotonically non-increasing and converges to the optimum.

Proof. First, we know that for $j = 0, 1, \ldots, \theta$, $V_{j+1}(x_k) = V_j(x_k)$ holds for all $x_k \in \bar{\mathscr{L}}_x^j$. We can easily derive that for $j = 0, 1, \ldots$, the inequality (2.48) holds for all $x_k \in \Omega_x$.

Now, we will prove that inequality (2.30) holds for any $i = 0, 1, \ldots$. For $i = \theta + 1$, as the state subset $\mathscr{L}_x^j$, $j = 0, 1, \ldots, \theta$, satisfies (2.47), for an arbitrary state $\bar{x}_k \in \Omega_x$, it must locate at least in one state subset. Without loss of generality, we assume

$$\bar{x}_k \in \left\{ \mathscr{L}_x^{j_0} \cap \mathscr{L}_x^{j_1} \cap \cdots \cap \mathscr{L}_x^{j_{\bar{\theta}}} \right\}, \tag{2.49}$$

where $0 \leq j_0 \leq j_1 \leq \cdots \leq j_{\bar{\theta}} \leq \theta$. Then we have $V_{j_{\bar{\theta}}+1}(\bar{x}_k) = V_{j_{\bar{\theta}}+2}(\bar{x}_k) = \cdots = V_{\theta+1}(\bar{x}_k)$. For $i = j_{\bar{\theta}}$, according to (2.48), we have

$$\begin{aligned} V_{j_{\bar{\theta}}+1}(\bar{x}_k) &= (1-\alpha_{j_{\bar{\theta}}}(\bar{x}_k))V_{j_{\bar{\theta}}}(\bar{x}_k) + \alpha_{j_{\bar{\theta}}}(\bar{x}_k)\mathcal{V}_{j_{\bar{\theta}}+1}(\bar{x}_k) \\ &= (1-\alpha_{j_{\bar{\theta}}}(\bar{x}_k))V_{j_{\bar{\theta}}}(\bar{x}_k) + \alpha_{j_{\bar{\theta}}}(\bar{x}_k)\Gamma_{j_{\bar{\theta}}+1}(\bar{x}_k) \\ &\geq (1-\alpha_{j_{\bar{\theta}}}(\bar{x}_k))V_{j_{\bar{\theta}}+1}(\bar{x}_k) + \alpha_{j_{\bar{\theta}}}(\bar{x}_k)\Gamma_{j_{\bar{\theta}}+1}(\bar{x}_k), \end{aligned} \tag{2.50}$$

which obtains $\Gamma_{j_{\bar{\theta}}+1}(\bar{x}_k) \leq V_{j_{\bar{\theta}}+1}(\bar{x}_k)$ for $\bar{x}_k$ satisfying (2.49). According to (2.31), we can obtain

$$\begin{aligned} \Gamma_{\theta+2}(\bar{x}_k) &= \min_{u_k}\{U(\bar{x}_k, u_k) + V_{\theta+1}(F(\bar{x}_k, u_k))\} \\ &\leq \min_{u_k}\{U(\bar{x}_k, u_k) + V_{\theta}(F(\bar{x}_k, u_k))\} \\ &\quad \vdots \\ &\leq \min_{u_k}\{U(\bar{x}_k, u_k) + V_{j_{\bar{\theta}}}(F(\bar{x}_k, u_k))\} \\ &= \Gamma_{j_{\bar{\theta}}+1}(\bar{x}_k). \end{aligned} \tag{2.51}$$

Then, for the state $\bar{x}_k$, we can obtain

$$\begin{aligned} V_{\theta+2}(\bar{x}_k) &= (1-\alpha_{\theta}(\bar{x}_k))V_{\theta+1}(\bar{x}_k) + \alpha_{\theta}(\bar{x}_k)\mathcal{V}_{\theta+2}(\bar{x}_k) \\ &= (1-\alpha_{\theta}(\bar{x}_k))V_{\theta+1}(\bar{x}_k) + \alpha_{\theta}(\bar{x}_k)\Gamma_{\theta+2}(\bar{x}_k) \\ &\leq (1-\alpha_{\theta}(\bar{x}_k))V_{j_{\bar{\theta}}+1}(\bar{x}_k) + \alpha_{\theta}(\bar{x}_k)\Gamma_{j_{\bar{\theta}}+1}(\bar{x}_k) \\ &= V_{j_{\bar{\theta}}+1}(\bar{x}_k) \\ &= V_{\theta+1}(\bar{x}_k). \end{aligned} \tag{2.52}$$

Hence, we have $V_{\theta+2}(\bar{x}_k) \leq V_{\theta+1}(\bar{x}_k)$. As $\bar{x}_k \in \Omega_x$ is arbitrary, then we have $V_{\theta+2}(x_k) \leq V_{\theta+1}(x_k)$, $\forall x_k \in \Omega_x$. According to mathematical induction, we can prove the conclusion for all $i = 0, 1, \ldots$. The proof is complete. □

Theorem 2.3. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let the iterative value function $V_i(x_k)$ and the iterative control law $v_i(x_k)$ be obtained by* (2.6)–(2.14). *For all $x_k \in \Omega_x$, let the learning rate function $\alpha_i(x_k)$, $i = 0, 1, \ldots$ satisfy* (2.13), (2.15) *and* (2.28). *If for all $x_k \in \Omega_x$, the*

iterative value function satisfies

$$V_1(x_k) \geq V_0(x_k), \tag{2.53}$$

then the iterative value function $V_i(x_k)$ *is monotonically non-decreasing and converges to the optimum, i.e.,*

$$V_{i+1}(x_k) \geq V_i(x_k). \tag{2.54}$$

Proof. The conclusion can be derived by mathematical induction according to (2.33) to (2.46) and the detail is omitted here. □

Theorem 2.4. *For* $i = 0, 1, \ldots$ *and for all* $x_k \in \Omega_x$, *let the iterative control law* $v_i(x_k)$ *and iterative value function* $V_i(x_k)$ *be obtained by* (2.6)–(2.14). *For all* $x_k \in \Omega_x$, *let the learning rate function* $\alpha_i(x_k)$, $i = 0, 1, \ldots$ *satisfy* (2.9), (2.13), (2.15), *and* (2.28). *If for all* $x_k \in \Omega_x$ *the iterative value function is non-increasing monotonically convergent, for any* $i = 0, 1, \ldots$, *then the iterative value function*

$$V_i(x_k) \geq J^*(x_k). \tag{2.55}$$

Proof. According to (2.30), for any $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, we have

$$V_i(x_k) \geq V_{i+1}(x_k) \geq V_{i+2}(x_k) \geq \cdots . \tag{2.56}$$

Then, for all $l \geq i$, we can get

$$V_i(x_k) \geq V_l(x_k). \tag{2.57}$$

Let $l \to \infty$, and according to Theorem 2.1, we can obtain

$$V_i(x_k) \geq \lim_{l \to \infty} V_l(x_k) = J^*(x_k). \tag{2.58}$$

The proof is complete. □

Corollary 2.2. *For* $i = 0, 1, \ldots$ *and for all* $x_k \in \Omega_x$, *let the iterative control law* $v_i(x_k)$ *and iterative value function* $V_{i+1}(x_k)$ *be obtained by* (2.6)–(2.14). *For all* $x_k \in \Omega_x$, *let the learning rate* $\alpha_i(x_k)$, $i = 0, 1, \ldots$ *satisfy* (2.9), (2.15), *and* (2.28). *If for all* $x_k \in \Omega_x$, *the iterative value function satisfies* (2.29), *then for all* $i = 0, 1, \ldots$, *the iterative value function*

$$V_i(x_k) \leq J^*(x_k). \tag{2.59}$$

Remark 2.3. For Theorems 2.1–2.4, the convergence, optimality and monotonicity properties of the developed local iterative ADP algorithm are analyzed. We show that the iterative value function $V_i(x_k)$ is proven to be the optimum as the iteration index $i \to \infty$. For traditional iterative ADP algorithms [2, 3, 7, 16], to guarantee that the algorithm terminates within finite iterations, the following convergence criterion

$$|V_{i+1}(x_k) - V_i(x_k)| \leq \varepsilon, \tag{2.60}$$

is established, where ε is the computation precision. However, the convergence termination condition (2.60) is not feasible for the developed local iterative ADP algorithm. For example, if we let $\alpha_i(x_k) \equiv 0$, $\forall\, x_k \in \Omega_x$, then we have $V_{i+1}(x_k) = V_i(x_k)$, while it cannot guarantee $V_i(x_k) \to J^*(x_k)$. Hence, a new convergence criterion should be established for the developed local iterative ADP algorithm.

Before the next theorem, the following lemma is necessary.

Lemma 2.2. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let $v_i(x_k)$ and $V_i(x_k)$ be obtained by* (2.6)–(2.14). *Let $\vartheta = 0, 1, \ldots$ be a non-negative integer. For any $i = 0, 1, \ldots$, if the state subset $\mathscr{L}_x^{i+j}$, $j = 0, 1, \ldots, \vartheta$, satisfies*

$$\mathscr{L}_x^i \cup \mathscr{L}_x^{i+1} \cup \cdots \cup \mathscr{L}_x^{i+\vartheta} = \bigcup_{j=0}^{\vartheta} \mathscr{L}_x^{i+j} = \Omega_x, \tag{2.61}$$

then the iterative value function $V_\ell(x_k)$ is a positive definite function for all $\ell \geq i + \vartheta$.

Proof. For any $i = 0, 1, \ldots$, we have $V_i(x_k)$ is a positive semi-definite function for x_k. Let $j = 0$. According to (2.12), we have $\mathcal{V}_{i+1}(x_k) > 0$ for all $x_k \in \mathscr{L}_x^i$ and $x_k \neq 0$. According to (2.14), we can obtain

$$\begin{cases} V_{i+1}(x_k) > 0, & x_k \in \mathscr{L}_x^i, \\ V_{i+1}(x_k) \geq 0, & x_k \in \bar{\mathscr{L}}_x^i. \end{cases} \tag{2.62}$$

For $j = 1$, the iterative value function can be expressed as

$$V_{i+2}(x_k) = (1 - \alpha_{i+1}(x_k))V_{i+1}(x_k) + \alpha_{i+1}(x_k)\mathcal{V}_{i+2}(x_k). \tag{2.63}$$

According to (2.12), we have $\mathcal{V}_{i+2}(x_k) > 0$ for $x_k \in \mathscr{L}_x^{i+1}$ and $x_k \neq 0$. If $x_k \in \bar{\mathscr{L}}_x^{i+1}$, we have $\alpha_{i+1}(x_k) = 0$, which means $V_{i+2}(x_k) = V_{i+1}(x_k)$. According to (2.62), we have $V_{i+2}(x_k) > 0$ for all $x_k \in \mathscr{L}_x^i$ and $x_k \neq 0$. Thus, for $x_k \neq 0$, we can obtain

$$\begin{cases} V_{i+2}(x_k) > 0, & x_k \in \mathscr{L}_x^i \cup \mathscr{L}_x^{i+1}, \\ V_{i+2}(x_k) \geq 0, & \text{Else.} \end{cases} \tag{2.64}$$

According to mathematical induction, for $0 \leq j \leq \vartheta$ and $x_k \neq 0$, we can prove

$$\begin{cases} V_{i+j+1}(x_k) > 0, & x_k \in \mathscr{L}_x^i \cup \mathscr{L}_x^{i+1} \cup \cdots \cup \mathscr{L}_x^{i+j}, \\ V_{i+j+1}(x_k) \geq 0, & \text{Else.} \end{cases} \tag{2.65}$$

For $j = \vartheta$, we have $\cup_{j=0}^{\vartheta} \mathscr{L}_x^{i+j} = \Omega_x$, according to (2.61). We know that $V_{i+\vartheta+1}(x_k) > 0$ for all $x_k \in \Omega_x$ and $x_k \neq 0$, which means $V_{\vartheta+1}(x_k)$ is positive definite. According to Lemma 2.1(iii), we can prove that $V_\ell(x_k)$ is a positive definite function for all $\ell \geq i + \vartheta$. The proof is complete. □

Remark 2.4. According to the definition of the performance index function in (2.2), under Assumption 2.1, we can derive that $J^*(x_k)$ is a positive definite function. Hence, it obviously requires approximating the optimal performance index function using a positive definite iterative value function. If the initial value function $\Psi(x_k) > 0$, according to Lemma 2.1, we know that $V_i(x_k) > 0$, $\forall\, i \geq 0$. It means that we can use $V_i(x_k)$, $\forall\, i \geq 0$ to approximate the optimal performance index function. If $\Psi(x_k) \geq 0$, then the developed algorithm cannot terminate before it outputs a positive definite function. For $\theta = 0, 1, \ldots,$ define a new parameter ρ as

$$\rho = \arg\min_{\theta} \left\{ \bigcup_{j=0}^{\theta} \mathscr{L}_x^j = \Omega_x \right\}. \tag{2.66}$$

Then, for all $x_k \in \Omega_x$, we can define the minimum iteration condition as

$$i_{\min} = \begin{cases} 0, & \text{if } \Psi(x_k) > 0, \\ \rho, & \text{if } \Psi(x_k) \geq 0. \end{cases} \tag{2.67}$$

Based on above definitions, we can derive the the following convergence criterion theorem.

Theorem 2.5. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let $v_i(x_k)$ and $V_i(x_k)$ be obtained by (2.6)–(2.14). Let $\varepsilon > 0$ be a positive constant. If for $j = 0, 1, \ldots, \vartheta$, $\forall\, x_k \in \mathscr{L}_x^i$, the iteration index $i \geq i_{\min}$, and the iterative value function satisfies*

$$|V_{i+j+1}(x_k) - V_{i+j}(x_k)| \leq \alpha_{i+j}(x_k)\varepsilon, \tag{2.68}$$

where $i_{\min}$ is defined in (2.67) and $\mathscr{L}_x^{i+j}$ satisfies (2.61), then the iterative value function $V_{i+\vartheta}(x_k)$ converges to the optimum. i.e., $V_{i+\vartheta}(x_k) \to J^*(x_k)$, as $\varepsilon \to 0$.

Proof. The conclusion can be proven in five steps.

(1) *Show that for all $x_k \in \Omega_x$, the iterative value function $V_{i+\vartheta}(x_k)$ satisfies the following HJB function*

$$V_{i+\vartheta}(x_k) = \min_{u_k}\{U(x_k, u_k) + V_{i+\vartheta}(x_{k+1})\}. \tag{2.69}$$

For $j = \vartheta$, according to (2.14), we have

$$V_{i+\vartheta+1}(x_k) = (1 - \alpha_{i+\vartheta}(x_k))V_{i+\vartheta}(x_k) + \alpha_{i+\vartheta}(x_k)\mathcal{V}_{i+\vartheta+1}(x_k). \tag{2.70}$$

According to (2.68), we can get

$$\begin{aligned} V_{i+\vartheta}(x_k) - \alpha_{i+\vartheta}(x_k)\,\varepsilon \leq{}& (1 - \alpha_{i+\vartheta}(x_k))V_{i+\vartheta}(x_k) \\ &+ \alpha_{i+\vartheta}(x_k)\mathcal{V}_{i+\vartheta+1}(x_k) \\ \leq{}& V_{i+\vartheta}(x_k) + \alpha_{i+\vartheta}(x_k)\,\varepsilon. \end{aligned} \tag{2.71}$$

For all $x_k \in \mathscr{L}_x^{i+\vartheta}$, we have $0 < \alpha_{i+\vartheta} \leq 1$. According to (2.12) and (2.68), for all $x_k \in \mathscr{L}_x^{i+\vartheta}$, we can obtain

$$\begin{aligned} V_{i+\vartheta}(x_k) - \varepsilon &\leq U(x_k, v_{i+\vartheta}(x_k)) + V_{i+\vartheta}(F(x_k, v_{i+\vartheta}(x_k))) \\ &\leq V_{i+\vartheta}(x_k) + \varepsilon. \end{aligned} \tag{2.72}$$

As $v_{i+\vartheta}(x_k)$ satisfies (2.11) for $i+\vartheta$, letting $\varepsilon \to 0$, for all $x_k \in \mathscr{L}_x^{i+\vartheta}$, we have (2.69) holds for all $x_k \in \mathscr{L}_x^{i+\vartheta}$. Let $j = \vartheta - 1$. For all

$x_k \in \{\mathscr{L}_x^{i+\vartheta-1} \cap \bar{\mathscr{L}}_x^{i+\vartheta}\}$, we have

$$\begin{aligned} V_{i+\vartheta}(x_k) = &(1-\alpha_{i+\vartheta-1}(x_k))V_{i+\vartheta-1}(x_k) \\ &+ \alpha_{i+\vartheta-1}(x_k)\big(U(x_k, v_{i+\vartheta-1}(x_k)) \\ &+ V_{i+\vartheta-1}(F(x_k, v_{i+\vartheta-1}(x_k)))\big), \end{aligned} \tag{2.73}$$

where $0 < \alpha_{i+\vartheta-1}(x_k) \leq 1$. According to (2.68), for all $x_k \in \{\mathscr{L}_x^{i+\vartheta-1} \cap \bar{\mathscr{L}}_x^{i+\vartheta}\}$, we can get

$$\begin{aligned} V_{i+\vartheta-1}(x_k) - \varepsilon &\leq U(x_k, v_{i+\vartheta-1}(x_k)) + V_{i+\vartheta-1}(F(x_k, v_{i+\vartheta-1}(x_k))) \\ &\leq V_{i+\vartheta-1}(x_k) + \varepsilon. \end{aligned} \tag{2.74}$$

As $v_{i+\vartheta-1}(x_k)$ satisfies (2.11) for $i+\vartheta-1$, letting $\varepsilon \to 0$, we can obtain

$$V_{i+\vartheta-1}(x_k) = \min_{u_k}\{U(x_k, u_k) + V_{i+\vartheta-1}(x_{k+1})\} \tag{2.75}$$

holds for all $x_k \in \{\mathscr{L}_x^{i+\vartheta-1} \cap \bar{\mathscr{L}}_x^{i+\vartheta}\}$. For $\varepsilon \to 0$, we can obtain $V_{i+\vartheta}(x_k) = V_{i+\vartheta-1}(x_k)$, $\forall x_k \in \Omega_x$. Hence, we can obtain (2.69) for all $x_k \in \{\mathscr{L}_x^{i+\vartheta-1} \cap \bar{\mathscr{L}}_x^{i+\vartheta}\}$. According to mathematical induction, for any $j = 0, 1, \ldots, \vartheta$, we can obtain (2.69) for all $x_k \in \{\mathscr{L}_x^{i+j} \cap \bar{\mathscr{L}}_x^{i+j+1} \cap \cdots \cap \bar{\mathscr{L}}_x^{i+\vartheta-1} \cap \bar{\mathscr{L}}_x^{i+\vartheta}\}$. As $\mathscr{L}_x^{i+j}$ satisfies (2.61), we can obtain that (2.69) holds for all $x_k \in \Omega_x$ as $\varepsilon \to 0$.

(2) *Show that for all $x_k \in \Omega_x$, the iterative control law $v_{i+\vartheta}(x_k)$ satisfies*

$$v_{i+\vartheta}(x_k) = \arg\min_{u_k}\{U(x_k, u_k) + V_{i+\vartheta}(x_{k+1})\}. \tag{2.76}$$

For all $x_k \in \mathscr{L}_x^{i+\vartheta}$, we have (2.76) obviously holds. As $\forall x_k \in \Omega_x$, $V_{i+\vartheta}(x_k) = V_{i+\vartheta-1}(x_k)$ holds, when $\varepsilon \to \infty$, for all $x_k \in \{\mathscr{L}_x^{i+\vartheta-1} \cap \bar{\mathscr{L}}_x^{i+\vartheta}\}$, the iterative control law $v_{i+\vartheta-1}(x_k)$ satisfies

$$\begin{aligned} v_{i+\vartheta-1}(x_k) &= \arg\min_{u_k}\{U(x_k, u_k) + V_{i+\vartheta-1}(x_{k+1})\} \\ &= \arg\min_{u_k}\{U(x_k, u_k) + V_{i+\vartheta}(x_{k+1})\} \\ &= v_{i+\vartheta}(x_k). \end{aligned} \tag{2.77}$$

According to mathematical induction, for any $j = 0, 1, \ldots, \vartheta$, we can obtain (2.76) for all $x_k \in \{\mathscr{L}_x^{i+j} \cap \bar{\mathscr{L}}_x^{i+j+1} \cap \cdots \cap \bar{\mathscr{L}}_x^{i+\vartheta-1} \cap \bar{\mathscr{L}}_x^{i+\vartheta}\}$. As $\mathscr{L}_x^{i+j}$ satisfies (2.61), we can obtain that (2.76) holds for all $x_k \in \Omega_x$.

Now, we let $\mu(x_k)$ be an arbitrary admissible control law, and define a new value function $P_\tau(x_k)$, $\tau = 0, 1, \ldots$, which satisfies

$$P_{\tau+1}(x_k) = U(x_k, \mu(x_k)) + P_\tau(x_{k+1}), \tag{2.78}$$

where $P_0(x_k) = V_{i+\theta}(x_k)$ and $x_{k+1} = F(x_k, \mu(x_k))$. Then, we can declare the third step of the proof.

(3) *Show that for all $x_k \in \Omega_x$ and any $\tau = 0, 1, \ldots$, the iterative value function satisfies*

$$V_{i+\vartheta}(x_k) \leq P_\tau(x_k). \tag{2.79}$$

The conclusion can be proven by mathematical induction. The inequality (2.79) obviously holds for $\tau = 0$. For $\tau = 1$, according to (2.69) and (2.76), for all $x_k \in \Omega_x$, we have

$$\begin{aligned} V_{i+\vartheta}(x_k) &= U(x_k, v_{i+\vartheta}(x_k)) + V_{i+\vartheta}(F(x_k, v_{i+\vartheta}(x_k))) \\ &= \min_{u_k}\{U(x_k, u_k) + V_{i+\vartheta}(x_{k+1})\} \\ &\leq U(x_k, \mu(x_k)) + V_{i+\vartheta}(F(x_k, \mu(x_k))) \\ &= U(x_k, \mu(x_k)) + P_0(x_{k+1}) \\ &= P_1(x_k). \end{aligned} \tag{2.80}$$

Assume (2.79) holds for $\tau = \ell - 1$, $\ell = 1, 2, \ldots$ and $\forall\, x_k \in \Omega_x$. For $\tau = \ell$ and $\forall\, x_k \in \Omega_x$, we have

$$\begin{aligned} V_{i+\vartheta}(x_k) &= U(x_k, v_{i+\vartheta}(x_k)) + V_{i+\vartheta}(F(x_k, v_{i+\vartheta}(x_k))) \\ &\leq U(x_k, \mu(x_k)) + V_{i+\vartheta}(F(x_k, \mu(x_k))) \\ &\leq U(x_k, \mu(x_k)) + P_{\ell-1}(F(x_k, \mu(x_k))) \\ &= P_\ell(x_k). \end{aligned} \tag{2.81}$$

Thus, we have (2.79) holds for any $\tau = 0, 1, \ldots$.

(4) *Show that for all $x_k \in \Omega_x$ and any admissible control law $\mu(x_k)$, the iterative value function $V_{i+\vartheta}(x_k)$ satisfies*

$$V_{i+\vartheta}(x_k) \leq \sum_{\tau=0}^{\infty} U(x_{k+\tau}, \mu(x_{k+\tau})). \tag{2.82}$$

According to (2.78), for all $x_k \in \Omega_x$, we can get

$$\begin{aligned} P_\tau(x_k) &= \sum_{\ell=0}^{\tau-1} U(x_{k+\ell}, \mu(x_{k+\ell})) + P_0(x_{k+\tau}) \\ &= \sum_{\ell=0}^{\tau-1} U(x_{k+\ell}, \mu(x_{k+\ell})) + V_{i+\vartheta}(x_{k+\tau}), \end{aligned} \tag{2.83}$$

where $\sum_y^z(\cdot) = 0$ for $y > z$. We have $\lim_{\tau\to\infty} \sum_{\ell=0}^{\tau-1} U(x_{k+\ell}, \mu(x_{k+\ell})) < \infty$ and $x_{k+\tau} \to 0$ as $\tau \to \infty$. According to Lemma 2.2, we have $V_{i+\vartheta}(x_k)$ is positive definite, which means $V_{i+\vartheta}(x_{k+\tau}) \to 0$ as $\tau \to \infty$. Thus, for all $x_k \in \Omega_x$, we have

$$\lim_{\tau\to\infty} P_\tau(x_k) = \lim_{\tau\to\infty} \sum_{\ell=0}^{\tau-1} U(x_{k+\ell}, \mu(x_{k+\ell})) < \infty. \tag{2.84}$$

According to (2.79), letting $\tau \to \infty$, we can obtain (2.82).

(5) *Show that the iterative value function $V_{i+\vartheta}(x_k)$ equals to the optimal performance index function $J^*(x_k)$.*

According to the expression of $V_{i+\vartheta}(x_k)$ in (2.69), for all $x_k \in \Omega_x$, we have

$$V_{i+\vartheta}(x_k) \geq J^*(x_k). \tag{2.85}$$

On the other hand, for an arbitrary admissible control law $\mu(x_k)$, we have (2.82) holds. Let $\mu(x_k) = u^*(x_k)$, where $u^*(x_k)$ is an optimal control law. Then, we can get

$$\begin{aligned} V_{i+\vartheta+1}(x_k) &\leq \sum_{\tau=0}^{\infty} U(x_{k+\tau}, u^*(x_{k+\tau})) \\ &= \min_{\underline{u}_k} \left\{ \sum_{\tau=0}^{\infty} U(x_{k+\tau}, u_{k+\tau}) \right\} \\ &= J^*(x_k). \end{aligned} \tag{2.86}$$

Hence, we can obtain the conclusion. □

2.4 Simulation Example

To evaluate the performance of our local iterative ADP algorithm, the optimal flight control problem for the F-8 Crusader [6] is considered. The system of F-8 can be expressed by the following non-affine nonlinear system

$$\begin{cases} \dot{x}_1 = -0.877x_1 + x_3 - 0.088x_1x_3 + 0.47x_1^2 - 0.019x_2^2 \\ \qquad - x_1^2x_3 + 3.846x_1^3 - 0.215u + 0.28x_1^2u \\ \qquad + 0.47x_1u^2 + 0.63u^3, \\ \dot{x}_2 = x_3, \\ \dot{x}_3 = -4.208x_1 - 0.396x_3 - 0.47x_1^2 - 3.564x_1^3 - 20.967u \\ \qquad + 6.265x_1^2u + 46x_1u^2 + 61.4u^3, \end{cases} \tag{2.87}$$

where x_1 is the angle of attack (rad), x_2 is the pitch angle (rad), x_3 is the pitch rate (rad/s), and u is the control input (manipulated variable) provided by the tail detection (or elevator) angle (rad). Discretizing the system function with the sampling interval $\Delta t = 0.1$s leads to

$$\begin{cases} x_{1(k+1)} = 0.9123x_{1k} + 0.1x_{3k} - 0.0088x_{1k}x_{3k} + 0.047x_{1k}^2 \\ \qquad - 0.0019x_{2k}^2 - 0.1x_{1k}^2x_{3k} + 0.3846x_{1k}^3 - 0.0215u_k \\ \qquad + 0.028x_{1k}^2u_k + 0.047x_{1k}u_k^2 + 0.063u_k^3, \\ x_{2(k+1)} = x_{2k} + 0.1x_{3k}, \\ x_{3(k+1)} = -0.4208x_{1k} + 0.9604x_{3k} - 0.047x_{1k}^2 \\ \qquad - 0.3564x_{1k}^3 - 2.0967u_k + 0.6265x_{1k}^2u_k \\ \qquad + 4.6x_{1k}u_k^2 + 6.14u_k^3. \end{cases} \tag{2.88}$$

The utility function is chosen as $U(x_k, u_k) = x_k^{\mathsf{T}}Qx_k + u_k^{\mathsf{T}}Ru_k$, where $Q = I_1$, $R = 2I_2$ and I_1 and I_2 denote the identity matrices with suitable dimensions. Define the state space as $\Omega_x^2 = \{(x_{1k}, x_{2k}, x_{3k}) | -1 \le x_{1k} \le 1, -1 \le x_{2k} \le 1, -1 \le x_{3k} \le 1\}$. We collect the state data by discretizing the state space. The discretized state set is expressed as $\{(x_{1k}, x_{2k}, x_{3k})\} = \{(-1 + 0.02\varrho_1, -1 + 0.02\varrho_2, -1 + 0.02\varrho_3)\}$, $\varrho_1, \varrho_2, \varrho_3 = 0, 1, \ldots, 100$. Thus, we collect $\bar{p} = 101^3$ state data to train the neural networks. Choose an initial value function $\bar{\Psi}(x_k) = x_k^{\mathsf{T}}\bar{P}x_k$, where $\bar{P}$ is arbitrarily chosen

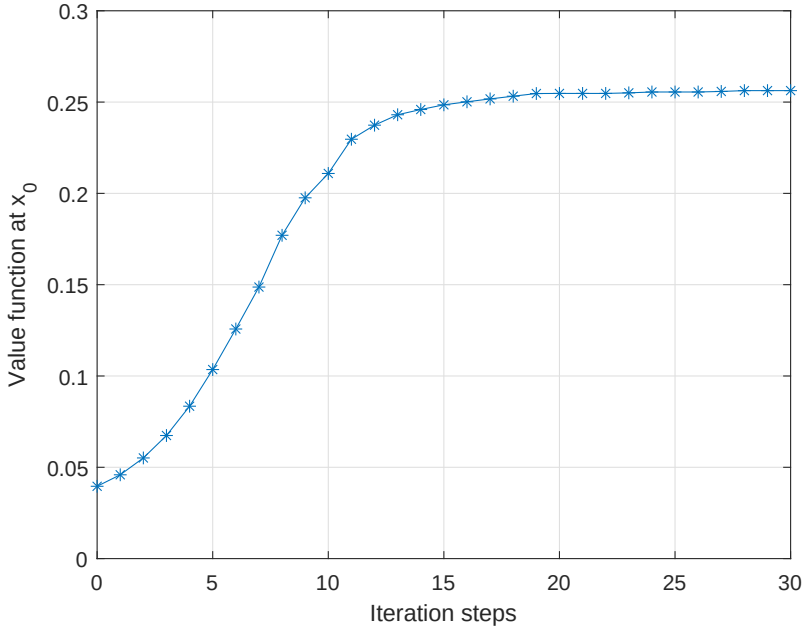

Fig. 2.1. Iterative value function at x_0 with $\bar{\alpha}_i^0(x_k)$.

as $\bar{P} = \begin{bmatrix} 0.1584 & -0.1065 & 0.0054 \\ -0.1065 & 0.0893 & -0.0020 \\ 0.0054 & -0.0020 & 0.033 \end{bmatrix}$. Let the initial state $x_0 = [0.4, 0, 0]^{\mathsf{T}}$. We choose two learning rate function sequences $\{\bar{\alpha}_i^{\zeta}(x_k)\}$, $\zeta = 0, 1$ and $i = 0, 1, \ldots$. Let $\bar{\alpha}_i^0(x_k) \equiv 1$, $i = 0, 1, \ldots$. For $y \geq 0$ and $z > 0$, let $\mathrm{fix}(y, z)$ denote the quotient of y/z. Define $\bar{\mathcal{C}}^i$ as the center points of the learning rate function $\bar{\alpha}_i^1(x_k)$. Let the coordinates of $\bar{\mathcal{C}}^i$ be expressed as $\bar{\mathcal{C}}^i = (-0.75 + 0.5\,\mathrm{fix}((\mathrm{rem}(i, 64))/16), -0.75 + 0.5\,\mathrm{fix}((\mathrm{rem}(i, 64))/4), -0.75 + 0.5\,\mathrm{rem}((\mathrm{rem}(i, 64))/4))$. For $i = 0, 1, \ldots$ and $x_k \in \Omega_x$, let the learning rate function $\bar{\alpha}_i^1(x_k)$ be expressed as

$$\bar{\alpha}_i^1(x_k) = 0.8\, e^{-\frac{\left(\|x_k - \bar{\mathcal{C}}^i\|\right)^2}{2\bar{\sigma}^2}}, \tag{2.89}$$

where $\bar{\sigma} = 0.5$. For $\bar{\alpha}_i^0(x_k)$, $i = 0, 1, \ldots$, the plot of the iterative value function at x_0, i.e., $V_i(x_0)$, is shown in Fig. 2.1. The iterative state and control trajectories with $\bar{\alpha}_i^0(x_k)$ is shown in Figs. 2.3(a)

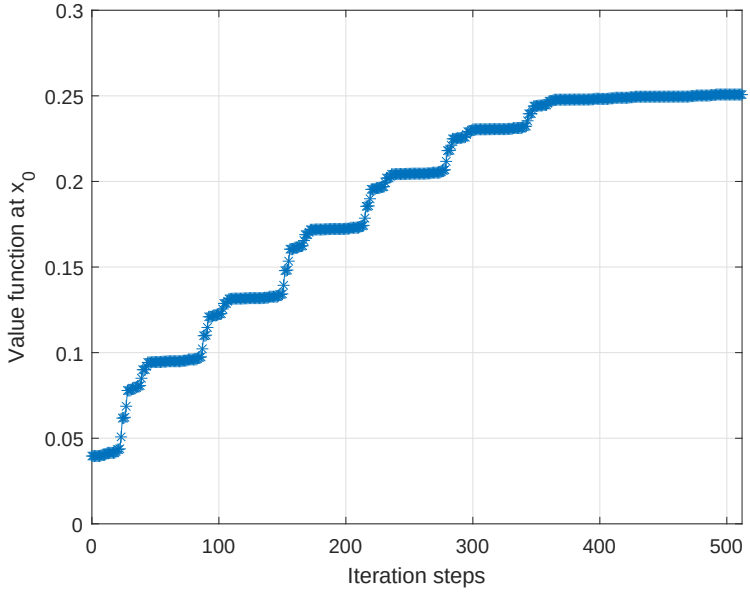

Fig. 2.2. Iterative value function at x_0 with $\bar{\alpha}_i^1(x_k)$.

and (b), where "In" means initial iteration and "Lm" means limiting iteration. The plot of the iterative value function at x_0 with $\bar{\alpha}_i^1(x_k)$ is shown in Fig. 2.2. The iterative state and control trajectories with $\bar{\alpha}_i^1(x_k)$ are shown in Figs. 2.3(c) and (d).

From Figs. 2.1–2.3, we can see that using the developed local iterative ADP algorithm, the iterative value function converges to the optimum. The optimal state and control are shown in Fig. 2.4. Despite initializing by the same initial value function, we can notice that the iterative processes are different with the two different learning rate functions. According to $\bar{\alpha}_i^0(x_k) = 1$ and $\bar{\Psi}(x_k)$, the local ADP algorithm is transformed into a global one [26]. The plots of the iterative value functions at x_0 is shown in Fig. 2.1. It takes 30 iterations to guarantee the iterative value function to be convergent. For the local iterative ADP algorithm with $\bar{\alpha}_i^1(x_k)$, the iterative value function is updated in a subset of the state space. Thus, in Fig. 2.2, we can see that the iterative value function is not necessarily updated in each iteration at x_0 and it takes 512 iterations to make the iterative value function converge to the optimum.

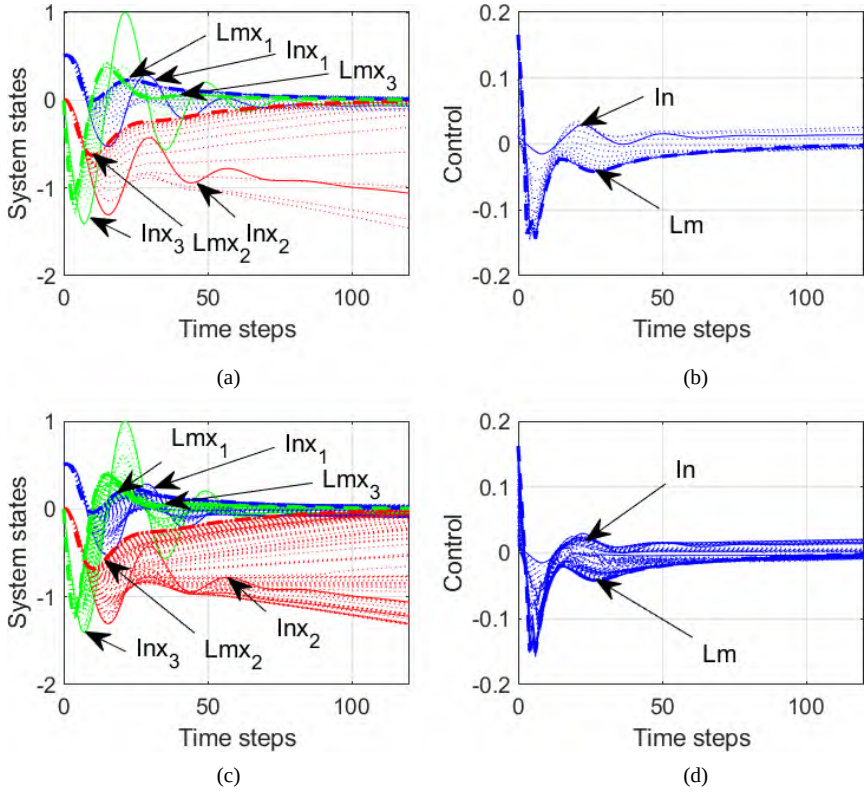

Fig. 2.3. State and control trajectories. (a) States with $\bar{\alpha}_i^0(x_k)$. (b) Controls with $\bar{\alpha}_i^0(x_k)$. (c) States with $\bar{\alpha}_i^1(x_k)$. (d) Controls with $\bar{\alpha}_i^1(x_k)$.

However, we point out that using the global iterative ADP algorithm in Ref. [26], in each iteration, the iterative value function and iterative control law are updated using $\bar{p} = 101^3$ state data. It takes 22901.51 seconds to train the critic and action networks in each iteration. Thus, we say that the computation burden is heavy for the global iterative ADP algorithm in Ref. [26]. On the other hand, for $\bar{\alpha}_i^1(x_k)$, there are 64 center points in the 3-dimensional state space, which separates the into state space into 64 state subsets. In each iteration, iterative value function and iterative control law are updated in one state subset using about $\bar{p}/64 = 16098$ state data, which takes 15 seconds to train the neural networks. Hence, comparing with the global iterative ADP algorithm in Ref. [26], we emphasize that the computation burden is relaxed using the developed local iterative

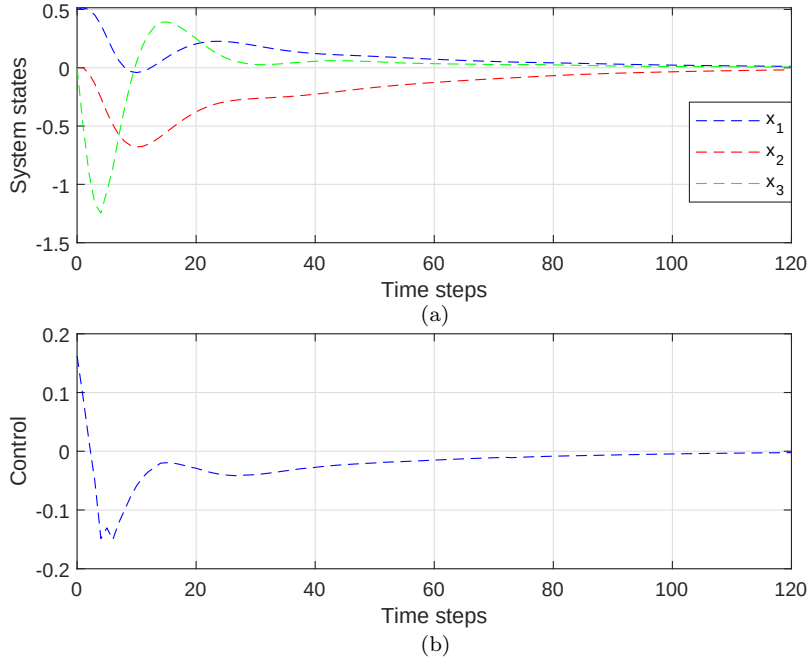

Fig. 2.4. Optimal trajectories. (a) Optimal states. (b) Optimal control.

ADP algorithm and this is a highlighted merit for the developed algorithm.

2.5 Conclusion

In this chapter, an effective local iterative ADP algorithm is developed to solve infinite horizon optimal control problems for discrete-time nonlinear systems. The developed local value iteration algorithm of ADP permits an arbitrary positive semi-definite function to initialize the algorithm. It is the first time that the concept of "local iteration" is introduced into iterative ADP algorithms. Employing a state-depended learning rate function, the iterative value function and iterative control law can both be updated in a given subset of the state space, instead of the whole state space for the traditional global iterative ADP algorithms. Under some mild convergence criteria, it is proven that the iterative value function is convergent to

the optimum. Monotonicity and optimality properties of the iterative value function are also presented. Simulation results are given to illustrate the performance of the developed algorithm.

References

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

[2] Al-Tamimi, A., Lewis, F.L. and Abu-Khalaf, M. (2008). Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38(4), 943–949.

[3] Bertsekas, D.P. (2007). *Dynamic Programming and Optimal Control*(3rd edn.), Athena Scientific, Belmont, MA.

[4] Bertsekas, D.P. and Tsitsiklis, J.N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

[5] Bian, T., Jiang, Y. and Jiang, Z.P. (2014). Adaptive dynamic programming and optimal control of nonlinear nonaffine systems. *Automatica* 50(10), 2624–2632.

[6] Cimen, T. and Banks, S.P. (2004). Global optimal feedback control for general nonlinear systems with nonquadratic performance criteria. *Systems & Control Letters* 53(5), 327–346.

[7] Jiang, Y. and Jiang, Z.P. (2015). Global adaptive dynamic programming for continuous-time nonlinear systems. *IEEE Transactions on Automatic Control* 60(11), 2917–2929.

[8] Kiumarsi, B. and Lewis, F.L. (2015). Actor–critic-based optimal tracking for partially unknown nonlinear discrete-time systems. *IEEE Transactions on Neural Networks and Learning Systems* 26(1), 140–151.

[9] Lewis, F.L., Vrabie, D. and Vamvoudakis, K.G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine* 32(6), 76–105.

[10] Lincoln, B. and Rantzer, A. (2006). Relaxing dynamic programming. *IEEE Transactions on Automatic Control* 51(8), 1249–1260.

[11] Liu, D. and Wei, Q. (2013). Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems. *IEEE Transactions on Cybernetics* 43(2), 779–789.

[12] Liu, D. and Wei, Q. (2014). Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems* 25(3), 621–634.

[13] Liu, D., Li, H. and Wang, D. (2014). Online synchronous approximate optimal learning algorithm for multi-player non-zero-sum games with unknown dynamics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44(8), 1015–1027.

[14] Liu, D., Wei, Q. and Yan, P. (2015). Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45(12), 1577–1591.

[15] Murray, J., Cox, C., Lendaris, G. and Saeks, R. (2002). Adaptive dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 32(2), 140–153.

[16] Ni, Z., He, H. and Wen, J. (2013). Adaptive learning in tracking control based on the dual critic network design. *IEEE Transactions on Neural Networks and Learning Systems* 24(6), 913–928.

[17] Ni, Z., He, H., Zhao, D., Xu, X. and Prokhorov, D.V. (2015). Grdhp: A general utility function representation for dual heuristic dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems* 26(3), 614–627.

[18] Si, J. and Wang, Y.T. (2001). Online learning control by association and reinforcement. *IEEE Transactions on Neural Networks* 12(2), 264–276.

[19] Song, R., Xiao, W., Zhang, H. and Sun, C. (2014). Adaptive dynamic programming for a class of complex-valued nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems* 25(9), 1733–1739.

[20] Song, R., Lewis, F., Wei, Q., Zhang, H.G., Jiang, Z.P. and Levine, D. (2015). Multiple actor-critic structures for continuous-time optimal control using input-output data. *IEEE Transactions on Neural Networks and Learning Systems* 26(4), 851–865.

[21] Song, R., Lewis, F.L., Wei, Q. and Zhang, H. (2016). Off-policy actor-critic structure for optimal control of unknown systems with disturbances. *IEEE Transactions on Cybernetics* 46(5), 1041–1050.

[22] Wang, D., Liu, D., Zhang, Q. and Zhao, D. (2016). Data-based adaptive critic designs for nonlinear robust optimal control with uncertain dynamics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46(11), 1544–1555.

[23] Wei, Q. and Liu, D. (2014). Data-driven neuro-optimal temperature control of water–gas shift reaction using stable iterative adaptive dynamic programming. *IEEE Transactions on Industrial Electronics* 61(11), 6399–6408.

[24] Wei, Q., Wang, F.Y., Liu, D. and Yang, X. (2014). Finite-approximation-error-based discrete-time iterative adaptive dynamic programming. *IEEE Transactions on Cybernetics* 44(12), 2820–2833.

[25] Wei, Q., Liu, D. and Yang, X. (2015). Infinite horizon self-learning optimal control of nonaffine discrete-time nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems* 26(4), 866–879.

[26] Wei, Q., Liu, D. and Lin, H. (2016). Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems. *IEEE Transactions on Cybernetics* 46(3), 840–853.

[27] Zhang, H., Wei, Q. and 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 Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38(4), 937–942.

[28] Zhang, H., Qin, C. and Luo, Y. (2014). Neural-network-based constrained optimal control scheme for discrete-time switched nonlinear system using dual heuristic programming. *IEEE Transactions on Automation Science and Engineering* 11(3), 839–849.

[29] Zhang, H., Zhang, J., Yang, G.H. and Luo, Y. (2015). Leader-based optimal coordination control for the consensus problem of multiagent differential games via fuzzy adaptive dynamic programming. *IEEE Transactions on Fuzzy Systems* 23(1), 152–163.

Chapter 3

Discrete-Time Local Value Iterative Adaptive Dynamic Programming: Admissibility and Terminations Analysis

3.1 Introduction

In 2008, a value iteration based iterative ADP algorithm [1], which was inspired by [3, 4], was proposed to solve the optimal control for a class of discrete-time nonlinear systems. In Ref. [1], considering the following deterministic discrete-time affine nonlinear system

$$x_{k+1} = f(x_k) + g(x_k)u_k, \tag{3.1}$$

with the performance index function

$$J(x_k) = \sum_{j=k}^{\infty} \Big(x_j^{\mathsf{T}} Q x_j + u_j^{\mathsf{T}} R u_j \Big), \tag{3.2}$$

where $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ are positive definite matrices, a value iteration algorithm, which was referred to as heuristic dynamic programming (HDP), was proposed to find the optimal control law. Starting from $V_0(x_k) \equiv 0$, for all $x_k \in \mathbb{R}^n$, we define

$$\begin{cases} u_i(x_k) = -\dfrac{1}{2} R^{-1} g(x_k)^{\mathsf{T}} \dfrac{\partial V_i(x_{k+1})}{\partial x_{k+1}}, \\ V_{i+1}(x_k) = x_k^{\mathsf{T}} Q x_k + u_i^{\mathsf{T}}(x_k) R u_i(x_k) + V_i(x_{k+1}), \end{cases} \tag{3.3}$$

for $i = 0, 1, 2, \ldots$, where $x_{k+1} = f(x_k) + g(x_k)u_i(x_k)$. It was proven that $V_i(x_k)$ is nondecreasing and bounded, and hence converged to $J^*(x_k)$ as i increases to infinity.

However, in previous iterative ADP algorithms, to guarantee the convergence and optimality, nearly all the previous iterative ADP algorithms require that the iterative value functions and iterative control laws are updated for the whole state space in each iteration [1, 3, 4, 6–8, 10, 15]. These iterative ADP algorithms can be called "global iterative ADP algorithms". It is worth mentioning that global iterative ADP algorithms may not be suitable for real-world applications. For example, for many real-world control systems, the system states are generally operated in a subset of the state space in an iteration period. Thus, we generally collect state data in a subset of the state space instead of the whole state space. In this situation, the global iterative ADP algorithms must stop running to wait until all the data for the whole state space are collected. This makes the computation efficiency of the global iterative ADP algorithms very low. On the other hand, previous global iterative ADP algorithms require the algorithms to implement infinite iterations to reach their optimums, which are impossible to realize in real-world applications. We say that the algorithms must be terminated within finite iterations to find an effective iterative control law to control the systems. Unfortunately, to the best of our knowledge, there are no discussions on the properties of the iterative control laws after the termination of the iterative ADP algorithms within finite iterations using local data. Hence, how to implement the iterative ADP algorithms using local data in each iteration and how to find an effective control law to control the systems within finite iterations are key problems for the developments of ADP algorithms. These motivate our research.

In this chapter, a novel local iterative ADP algorithm is developed to solve infinite horizon optimal control problems of discrete-time nonlinear systems. A new state-dependent learning rate function is employed in the iterative ADP algorithm, where the iterative value function and iterative control law can both be updated in a given subset of the state space, instead of updating in the whole state space. Initialized by an arbitrary positive semi-definite function, it is shown that the iterative value function converges to the optimum under some mild constraints. We emphasize that for the first time, the admissibility properties for the local iterative ADP algorithm

are developed. New termination criteria are established for the local iterative ADP algorithm. New convergence criteria are established to guarantee that the iterative value function is close to the optimum sufficiently. Admissibility criteria are constructed which guarantee the admissibility of the iterative control law. It will be shown that the iterative ADP algorithm can be terminated if the convergence and admissibility criteria are both satisfied. Finally, simulation results are given to illustrate the performance of the developed method.

3.2 Problem Formulations

3.2.1 *System descriptions*

In this chapter, we will study the following deterministic discrete-time nonlinear system

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \dots, \tag{3.4}$$

where $x_k \in \mathbb{R}^n$ is the state vector and $u_k \in \mathbb{R}^m$ is the 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 ∞. The performance index function for state x_0 under the control sequence $\underline{u}_0 = (u_0, u_1, \dots)$ is defined as

$$J(x_0, \underline{u}_0) = \sum_{k=0}^{\infty} U(x_k, u_k), \tag{3.5}$$

where the utility function $U(x_k, u_k)$ is a positive definite function for x_k and u_k.

For convenience of analysis, results of this chapter are based on the following assumption.

Assumption 3.1. The system (3.4) is controllable on a compact set $\Omega_x \subset \mathbb{R}^n$ containing the origin, and the function $F(x_k, u_k)$ is Lipschitz continuous on Ω_x; the system state $x_k = 0$ is an equilibrium state of system (3.4) under the control $u_k = 0$, i.e., $F(0, 0) = 0$; the feedback control $u_k = u(x_k)$ satisfies $u_k = u(x_k) = 0$ for $x_k = 0$.

Define the control sequence set as $\underline{\mathfrak{U}}_k = \{\underline{u}_k \colon \underline{u}_k = (u_k, u_{k+1}, \dots), \forall u_{k+i} \in \mathbb{R}^m, i = 0, 1, \dots\}$. Then, for an arbitrary control sequence

$\underline{u}_k \in \underline{\mathfrak{U}}_k$, the optimal performance index function can be defined as $J^*(x_k) = \min_{\underline{u}_k}\{J(x_k, \underline{u}_k) : \underline{u}_k \in \underline{\mathfrak{U}}_k\}$. According to Bellman's principle of optimality, $J^*(x_k)$ satisfies the discrete-time HJB equation $J^*(x_k) = \min_{u_k}\{U(x_k, u_k) + J^*(F(x_k, u_k))\}$. Define the law of optimal control as $u^*(x_k) = \arg\min_{u_k}\{U(x_k, u_k) + J^*(F(x_k, u_k))\}$. Hence, the HJB equation can be written as

$$J^*(x_k) = U(x_k, u^*(x_k)) + J^*(F(x_k, u^*(x_k))). \tag{3.6}$$

Generally, the HJB equation (3.6) is computationally untenable by traditional dynamic programming, i.e., "curse of dimensionality." To overcome this difficulty, a new iterative ADP algorithm will be developed.

3.2.2 *Descriptions of the local iterative ADP algorithm*

In this section, a new local iterative ADP algorithm is developed to obtain the optimal control law for the nonlinear system (3.4). In the local iterative ADP algorithm, the value function and control law are updated by iteration, with the iteration index i increasing from 0 to infinity. For $i = 0, 1, \ldots$, define $\{\mathscr{L}_x^i\}$ as a sequence of state sets, which satisfies $\mathscr{L}_x^i \subseteq \Omega_x$, $\forall\, i = 0, 1, \ldots$.

Let the initial function $\Psi(x_k)$, $\forall x_k \in \Omega_x$, be an arbitrary positive semi-definite function and let the initial value function be expressed by

$$V_0(x_k) = \Psi(x_k). \tag{3.7}$$

For all $x_k \in \mathscr{L}_x^0$, the local iterative control law $v_0(x_k)$ is computed as

$$\begin{aligned} v_0(x_k) &= \arg\min_{u_k}\{U(x_k, u_k) + V_0(x_{k+1})\} \\ &= \arg\min_{u_k}\{U(x_k, u_k) + \Psi(F(x_k, u_k))\}, \end{aligned} \tag{3.8}$$

and let $v_0(x_k) = 0$, for all $x_k \in \Omega_x \backslash \mathscr{L}_x^0$. For all $x_k \in \mathscr{L}_x^0$, the local iterative value function is updated as

$$\begin{aligned} \mathcal{V}_1(x_k) &= \min_{u_k}\big\{U(x_k, u_k) + V_0(x_{k+1})\big\} \\ &= U(x_k, v_0(x_k)) + V_0(F(x_k, v_0(x_k))), \end{aligned} \tag{3.9}$$

and let $\mathcal{V}_1(x_k) = \Psi(x_k)$, for all $x_k \in \Omega_x \backslash \mathscr{L}_x^0$. Define $\alpha_0(x_k)$ as a state-dependent learning rate function. For all $x_k \in \Omega_x$, let the learning rate function $\alpha_0(x_k)$ be a scalar function, which satisfies

$$\begin{cases} 0 < \alpha_0(x_k) \leq 1, & \forall x_k \in \mathscr{L}_x^0, \\ \alpha_0(x_k) = 0, & \forall x_k \in \Omega_x \backslash \mathscr{L}_x^0. \end{cases} \tag{3.10}$$

Then, for all $x_k \in \Omega_x$, the global iterative value function is defined as

$$V_1(x_k) = (1 - \alpha_0(x_k))V_0(x_k) + \alpha_0(x_k)\mathcal{V}_1(x_k). \tag{3.11}$$

For $i = 1, 2, \ldots$, for all $x_k \in \mathscr{L}_x^i$, the local iterative control law $v_i(x_k)$ can be computed as

$$\begin{aligned} v_i(x_k) &= \arg\min_{u_k} \{U(x_k, u_k) + V_i(x_{k+1})\} \\ &= \arg\min_{u_k} \{U(x_k, u_k) + V_i(F(x_k, u_k))\}, \end{aligned} \tag{3.12}$$

and for all $x_k \in \Omega_x \backslash \mathscr{L}_x^i$, let $v_i(x_k) = v_{i-1}(x_k)$. For all $x_k \in \mathscr{L}_x^i$, the local iterative value function can be updated as

$$\begin{aligned} \mathcal{V}_{i+1}(x_k) &= \min_{u_k} \left\{U(x_k, u_k) + V_i(x_{k+1})\right\} \\ &= U(x_k, v_i(x_k)) + V_i(F(x_k, v_i(x_k))), \end{aligned} \tag{3.13}$$

and for all $x_k \in \Omega_x \backslash \mathscr{L}_x^i$, let $\mathcal{V}_{i+1}(x_k) = V_i(x_k)$. Define $\{\alpha_i(x_k)\}$, $i = 1, 2, \ldots$, as a sequence of learning rate functions. For $i = 1, 2, \ldots$ and $\forall x_k \in \Omega_x$, let the learning rate function $\alpha_i(x_k)$ be a scalar function, which satisfies

$$\begin{cases} 0 < \alpha_i(x_k) \leq 1, & x_k \in \mathscr{L}_x^i, \\ \alpha_i(x_k) = 0, & \forall x_k \in \Omega_x \backslash \mathscr{L}_x^i. \end{cases} \tag{3.14}$$

Then, for all $x_k \in \Omega_x$, the global iterative value function is defined as

$$V_{i+1}(x_k) = (1 - \alpha_i(x_k))V_i(x_k) + \alpha_i(x_k)\mathcal{V}_{i+1}(x_k). \tag{3.15}$$

Introducing a state-dependent learning rate function $\alpha_i(x_k)$, from (3.7)–(3.15), in each iteration, the iterative value function and iterative control law are only updated in the given state subset $\mathscr{L}_x^i$,

$i = 0, 1, \ldots$, instead of the whole state space. On the other hand, if the learning rate function $\alpha_i(x_k)$ satisfies $\alpha_i(x_k) \equiv 1$, $\forall\, i = 0, 1, \ldots$, and $V_0(x_k) \equiv 0$, $\forall x_k \in \Omega_x$, then the local iterative ADP algorithm reduces to the traditional value iteration ADP algorithms [1, 3, 4, 6–8, 10, 15]. Thus, we say that traditional value iteration ADP algorithms are special cases of the developed local iterative one and the local iterative ADP algorithm in this chapter possesses more application potentials.

3.3 Property Analysis

In this section, the properties of the local iterative ADP algorithm will be discussed. First, we will show that the iterative value iteration converges to the optimum under some mild constraints. Then, the main contributions of this paper, that is the admissibility of the iterative control law, will be discussed. New termination criteria of the local iterative ADP algorithm are derived, where the convergence of the iterative value function and admissibility of the iterative control law can both be guaranteed.

To derive the main theoretical results of this chapter, the following lemma is necessary.

Lemma 3.1. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let iterative value function $V_i(x_k)$ and the iterative control law $v_i(x_k)$ be obtained by* (3.7)–(3.15). *Then,*

(i) *for $i = 0, 1, \ldots, V_i(x_k)$ is positive semi-definite for x_k;*
(ii) *for $i = 0, 1, \ldots, V_i(x_k)$ is positive definite for x_k if $\Psi(x_k)$ is positive definite;*
(iii) *if $V_i(x_k)$, $i = 0, 1, \ldots$, is positive definite for x_k, then $V_j(x_k)$ is positive definite for all $j \geq i$;*
(iv) *if for any $i = 0, 1, \ldots$ and $0 \leq j \leq \vartheta$, $\vartheta = 0, 1, \ldots$, the state set $\mathscr{L}_x^{i+j}$ satisfies*

$$\mathscr{L}_x^i \cup \mathscr{L}_x^{i+1} \cup \cdots \cup \mathscr{L}_x^{\vartheta} = \bigcup_{j=0}^{\vartheta} \mathscr{L}_x^{i+j} = \Omega_x, \tag{3.16}$$

then the iterative value function $V_\ell(x_k)$ is a positive definite function for all $\ell \geq i + \vartheta$.

The convergence property of the local iterative ADP algorithm can be derived in the following lemma.

Lemma 3.2. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let the iterative control law $v_i(x_k)$ and iterative value function $V_i(x_k)$ be obtained by* (3.7)–(3.15). *If for any $i = 0, 1, \ldots$ and $\forall\, x_k \in \Omega_x$, the learning rate $\alpha_i(x_k)$ satisfies* (3.10), (3.14) *and*

$$\sum_{i=0}^{\infty} \alpha_i(x_k) = \infty, \forall x_k \neq 0, \tag{3.17}$$

then the iterative value function $V_i(x_k)$ converges to the optimal performance index function $J^*(x_k)$, as $i \to \infty$.

From Lemma 3.2, we know that under the condition (3.17), the iterative value function converges to $J^*(x_k)$, when the iteration index i increases to the infinity. As $J^*(x_k)$ is a positive definite function, the developed algorithm cannot stop before it outputs a positive definite function. Define a new parameter ρ as $\rho = \arg\min_\theta\{\cup_{j=0}^{\theta} \mathscr{L}_x^j = \Omega_x\}$. Letting

$$i_{\min} = \begin{cases} 0, & \text{if } \Psi(x_k) > 0, \\ \rho, & \text{if } \Psi(x_k) \geq 0, \end{cases} \tag{3.18}$$

according to Lemma 3.1(iv), we know that for any $i = 0, 1, \ldots,$ if $i \geq i_{\min}$, then the iterative value function $V_i(x_k)$ is positive definite. Thus, $i_{\min}$ is called the minimum iteration condition. Before the analysis, the following definition is necessary.

Definition 3.1. We say a solution is uniformly ultimately bounded (UUB) [5] if there exists a compact set $\Omega_x \subset \mathbb{R}^n$, such that for all $x_{k_0} = x_0 \in \Omega_x$, there exists an ε and a number $T(\varepsilon, x_0)$ such that $\|x_k\| \leq \varepsilon$ for all $k \geq k_0 + T$.

Generally, if for $j = 0, 1, \ldots, \vartheta$, $\forall\, x_k \in \mathscr{L}_x^i$, the iteration index $i \geq i_{\min}$, and the iterative value function satisfies

$$|V_{i+j+1}(x_k) - V_{i+j}(x_k)| \leq \alpha_{i+j}(x_k)\varepsilon, \tag{3.19}$$

for a constant $\varepsilon > 0$, where $i_{\min}$ is defined in (3.18) and $\mathscr{L}_x^{i+j}$ satisfies (3.16), then the algorithm is terminated and the optimal control law is regarded to achieve. Inequality (3.19) can be called as the "convergence criterion" of the local iterative ADP method. In real-world applications, the computation precision ε cannot be zero. For a

computation precision $\varepsilon > 0$, we actually obtain an iterative value function and iterative control law, which approximate their optimal ones. In this situation, we emphasize that the iterative control law may not be a stable control law. The following theorem will show this property.

Theorem 3.1. *For $i = 0, 1, \ldots$, let $V_i(x_k)$ and $v_i(x_k)$ be obtained by* (3.7)–(3.15). *Let the iteration index i satisfy $i \geq i_{\min}$, where $i_{\min}$ is defined in* (3.18), *and let the learning rate function $\alpha_i(x_k)$ satisfy* (3.14) *and* (3.17). *If for any $j = 0, 1, \ldots$, the iterative value function $V_{i+j}(x_k)$ satisfies* (3.19) *then the state of the nonlinear system* (3.4) *is uniformly ultimately bounded* (*UUB*) *under the iterative control law $v_{i+\vartheta}(x_k)$.*

Proof. According to the definition of $\mathscr{L}_x^{i+j}$ in (3.16), for a system state $x_k \in \Omega_x$, the state may be in several different state sets, i.e., $x_k \in \mathscr{L}_x^{i+j'}$, $0 \leq j' \leq \vartheta$, and simultaneously $x_k \in \mathscr{L}_x^{i+j''}$, $0 \leq j'' \leq \vartheta$, where $j' \neq j''$. It means that several iterative value functions, such as $V_{i+j'+1}(x_k)$ and $V_{i+j''+1}(x_k)$, may be updated at x_k. This leads to a difficulty in analysis. For convenience of analysis, letting $\bar{\mathscr{L}}_x^{i+j} = \Omega_x \backslash \mathscr{L}_x^{i+j}$, $i \geq i_{\min}$ and $j = 0, 1, \ldots, \vartheta$, we define a new state set $\mathcal{I}_{i+j}$ as

$$\mathcal{I}_x^{i+j} = \left(\bigcap_{l=j+1}^{\vartheta} \bar{\mathscr{L}}_x^{i+l} \right) \cap \mathscr{L}_x^{i+j}, \tag{3.20}$$

where we let $\cap_{l=j+1}^{\vartheta} \bar{\mathscr{L}}_x^{i+l} = \Omega_x$ for $j = \vartheta$, i.e., $\mathcal{I}_x^{i+\vartheta} = \mathscr{L}_x^{i+\vartheta}$.

According to the definition in (3.20), we can see that $\mathcal{I}_{i+j}$, $j = 0, 1, \ldots, \vartheta$, satisfies the following equation $\cup_{l=j+1}^{\vartheta} \mathcal{I}_x^{i+l} = \mathscr{L}_x^{i+j} \cup \mathscr{L}_x^{i+j+1} \cup \cdots \cup \mathscr{L}_x^{i+\vartheta} = \cup_{l=j}^{\vartheta} \mathscr{L}_x^{i+l}$, and for $j_0, j_1 = 0, 1, \ldots, \vartheta$, we can get $\mathcal{I}_x^{i+j_0} \cap \mathcal{I}_x^{i+j_1} = \emptyset, \forall j_0 \neq j_1$. If $\mathscr{L}_x^{i+j}$, $j = 0, 1, \ldots$, satisfies (3.16) then we can derive that $\cup_{j=0}^{\vartheta} \mathcal{I}_x^{i+j} = \Omega_x$. According to the definition of $\mathcal{I}_x^{i+j}$ in (3.20) and the definition of the iterative control law in (3.8) and (3.12), for any $j = 0, 1, \ldots, \vartheta$, we can get

$$v_{i+\vartheta}(x_k) = v_{i+j}(x_k), \quad \forall x_k \in \mathcal{I}_x^{i+j}. \tag{3.21}$$

Let $\ell_0, \ell_1 = 0, 1, \ldots, \vartheta$ be two constants. According to (3.19), for any $x_k \in \Omega_x$ and $\ell_0, \ell_1 = 0, 1, \ldots, \vartheta$, there exists a constant $\lambda > 0$

which satisfies $|V_{i+\ell_0}(x_k) - V_{i+\ell_1}(x_k)| \leq \lambda$, where $0 \leq \lambda \leq \vartheta\varepsilon$. For any $i \geq \rho$, the iterative value function $V_i(x_k)$ is a positive definite function for x_k. For any $j = 0, 1, \ldots$, there exist two scalar functions $\underline{\beta}(\|x_k\|)$ and $\overline{\beta}(\|x_k\|)$ which belong to class $\mathcal{K}$ and satisfy

$$\begin{aligned} 0 < \underline{\beta}(\|x_k\|) &\leq \inf_{j=0,1,\ldots,\vartheta} V_{i+j}(x_k) \leq V_{i+j}(x_k) \\ &\leq \sup_{j=0,1,\ldots,\vartheta} V_{i+j}(x_k) \leq \overline{\beta}(\|x_k\|). \end{aligned} \tag{3.22}$$

Define a new state set Λ_x as

$$\begin{aligned} \Lambda_x = \{x_k \,|\, & x_k \in \Omega_x, U(x_k, v_i(x_k)) \leq \varepsilon + 2\lambda, \\ & U(x_k, v_{i+1}(x_k)) \leq \varepsilon + 2\lambda, \\ & \ldots, U(x_k, v_{i+\vartheta}(x_k)) \leq \varepsilon + 2\lambda\}. \end{aligned} \tag{3.23}$$

Let $\tilde{x}_k = \arg\sup_{x_k \in \Lambda_x}\{\|x_k\|\}$, and we can see that $\|\tilde{x}_k\|$ is finite. Let $\Gamma_x = \{x_k \,|\, x_k \in \Omega_x, \underline{\beta}(x_k) \leq \bar{\beta}(\tilde{x}_k)\}$. Let $\bar{x}_k = \arg\sup_{x_k \in \Gamma_x}\{\|x_k\|\}$. For an arbitrary $\epsilon > \|\bar{x}_k\|$, there exists a $\varphi(\epsilon) > \|\tilde{x}_k\|$ that satisfies $\overline{\beta}(\varphi) \leq \underline{\beta}(\epsilon)$. According to the definition of the function $\overline{\beta}(\cdot)$, there exits a state x_k, such that $\|\tilde{x}_k\| \leq \|x_k\| \leq \varphi(\epsilon)$. For any $x_k \in \Omega_x$, according to (3.20), the state x_k must be located in one state set $\mathcal{I}_{i+j}$, $j = 0, 1, \ldots, \vartheta$. Without loss of generality, we assume that $x_k \in \mathcal{I}_{i+j_0}$, $j_0 = 0, 1, \ldots, \vartheta$.

For $x_k \in \mathcal{I}_x^{j_0}$, we have $V_{i+j_0}(x_k) \leq \overline{\beta}(\varphi)$. According to (3.11), (3.15), and (3.19), we have

$$V_{i+j_0+1}(x_k) - V_{i+j_0}(x_k) \leq \alpha_{i+j_0}(x_k)\varepsilon. \tag{3.24}$$

If $\|x_k\| \geq \|\tilde{x}_k\|$, then according to (3.19) and (3.21), for $x_k \in \mathcal{I}_x^{i+j_0}$, we have

$$\begin{aligned} \Delta V_{i+j_0}(x_k) &= V_{i+j_0}(x_{k+1}) - V_{i+j_0}(x_k) \\ &= V_{i+j_0}(F(x_k, v_{i+j_0}(x_k))) - V_{i+j_0}(x_k) \\ &= V_{i+j_0}(F(x_k, v_{i+\vartheta}(x_k))) - V_{i+j_0}(x_k) \\ &= -U(x_k, v_{i+\vartheta}(x_k)) + \varepsilon \\ &\leq -2\lambda \leq 0. \end{aligned} \tag{3.25}$$

Assume $x_{k+1} = F(x_k, v_{i+\vartheta}(x_k)) \in \mathcal{I}_x^{i+j_1}$. If $\|x_{k+1}\| \geq \|\tilde{x}_k\|$, according to (3.19) and (3.21), for $x_{k+1} \in \mathcal{I}_x^{i+j_0}$, we can obtain

$$\begin{aligned}\Delta V_{i+j_1}(x_{k+1}) &= V_{i+j_1}(x_{k+2}) - V_{i+j_1}(x_{k+1}) \\ &= V_{i+j_1}(F(x_{k+1}, v_{i+j_1}(x_{k+1}))) - V_{i+j_1}(x_{k+1}) \\ &= V_{i+j_1}(F(x_{k+1}, v_{i+\vartheta}(x_{k+1}))) - V_{i+j_1}(x_{k+1}) \\ &\leq -2\lambda. \end{aligned} \tag{3.26}$$

For any $x_{k+1} \in \Omega_x$, we can get $V_{i+j_1}(x_{k+1}) \leq V_{i+j_0}(x_{k+1}) + \lambda$. Then, we have

$$V_{i+j_1}(x_{k+2}) \leq V_{i+j_0}(x_{k+1}) - \lambda, \tag{3.27}$$

where $x_{k+2} = F(x_{k+1}, v_{i+\vartheta}(x_{k+1}))$. On the other hand, for all $x_k \in \Omega_x$, we can obtain $V_{i+j_1}(x_{k+2}) \geq V_{i+j_0}(x_{k+2}) - \lambda$. Then, for $x_{k+1} \in \mathcal{I}_x^{i+j_1}$, we can obtain that

$$V_{i+j_0}(x_{k+2}) - V_{i+j_0}(x_{k+1}) \leq 0. \tag{3.28}$$

According to mathematical induction, for any $T = 0, 1, \ldots$, if $\|x_{k+T}\| \geq \|\tilde{x}_k\|$, there exists $j_T = 0, 1, \ldots, \vartheta$, such that for any $x_{k+T} \in \mathcal{I}_x^{i+j_T}$, the iterative value function $V_{i+j_0}(x_{k+T})$ satisfies

$$V_{i+j_0}(x_{k+T+1}) - V_{i+j_0}(x_{k+T}) \leq 0. \tag{3.29}$$

For any ϵ that satisfies $\epsilon \geq \|\bar{x}_k\|$, there exists a positive integer N that satisfies the following inequality

$$\begin{aligned}\underline{\beta}(\epsilon) \geq \bar{\beta}(\varphi) &\geq V_{i+j_0}(x_k) \geq V_{i+j_0}(x_{k+1}) \\ &\geq \cdots \geq V_{i+j_0}(x_{k+N}) \geq \underline{\beta}(\|x_{k+N}\|) \end{aligned} \tag{3.30}$$

for $\|x_{k+N}\| \geq \|\tilde{x}_k\|$, which makes $\|x_{k+N}\| \to \|\tilde{x}_k\|$. According to the Definition 3.1, we can draw the conclusion. The proof is complete. □

From Theorem 3.1, we can see that for any constant $\varepsilon > 0$, the iterative control law $v_{i+\vartheta}(x_k)$ is UUB, which is not ensured to be an asymptotically stable control law. For optimal control problems, the developed control scheme must not only stabilize the control systems, but also make the performance index function finite, i.e., the control law must be admissible [1].

Definition 3.2. A control law $u(x_k)$ is defined to be admissible with respect to (3.5) on Ω_x if $u(x_k)$ is continuous on Ω_x, $u(0) = 0$, $u(x_k)$ stabilizes (3.4) on Ω_x, and $\forall\, x_0 \in \Omega_x$, $J(x_0)$ is finite.

Theorem 3.2. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let $v_i(x_k)$ and $V_i(x_k)$ be obtained by* (3.7)–(3.15). *Let the iteration index $i \geq i_{\min}$, where $i_{\min}$ is defined in* (3.18) *and $\mathscr{L}_x^{i+j}$ satisfies* (3.16). *If the iterative value function $V_{i+j}(x_k)$, $j = 0, 1, \ldots, \vartheta$, satisfies*

(i) *for all $x_k \in \mathscr{L}_x^{i+j}$,*

$$V_{i+j+1}(x_k) - V_{i+j}(x_k) < \alpha_{i+j}(x_k) p_{i+j}(x_k) U(x_k, v_{i+j}(x_k)), \tag{3.31}$$

where $-\infty < p_{i+j}(x_k) < 1$,

(ii) *for arbitrary iteration indices $l, w = 0, 1, \ldots \vartheta$ and $\forall x_k \in \mathscr{L}_x^{i+l}$,*

$$V_{i+l}(x_k) - V_{i+\omega}(x_k) < q_{i+l}(x_k) U(x_k, v_{i+l}(x_k)), \tag{3.32}$$

where $-\infty < q_{i+l}(x_k) < 1$ and $-\infty < p_{i+l}(x_k) + q_{i+l}(x_k) < 1$,

then iterative control law $v_{i+\vartheta}(x_k)$ is an admissible control law.

Proof. According to (3.20), for an arbitrary state $x_k \in \Omega_x$, the state x_k must locate in one and only one state set $\mathcal{I}_x^{i+j}$, $j = 0, 1, \ldots$. Without loss of generality, we assume $x_k \in \mathcal{I}_x^{i+j_0}$, $j_0 = 0, 1, \ldots, \vartheta$. According to (3.31), we have

$$V_{i+j_0+1}(x_k) - V_{i+j_0}(x_k) < \alpha_{i+j_0}(x_k) p_{i+j_0}(x_k) U(x_k, v_{i+j_0}(x_k)). \tag{3.33}$$

For any $x_k \in \mathcal{I}_x^{i+j_0}$, we have $0 < \alpha_{i+j_0}(x_k) \leq 1$. According to (3.13) and (3.21), we can get

$$\begin{aligned} V_{i+j_0}(x_{k+1}) - V_{i+j_0}(x_k) &< (p_{i+j_0}(x_k) - 1) U(x_k, v_{i+j_0}(x_k)) \\ &= (p_{i+j_0}(x_k) - 1) U(x_k, v_{i+\vartheta}(x_k)), \end{aligned} \tag{3.34}$$

where $v_{i+j_0}(x_k) = v_{i+\vartheta}(x_k)$, $\forall x_k \in \mathcal{I}_x^{i+j_0}$, and $x_{k+1} = F(x_k, v_{i+j_0}(x_k)) = F(x_k, v_{i+\vartheta}(x_k))$. Assume $x_{k+1} \in \mathcal{I}_x^{i+j_1}$, $j_1 = 0, 1, \ldots, \vartheta$.

According to (3.32), as $l, \omega = 0, 1, \ldots, \vartheta$ are arbitrary constants, letting $l = j_1$ and $\omega = j_0$, for all $x_k \in \mathcal{I}_x^{i+j_1}$, we can get

$$\begin{aligned} &V_{i+j_1}(x_{k+1}) - V_{i+j_0}(x_{k+1}) \\ &\quad < q_{i+j_1}(x_{k+1}) U(x_{k+1}, v_{i+j_1}(x_{k+1})) \\ &\quad = q_{i+j_1}(x_{k+1}) U(x_{k+1}, v_{i+\vartheta}(x_{k+1})). \end{aligned} \tag{3.35}$$

For $x_{k+1} \in \mathcal{I}_x^{i+j_1}$, according to (3.31), we can obtain

$$\begin{aligned} &V_{i+j_1}(x_{k+2}) - V_{i+j_1}(x_{k+1}) \\ &\quad < (p_{i+j_1}(x_{k+1}) - 1) U(x_{k+1}, v_{i+j_1}(x_{k+1})) \\ &\quad = (p_{i+j_1}(x_{k+1}) - 1) U(x_{k+1}, v_{i+\vartheta}(x_{k+1})). \end{aligned} \tag{3.36}$$

According to (3.35) and (3.36), for $x_{k+1} \in \mathcal{I}_x^{i+j_1}$ we have

$$\begin{aligned} &V_{i+j_1}(x_{k+2}) - V_{i+j_0}(x_{k+1}) \\ &\quad < (p_{i+j_1}(x_{k+1}) + q_{i+j_1}(x_{k+1}) - 1) U(x_{k+1}, v_{i+\vartheta}(x_{k+1})), \end{aligned} \tag{3.37}$$

where $x_{k+2} = F(x_{k+1}, v_{i+j_1}(x_{k+1})) = F(x_{k+1}, v_{i+\vartheta}(x_{k+1}))$.

Using mathematical induction, for any $\tau = 0, 1, \ldots$ and $x_{k+\tau+1} \in \mathcal{I}_x^{i+j_{\tau+1}}$, we can obtain

$$\begin{aligned} &V_{i+j_{\tau+1}}(x_{k+\tau+2}) - V_{i+j_\tau}(x_{k+\tau+1}) \\ &\quad < (p_{i+j_{\tau+1}}(x_{k+\tau+1}) + q_{i+j_{\tau+1}}(x_{k+\tau+1}) - 1) \\ &\qquad \times U(x_{k+\tau+1}, v_{i+j_{\tau+1}}(x_{k+\tau+1})) \\ &\quad = (p_{i+j_{\tau+1}}(x_{k+\tau+1}) + q_{i+j_{\tau+1}}(x_{k+\tau+1}) - 1) \\ &\qquad \times U(x_{k+\tau+1}, v_{i+\vartheta}(x_{k+\tau+1})), \end{aligned} \tag{3.38}$$

where $x_{k+\tau+2} = F(x_{k+\tau+1}, v_{i+j_{\tau+1}}(x_{k+\tau+1})) = F(x_{k+\tau+1}, v_{i+\vartheta}(x_{k+\tau+1}))$. According to (3.38), for any $x_k \in \mathcal{I}_x^{i+j_0}$ and any $\tau =$

$0, 1, \ldots,$ we can obtain

$$\begin{aligned} & V_{i+j_{\tau+1}}(x_{k+\tau+2}) - V_{i+j_0}(x_{k+1}) \\ & \quad < \sum_{l=0}^{\tau}((p_{i+j_{l+1}}(x_{k+l+1}) + q_{i+j_{l+1}}(x_{k+l+1}) - 1) \\ & \qquad \times U(x_{k+l+1}, v_{i+j_{l+1}}(x_{k+l+1}))) \\ & \quad = \sum_{l=0}^{\tau}((p_{i+j_{l+1}}(x_{k+l+1}) + q_{i+j_{l+1}}(x_{k+l+1}) - 1) \\ & \qquad \times U(x_{k+l+1}, v_{i+\vartheta}(x_{k+l+1}))), \end{aligned} \tag{3.39}$$

where $x_{k+l+1} = F(x_{k+l}, v_{i+j_l}(x_{k+l})) = F(x_{k+l}, v_{i+\vartheta}(x_{k+l}))$, $l = 0, 1, \ldots, \tau+1$. According to (3.34) and (3.39), we can get

$$\begin{aligned} V_{i+j_0}(x_k) > & (1 - p_{i+j_0}(x_k))U(x_k, v_{i+\vartheta}(x_k)) \\ & + \sum_{l=0}^{\tau}((1 - p_{i+j_{l+1}}(x_{k+l+1}) - q_{i+j_{l+1}}(x_{k+l+1})) \\ & \times U(x_{k+l+1}, v_{i+\vartheta}(x_{k+l+1}))) \\ & + V_{i+j_{\tau+1}}(x_{k+\tau+2}). \end{aligned} \tag{3.40}$$

As $0 < q_{i+j_0}(x_k) < 1$ and $p_{i+j_0}(x_k) + q_{i+j_0}(x_k) < 1$, $\forall\, x_k \in \mathscr{L}_x^{i+j_0}$, we have

$$\begin{aligned} & (1 - p_{i+j_0}(x_k))\, U(x_k, v_{i+j_0}(x_k)) \\ & \quad > (1 - p_{i+j_0}(x_k) - q_{i+j_0}(x_k))U(x_k, v_{i+j_0}(x_k)). \end{aligned} \tag{3.41}$$

Thus, inequality (3.40) can be expressed as

$$\begin{aligned} V_{i+j_0}(x_k) > & \sum_{l=0}^{\tau+1}((1 - p_{i+j_l}(x_{k+l}) - q_{i+j_l}(x_{k+l})) \\ & \times U(x_{k+l}, v_{i+\vartheta}(x_{k+l}))) \\ & + V_{i+j_{\tau+1}}(x_{k+\tau+2}). \end{aligned} \tag{3.42}$$

Letting $\tau \to \infty$, as $V_{i+j_0}(x_k)$ is finite $\forall\, x_k \in \mathcal{I}_x^{i+j_0}$, we have

$$\lim_{\tau\to\infty}\left(\sum_{l=0}^{\tau+1}((1-p_{i+j_l}(x_{k+l})-q_{i+j_l}(x_{k+l}))\right.$$
$$\left.\times U(x_{k+l}, v_{i+\vartheta}(x_{k+l})))\right) < \infty. \tag{3.43}$$

According to the definitions of the functions $p_{i+j}(x_k)$ and $q_{i+j}(x_k)$, $j = 0, 1, \ldots, \vartheta$, in (3.31) and (3.32), for any $l = 0, 1, \ldots,\ j_l = 0, 1, \ldots, \vartheta$, there exits constants $0 < p < 1$ and $0 < q < 1$, such that $1-p_{i+j_l}(x_{k+l})-q_{i+j_l}(x_{k+l}) > 1-p-q > 0$ holds for all $x_{k+l} \in \mathcal{I}_x^{i+j_l}$. According to (3.43), we have

$$\begin{aligned}
&(1-p-q)\sum_{l=0}^{\infty} U(x_{k+l}, v_{i+\vartheta}(x_{k+l}))\\
&\le \lim_{\tau\to\infty}\left(\sum_{l=0}^{\tau+1}((1-p_{i+j_l}(x_{k+l})-q_{i+j_l}(x_{k+l}))\right.\\
&\quad\left.\times U(x_{k+l}, v_{i+\vartheta}(x_{k+l})))\right)\\
&< \infty,
\end{aligned} \tag{3.44}$$

which means $\sum_{l=0}^{\infty} U(x_{k+l}, v_{i+\vartheta}(x_{k+l})) < \infty$. On the other hand, as the utility function $U(x_k, u_k)$ is positive definite for x_k and u_k, we can derive that $\{U(x_{k+l}, v_{i+\vartheta}(x_{k+l}))\}$, $l = 0, 1, \ldots,$ is a positive series. Then, we have $U(x_{k+l}, v_{i+\vartheta}(x_{k+l})) \to 0$ as $l \to \infty$, which obtains $x_{k+l} \to 0$, as $l \to \infty$. According to the definition of admissible control law, we can draw the conclusion. □

Remark 3.1. From Theorem 3.2, we know that if the iterative value functions satisfy (3.31) and (3.32), then the admissibility of the iterative control laws can be guaranteed. Thus, the inequalities (3.31) and (3.32) can be called as "admissibility criteria" of the local iterative ADP algorithm. We emphasize that the developed local iterative ADP algorithm can be terminated if and only if the convergence and admissibility criteria are both satisfied.

Corollary 3.1. *For $i = 0, 1, \ldots$ and for all $x_k \in \Omega_x$, let $v_i(x_k)$ and $V_i(x_k)$ be obtained by (3.7)–(3.15). Let $\mathcal{L}_x^i = \Omega_x$, $\forall\, i = 0, 1, \ldots$ and*

$0 < \alpha_i(x_k) \leq 1$, $\forall x_k \in \Omega_x$. *If the iterative value function* $V_i(x_k)$, $i = 0, 1, \ldots,$ *satisfies*

$$V_{i+1}(x_k) - V_i(x_k) < \alpha_i(x_k) p_i(x_k) U(x_k, v_i(x_k)), \tag{3.45}$$

for a given function $-\infty < p_i(x_k) < 1$, $i = 0, 1, \ldots$ *and* $\forall x_k \in \Omega_x$, *then iterative control law* $v_i(x_k)$ *is an admissible control law.*

Remark 3.2. In each iteration, if $\mathscr{L}_x^i = \Omega_x$, $\forall\, i = 0, 1, \ldots$ and $0 < \alpha_i(x_k) \leq 1$, $\forall x_k \in \Omega_x$, then the local iterative ADP algorithm is reduced into a global one. In this situation, the admissibility criteria (3.31) and (3.32) are reduced into (3.45). Hence, we say that the global iterative ADP algorithm is a special case of the developed local iterative one and the admissibility criterion for the local iterative ADP algorithm is feasible for the global one [1, 3, 4, 6–8, 10, 15]. On the other hand, from Theorem 3.2, we can see that for the local iterative ADP algorithm (3.7)–(3.15), it requires at least ϑ iterative value functions to justify the admissibility of the iterative control law $v_{i+\vartheta}(x_k)$. To avoid this difficulty, a new admissibility criterion will be developed.

Lemma 3.3. *For* $i = 0, 1, \ldots$ *and for all* $x_k \in \Omega_x$, *let* $v_i(x_k)$ *and* $V_{i+1}(x_k)$ *be obtained by* (3.7)–(3.15). *Let the iteration index* i *satisfy* $i \geq i_{\min}$ *and the learning rate function* $\alpha_i(x_k)$ *satisfies* (3.10), (3.14), *and* (3.17). *Let* $\varepsilon > 0$ *be a given positive constant. For all* $x_k \in \Omega_x$, *define a new iterative value function* $\mathscr{V}_{i+j}(x_k)$ *as*

$$\begin{aligned} \mathscr{V}_{i+j+1}(x_k) = {} & (1 - \alpha_{i+j}(x_k)) V_i(x_k) + \alpha_{i+j}(x_k) \\ & \times \left(U(x_k, v_i(x_k)) + V_i(F(x_k, v_i(x_k))) \right). \end{aligned} \tag{3.46}$$

For $j = 0, 1, \ldots, \vartheta$, *and* $\forall\, x_k \in \Omega_x$, *if the iterative value function* $\mathscr{V}_{i+j+1}(x_k)$ *satisfies*

$$|\mathscr{V}_{i+j+1}(x_k) - V_i(x_k)| \leq \alpha_{i+j}(x_k)\, \varepsilon, \tag{3.47}$$

and the state set $\mathscr{L}_x^{i+j}$ *satisfies* (3.16), *then for all* $x_k \in \Omega_x$, *the iterative value function converges to the optimum. That is,* $V_i(x_k) \to J^*(x_k)$, *as* $\varepsilon \to 0$.

Theorem 3.3. *For* $i = 0, 1, \ldots,$ *let* $V_i(x_k)$ *and* $v_i(x_k)$ *be obtained by* (3.7)–(3.15). *Let the iteration index* i *satisfy* (3.18), *and let the*

learning rate function $\alpha_i(x_k)$ *satisfy* (3.10), (3.14), *and* (3.17). *For* $i = 0, 1, \ldots$ *and* $j = 0, 1, \ldots, \vartheta$, *let* $\{\hat{q}_{i+j}(x_k)\}$ *be a positive function sequence, which satisfies* $-\infty < \sup_{j=0,1,\ldots}\{\hat{q}_{i+j}(x_k)\} < 1$, $\forall x_k \in \Omega_x$. *Let* $\mathscr{L}_x^{i+j}$ *satisfy* (3.16). *If for all* $x_k \in \mathscr{L}_x^{i+j}$ *and for any* $j = 0, 1, \ldots$, *the iterative value function* $\mathscr{V}_{i+j}(x_k)$ *satisfy*

$$\mathscr{V}_{i+j+1}(x_k) - V_i(x_k) \leq \alpha_{i+j}(x_k)\hat{q}_{i+j}(x_k)U(x_k, v_i(x_k)), \tag{3.48}$$

then the iterative control law $v_i(x_k)$ *is an admissible control law.*

Proof. According to (3.46) and (3.48), for any $j = 0, 1, \ldots$ and for all $x_k \in \mathscr{L}_x^{i+j}$, we have

$$V_i(F(x_k, v_i(x_k))) - V_i(x_k) \leq (\hat{q}_{i+j}(x_k) - 1)U(x_k, v_i(x_k)). \tag{3.49}$$

As $\sup_{j=0,1,\ldots}\{\hat{q}_{i+j}(x_k)\} < 1$, there exists a constant $-\infty < \hat{q} < 1$ that satisfies $\sup_{j=0,1,\ldots}\{\hat{q}_{i+j}(x_k)\} < \hat{q}$, $\forall x_k \in \Omega_x$. On the other hand, as $\mathscr{L}_x^{i+j}$, $j = 0, 1, \ldots \vartheta$, satisfies (3.16), for all $x_k \in \Omega_x$, we can get

$$V_i(F(x_k, v_i(x_k))) - V_i(x_k) < (\hat{q} - 1)U(x_k, v_i(x_k)). \tag{3.50}$$

According to (3.50), for $\bar{\mathcal{N}}$, we can obtain

$$V_i(x_k) > (1 - \hat{q}) \sum_{l=0}^{\bar{\mathcal{N}}} U(x_{k+l}, v_i(x_{k+l})) + V_i(x_{k+\bar{\mathcal{N}}+1}). \tag{3.51}$$

As $V_i(x_k)$ is finite, $\forall\, x_k \in \Omega_x$ and $-\infty < \hat{q} < 1$, we can obtain $\sum_{\ell=0}^{\infty} U(x_{k+\ell}, v_i(x_{k+\ell}))$ is finite and $x_{k+\ell} \to 0$ as $\ell \to \infty$, which shows that the iterative control law $v_i(x_k)$ is an admissible control law. □

Theorem 3.4. *For* $i = 0, 1, \ldots$, *let* $V_i(x_k)$ *and* $v_i(x_k)$ *be obtained by* (3.7)–(3.15). *For any* $x_k \in \Omega_x$, *if the iterative control law* $v_i(x_k)$ *satisfies*

$$\lim_{j\to\infty} \frac{U\left(F(x_{k+j}, v_i(x_{k+j})), v_i(F(x_{k+j}, v_i(x_{k+j})))\right)}{U\left(x_{k+j}, v_i(x_{k+j})\right)} < 1, \tag{3.52}$$

then $v_i(x_k)$ *is an admissible control law.*

Proof. According to (3.52), there exists a positive integer $\Upsilon > 0$, such that for all $N \geq \Upsilon$,

$$\sup\left\{\frac{U\left(F(x_{k+N}, v_i(x_{k+N})), v_i(F(x_{k+N}, v_i(x_{k+N})))\right)}{U\left(x_{k+N}, v_i(x_{k+N})\right)}\right\} = \tilde{q} < 1 \tag{3.53}$$

holds. Using the mathematical induction, for any $j = 1, 2, \ldots$, we can get

$$U\left(x_{k+\Upsilon+j}, v_i(x_{k+\Upsilon+j})\right) < U(x_{k+\Upsilon}, v_i(x_{k+\Upsilon}))\,\tilde{q}^j. \tag{3.54}$$

As $0 < \tilde{q} < 1$, $\sum_{j=1}^{\infty} \tilde{q}^j$ is finite, which makes $\sum_{j=0}^{\infty} U(x_{k+\Upsilon+j}, v_i(x_{k+\Upsilon+j}))$ be finite. As Υ is finite, we have $\sum_{j=0}^{\Upsilon} U(x_{k+j}, v_i(x_{k+j}))$ is finite. Then we can conclude $\sum_{j=0}^{\infty} U(x_{k+j}, v_i(x_{k+j}))$ is finite. As $U(x_{k+j}, v_i(x_{k+j})) > 0$ for any $j = 0, 1, \ldots$, we have $\lim_{j\to\infty}(U(x_{k+j}, v_i(x_{k+j}))) = 0$, which means $x_{k+j} = 0$ as $j \to \infty$. Thus, the iterative control law $v_i(x_k)$ is an admissible control law. The proof is complete. □

Theorem 3.5. *For $i = 0, 1, \ldots$, let $V_i(x_k)$ and $v_i(x_k)$ be obtained by* (3.7)–(3.15). *If iterative control law $v_i(x_k)$ satisfies*

$$\lim_{x_k \to 0} \frac{U\left(F(x_k, v_i(x_k)), v_i(F(x_k, v_i(x_k)))\right)}{U\left(x_k, v_i(x_k)\right)} > 1, \tag{3.55}$$

then $v_i(x_k)$ is not an admissible control law.

Proof. The statement can be proven by contradiction. Assume that $v_i(x_k)$ is admissible which makes (3.55) hold. According to (3.55), there exists a compact set Ω_o around the equilibrium point, such that for all $x_k \in \Omega_o$, the following inequality

$$\sup_{x_k \in \Omega_o}\left\{\frac{U\left(F(x_k, v_i(x_k)), v_i(F(x_k, v_i(x_k)))\right)}{U\left(x_k, v_i(x_k)\right)}\right\} = \tilde{p} > 1 \tag{3.56}$$

holds. As $v_i(x_k)$ is admissible, we have the system (3.4) is stable under $v_i(x_k)$. Thus there exits a positive integer $\tilde{\Upsilon} > 0$, such that for all $\tilde{N} \geq \tilde{\Upsilon}$, $x_{k+\tilde{N}} \in \Omega_o$. According to (3.56), for any $l = 0, 1, \ldots$, we

can obtain that $U\left(x_{k+\tilde{\Upsilon}+l}, v_i(x_{k+\tilde{\Upsilon}+l})\right) > U(x_{k+\tilde{\Upsilon}}, v_i(x_{k+\tilde{\Upsilon}}))\,\tilde{p}^l$. For $j = 0, 1, \ldots, l$, we can get

$$\sum_{j=0}^{l} U(x_{k+\tilde{\Upsilon}+j}, v_i(x_{k+\tilde{\Upsilon}+j})) > U(x_{k+\tilde{\Upsilon}}, v_i(x_{k+\tilde{\Upsilon}})) \sum_{j=0}^{l} \tilde{p}^j. \tag{3.57}$$

As $\tilde{p} > 1$, we have $\sum_{j=0}^{l} \tilde{p}^j \to \infty$ as $l \to \infty$, which means $\sum_{j=0}^{\infty} U(x_{k+\tilde{\Upsilon}+j}, v_i(x_{k+\tilde{\Upsilon}+j})) \to \infty$. It is a contradiction against the definition of the admissible control law. Thus, the assumption is false and the iterative control law $v_i(x_k)$ is not an admissible control law. □

Theorem 3.6. *For $i = 0, 1, \ldots,$ let $V_i(x_k)$ and $v_i(x_k)$ be obtained by (3.7)–(3.15). For all $x_k \in \Omega_x$, if iterative control law $v_{i-1}(x_k)$ is admissible and satisfies*

$$\begin{aligned} &\frac{U(F(x_k, v_i(x_k)), v_i(F(x_k, v_i(x_k))))}{U(x_k, v_i(x_k))} \\ &\quad < \frac{U(F(x_k, v_{i-1}(x_k)), v_{i-1}(F(x_k, v_{i-1}(x_k))))}{U(x_k, v_{i-1}(x_k))}, \end{aligned} \tag{3.58}$$

then $v_i(x_k)$ is an admissible control law.

Proof. Let $\varrho_i(x_k)$ and $\varrho_{i-1}(x_k)$ satisfy

$$U(F(x_k, v_i(x_k)), v_i(F(x_k, v_i(x_k)))) = \varrho_i(x_k) U(x_k, v_i(x_k)), \tag{3.59}$$

and

$$\begin{aligned} &U(F(x_k, v_{i-1}(x_k)),\ v_{i-1}(F(x_k, v_{i-1}(x_k)))) \\ &\quad = \varrho_{i-1}(x_k) U(x_k, v_{i-1}(x_k)), \end{aligned} \tag{3.60}$$

respectively. According to (3.58), we have

$$\varrho_i(x_k) < \varrho_{i-1}(x_k) \tag{3.61}$$

holds, $\forall x_k \in \Omega_x$. As $v_{i-1}(x_k)$, $i = 1, 2, \ldots,$ is an admissible control law, there exists a finite value function $P_{i-1}(x_k)$ that satisfies

$$P_{i-1}(x_k) = \sum_{j=0}^{\infty} U(x_{k+j}, v_{i-1}(x_{k+j})). \tag{3.62}$$

According to (3.59), $P_{i-1}(x_k)$ can be expressed as

$$P_{i-1}(x_k) = \left(\sum_{j=0}^{\infty} \left(\prod_{l=0}^{j} \varrho_{i-1}(x_{k+l}) \right) \right) U(x_k, v_{i-1}(x_k)). \tag{3.63}$$

As the value function $P_{i-1}(x_k)$ is finite, we can get that $\sum_{j=0}^{\infty} (\Pi_{l=0}^{j} \varrho_{i-1}(x_{k+l}))$ is finite. If we let $P_i(x_k)$ be the value function created by $v_i(x_k)$, i.e.,

$$P_i(x_k) = \sum_{j=0}^{\infty} U(x_{k+j}, v_i(x_{k+j})), \tag{3.64}$$

then according to (3.61), we can obtain

$$\begin{aligned} P_i(x_k) &= \left(\sum_{j=0}^{\infty} \left(\prod_{l=0}^{j} \varrho_i(x_{k+l}) \right) \right) U(x_k, v_i(x_k)) \\ &\leq \left(\sum_{j=0}^{\infty} \left(\prod_{l=0}^{j} \varrho_{i-1}(x_{k+l}) \right) \right) U(x_k, v_i(x_k)) \\ &= P_{i-1}(x_k) \frac{U(x_k, v_i(x_k))}{U(x_k, v_{i-1}(x_k))}. \end{aligned} \tag{3.65}$$

For any $x_k \in \Omega_x$ and $x_k \neq 0$, we have $P_i(x_k)$ is finite. For $x_k = 0$, according to Lemma 3.1, we can derive that $P_i(x_k)$ and $P_{i-1}(x_k)$ are both positive semi-definite functions of x_k, which means $P_i(x_k) = P_{i-1}(x_k) = 0$ for $x_k = 0$. Hence, we have $P_i(x_k)$ is finite for all $x_k \in \Omega_x$. On the other hand, as $U(x_k, v_i(x_k))$ is positive definite, according to (3.64), we can get $x_k \to 0$ as $k \to \infty$, which means $v_i(x_k)$ is an admissible control law. The proof is complete. □

3.4 Simulation Example

To evaluate the performance of the local iterative ADP algorithm, we choose an example with quadratic utility function for numerical experiment. Consider the torsional pendulum system [9]. The

dynamics of the pendulum is given as follows

$$\begin{cases} \dfrac{\mathrm{d}\theta}{\mathrm{d}t} = \omega \\ J\dfrac{\mathrm{d}\omega}{\mathrm{d}t} = u - Mgl\sin\theta - f_d\dfrac{\mathrm{d}\theta}{\mathrm{d}t}, \end{cases} \tag{3.66}$$

where $M = 1/3\text{kg}$ and $l = 2/3\text{m}$ are the mass and length of the pendulum bar, respectively. Let $J = 4/3Ml^2$ and $f_d = 0.3$ be the rotary inertia and frictional factor, respectively. Let $g = 9.8\text{m/s}^2$ be the gravitational acceleration. Discretizing the system function with the sampling interval $\Delta t = 0.1\text{s}$ leads to

$$\begin{bmatrix} x_{1(k+1)} \\ x_{2(k+1)} \end{bmatrix} = \begin{bmatrix} 0.1x_{2k} + x_{1k} \\ -0.49\sin(x_{1k}) - 0.1f_d \times x_{2k} + x_{2k} \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u_k, \tag{3.67}$$

where $x_{1k} = \theta_k$ and $x_{2k} = \omega_k$. Let the initial state be $x_0 = [1, -1]^{\mathsf{T}}$. The utility function is chosen as $U(x_k, u_k) = x_k^{\mathsf{T}} Q x_k + u_k^{\mathsf{T}} R u_k$, where $Q = I_1$, $R = I_2$, and I_1, I_2 are identity matrices with suitable dimensions. Let the state space be expressed as $\Omega_x = \{(x_{1k}, x_{2k}) | -1 \leq x_{1k} \leq 1, -1 \leq x_{2k} \leq 1\}$. To illustrate the effectiveness of the algorithm, we choose two different initial value functions with the form $\Psi^{\kappa}(x_k) = x_k^{\mathsf{T}} P_{\kappa} x_k$, $\kappa = 0, 1$. Let $P_0 = 0$ and $P_1 = \begin{bmatrix} 6.96 & 3.54 \\ 3.54 & 15.9 \end{bmatrix}$. We choose two learning rate functions $\alpha_i^{\zeta}(x_k)$, $\zeta = 0, 1$, $i = 0, 1, \ldots$.

Let $\alpha_i^0(x_k) = 0.39\sin(i+1) + 0.6$ and $\alpha_i^1(x_k) = 0.2\sin(i+1) + 0.3$. Give four points $\mathcal{C}^i$, $i = 0, 1, 2, 3$ in state space, where the coordinates are expressed as $\mathcal{C}^0 = (1, 1)$, $\mathcal{C}^1 = (-1, 1)$, $\mathcal{C}^2 = (-1, -1)$, and $\mathcal{C}^3 = (1, -1)$, respectively. Based on the learning rate functions $\alpha_i^{\zeta}(x_k)$, $\zeta = 0, 1$, $i = 0, 1, \ldots$, the local iterative ADP algorithm can be implemented. The iterative value functions with $\alpha_i^{\zeta}(x_k)$, $\zeta = 0, 1$, $i = 0, 1, \ldots$, and $\Psi^j(x_k)$, $j = 0, 1$, are shown in Figs. 3.1 and 3.2, where "In" means initial iteration and "Lm" means limiting iteration. Let the computation precision $\varepsilon = 0.02$. After 80 iterations, the iterative value function $V_i(x_k)$ is convergent to the optimum. The trajectories of iterative states and iterative control law are displayed in Figs. 3.3 and 3.4, where we can see that the system is stable.

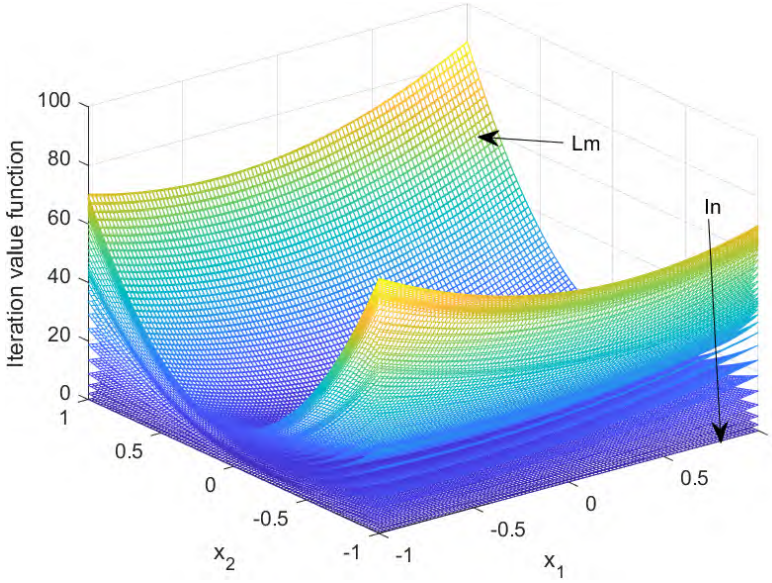

Fig. 3.1. Iterative value function $V_i(x_k)$ with $\alpha_i^0(x_k)$ and $\Psi^0(x_k)$.

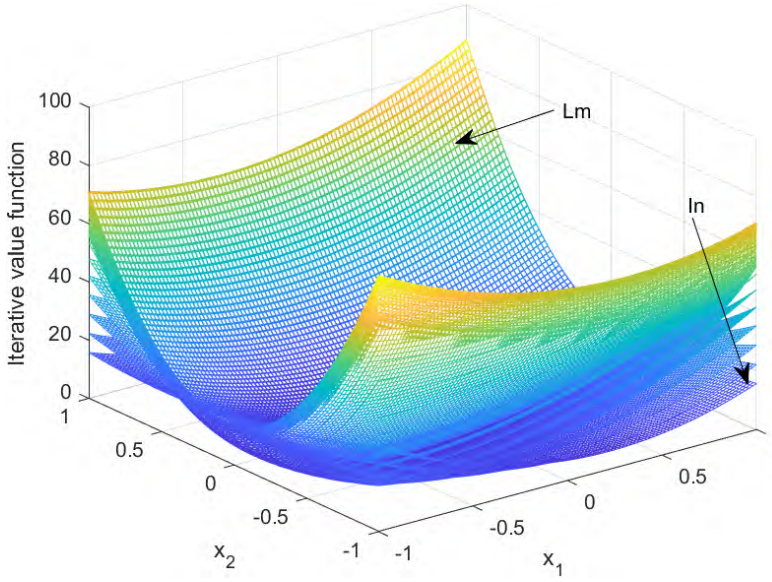

Fig. 3.2. Iterative value function $V_i(x_k)$ with $\alpha_i^1(x_k)$ and $\Psi^1(x_k)$.

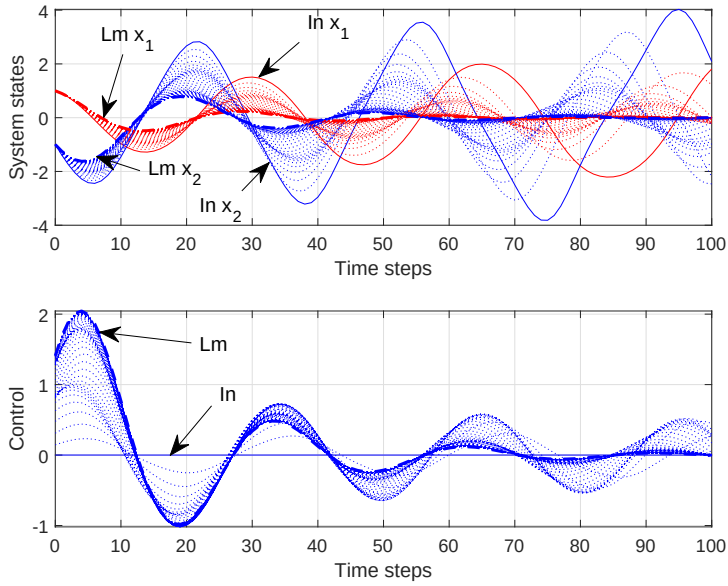

Fig. 3.3. Iterative states and control with $\alpha_i^0(x_k)$ and $\Psi^0(x_k)$.

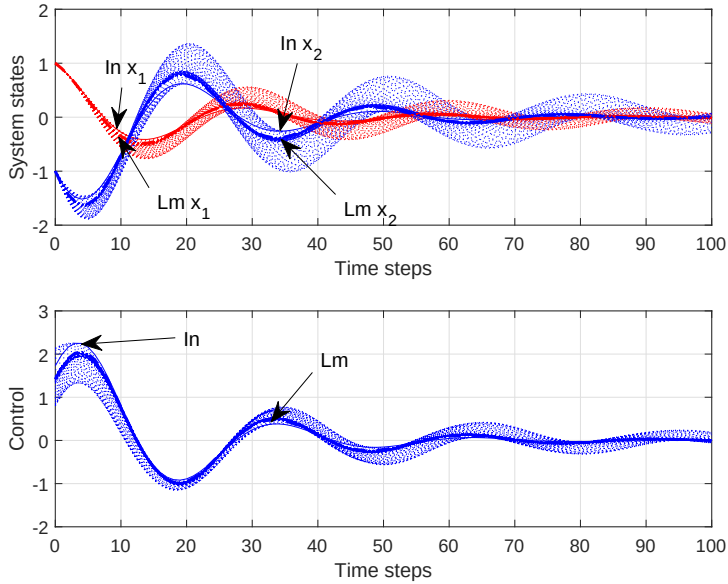

Fig. 3.4. Iterative states and control with $\alpha_i^1(x_k)$ and $\Psi^1(x_k)$.

3.5 Conclusion

In this chapter, an effective local iterative ADP algorithm is developed to solve infinite horizon optimal control problems for discrete-time nonlinear systems. Introducing a state-dependent learning rate function, the iterative value function and iterative control law in each iteration of the algorithms can both be updated using the partial state data in a given subset of the state space, instead of the whole state space. For many real control systems, such as process industry systems [11, 12], smart grids [13, 14], and so on, the system states are generally operated in a subset of the state space. Thus, the state data in a subset of the state space can be obtained instead of the whole state space. This makes the local iterative ADP algorithm more suitable for real applications than the traditional global ones. New termination criteria, including convergence and admissibility criteria, are presented to guarantee the effectiveness of the iterative control law.

References

[1] Al-Tamimi, A. Lewis, F.L. and Abu-Khalaf, M. (2008). Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics* 38(4), 943–949.

[2] Beard, R. (1995). Improving the closed-loop performance of nonlinear systems. PhD thesis, Rensselaer Polytechnic Institute, Troy, NY.

[3] Bertsekas, D.P. (2007). *Dynamic Programming and Optimal Control* (3rd edn.), Athena Scientific, Belmont, MA.

[4] Bertsekas, D.P. and Tsitsiklis, J.N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

[5] Khalil, H.K. (2002). *Nonlinear Systems*. Prentice-Hall, New York.

[6] Lewis, F.L., Vrabie, D. and Vamvoudakis, K.G. (2012). Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Control Systems Magazine* 32(6), 76–105.

[7] Lewis, F.L., Vrabie, D. and Vamvoudakis, K.G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems* 32(6), 76–105.

[8] Lincoln, B. and Rantzer, A. (2006). Relaxing dynamic programming. *IEEE Transactions on Automatic Control* 51(8), 1249–1260.

[9] Si, J. and Wang, Y.T. (2001). On-line learning control by association and reinforcement. *IEEE Transactions on Neural Networks* 12(2), 264–276.

[10] Wang, F., Zhang, H. and Liu, D. (2009). Adaptive dynamic programming: An introduction. *IEEE Computational Intelligence Magazine* 4(2), 39–47.

[11] Wei, Q. and Liu, D. (2014). Adaptive dynamic programming for optimal tracking control of unknown nonlinear systems with application to coal gasification. *IEEE Transactions on Automation Science and Engineering* 11(4), 1020–1036.

[12] Wei, Q. and Liu, D. (2014). Data-driven neuro-optimal temperature control of water gas shift reaction using stable iterative adaptive dynamic programming. *IEEE Transactions on Industrial Electronics* 61(11), 6399–6408.

[13] Wei, Q., Liu, D. and Shi, G. (2015). A novel dual iterative q-learning method for optimal battery management in smart residential environments. *IEEE Transactions on Industrial Electronics* 62(4), 2509–2518.

[14] Wei, Q., Liu, D., Shi, G. and Liu, Y. (2015). Optimal multi-battery coordination control for home energy management systems via distributed iterative adaptive dynamic programming. *IEEE Transactions on Industrial Electronics* 42(7), 4203–4214.

[15] Zhang, H., Wei, Q. and 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 Transactions on System, Man, and Cybernetics–Part B: Cybernetics* 38(4), 937–942.

Chapter 4

Discrete-Time Local Policy Iteration Adaptive Dynamic Programming

4.1 Introduction

In this chapter, a new policy iteration algorithm, called "local policy iteration algorithm", is developed to solve optimal control problems of discrete-time nonlinear systems. In the developed algorithm, for the first time, the iterative value function and iterative control law can both be updated in a given subset of the state space, instead of updated in the whole state space, which relaxes the computation burden of the traditional global policy iteration algorithms effectively. It is shown that the traditional discrete-time global policy iteration algorithms [1–5] are special cases of the developed local policy iteration algorithm. Convergence properties of the developed local policy iteration algorithm are analyzed, which shows that the iterative value function is monotonically non-increasing and converges to the optimum as the iteration index increases to infinity under some mild constraints. On the other hand, the admissibility of the iterative control law is proven. Although the iterative control law is updated by a subset of the state space, it is shown that the system can also be stable under any of the iterative control laws. Finally, simulation results are given to illustrate the performance of the developed method.

4.2 Problem Formulations

4.2.1 *System descriptions*

In this chapter, we study the following deterministic discrete-time nonlinear system:

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \ldots, \tag{4.1}$$

where $x_k \in \mathbb{R}^n$ is the state vector and $u_k \in \mathbb{R}^m$ is the control vector. Let x_0 be the initial state and $F(x_k, u_k)$ be the system function. Let $\underline{u}_k = (u_k, u_{k+1}, \ldots)$ be an arbitrary sequence of controls from k to ∞. The performance index function for state x_0 under the control sequence $\underline{u}_0 = (u_0, u_1, \ldots)$ is defined as

$$J(x_0, \underline{u}_0) = \sum_{k=0}^{\infty} U(x_k, u_k), \tag{4.2}$$

where the utility function $U(x_k, u_k)$ is a positive definite function for x_k and u_k.

The goal of this chapter is to find an optimal control scheme which stabilizes the system (4.1) and simultaneously minimizes the performance index function (4.2). For convenience of analysis, results of this chapter are based on the following assumption.

Assumption 4.1. The system (4.1) is controllable on a compact set $\Omega_x \subset \mathbb{R}^n$ containing the origin, and the function $F(x_k, u_k)$ is Lipschitz continuous on Ω_x; the system state $x_k = 0$ is an equilibrium state of system (4.1) under the control $u_k = 0$, i.e., $F(0, 0) = 0$; the feedback control $u_k = u(x_k)$ satisfies $u_k = u(x_k) = 0$ for $x_k = 0$.

Define the control sequence set as $\underline{\mathfrak{U}}_k = \{\underline{u}_k \colon \underline{u}_k = (u_k, u_{k+1}, \ldots), \forall u_{k+i} \in \mathbb{R}^m, i = 0, 1, \ldots\}$. Then, for an arbitrary control sequence $\underline{u}_k \in \underline{\mathfrak{U}}_k$, the optimal performance index function can be defined as

$$J^*(x_k) = \min_{\underline{u}_k} \{J(x_k, \underline{u}_k) \colon \underline{u}_k \in \underline{\mathfrak{U}}_k\}. \tag{4.3}$$

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

$$J^*(x_k) = \min_{u_k} \{U(x_k, u_k) + J^*(F(x_k, u_k))\}. \tag{4.4}$$

Define the law of optimal control as

$$u^*(x_k) = \arg\min_{u_k} \{U(x_k, u_k) + J^*(F(x_k, u_k))\}. \tag{4.5}$$

Hence, the HJB equation (4.4) can be written as

$$J^*(x_k) = U(x_k, u^*(x_k)) + J^*(F(x_k, u^*(x_k))). \tag{4.6}$$

Generally, the HJB equation (4.6) is computational untenable by traditional dynamic programming due to the curse of dimensionality. To overcome this difficulty, a new iterative ADP algorithm will be developed.

4.2.2 *Descriptions of the local iterative ADP algorithm*

In this section, a local iterative ADP algorithm is developed to obtain the optimal control law for the nonlinear system (4.1). In the local iterative ADP algorithm, the value function and control law are updated by iteration, with the iteration index i increasing from 0 to infinity. For $i = 0, 1, \ldots$, define $\{\mathscr{L}_x^i\}$ as a sequence of state sets, which satisfies $\mathscr{L}_x^i \subseteq \Omega_x$, $\forall\, i$.

For all $x_k \in \Omega_x$, let $v_0(x_k)$ be an admissible control law. For all $x_k \in \Omega_x$, let $V_0(x_k)$ be the initial iterative value function that satisfies the following generalized HJB (GHJB) equation:

$$V_0(x_k) = U(x_k, v_0(x_k)) + V_0(x_{k+1}), \tag{4.7}$$

where $x_{k+1} = F(x_k, v_0(x_k))$. Then, for all $x_k \in \mathscr{L}_x^0$, the local iterative control law $v_1(x_k)$ is computed as

$$v_1(x_k) = \arg\min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\} \tag{4.8}$$

and let $v_1(x_k) = v_0(x_k)$, for all $x_k \in \Omega_x \backslash \mathscr{L}_x^0$.

For all $x_k \in \Omega_x$, let $V_1(x_k)$ be the iterative value function that satisfies the following GHJB equation:

$$V_1(x_k) = U(x_k, v_1(x_k)) + V_1(F(x_k, v_1(x_k))). \tag{4.9}$$

For $i = 1, 2, \ldots$, let $V_i(x_k)$ be the iterative value function that satisfies the following GHJB equation:

$$V_i(x_k) = U(x_k, v_i(x_k)) + V_i(F(x_k, v_i(x_k))). \tag{4.10}$$

For all $x_k \in \mathscr{L}_x^i$, the iterative control law $v_{i+1}(x_k)$ can be computed as

$$\begin{aligned} v_{i+1}(x_k) &= \arg\min_{u_k} \{U(x_k, u_k) + V_i(x_{k+1})\} \\ &= \arg\min_{u_k} \{U(x_k, u_k) + V_i(F(x_k, u_k))\}, \end{aligned} \tag{4.11}$$

and for all $x_k \in \Omega_x \backslash \mathscr{L}_x^i$, let $v_{i+1}(x_k) = v_i(x_k)$.

Remark 4.1. In Ref. [2], a discrete-time global policy iteration algorithm was developed to obtain the optimal control law for the nonlinear system (4.1). However, in each iteration of the traditional global policy iteration algorithm in Ref. [2], it was required that the iterative value functions and iterative control laws were both updated for the whole state space, which limited its real-world applications. From the local policy iteration algorithm (4.7)–(4.11), we can see that for any $i = 0, 1, \ldots$, the iterative control law is updated in a given subset of the state space, i.e., $\mathscr{L}_x^i \subseteq \Omega_x$, instead of the whole state space. Thus, in each iteration, the developed local policy iteration algorithm can be implemented when the local data of the state space have been obtained, which avoids waiting for all the data of the whole state space for the traditional global policy iteration algorithms [1, 2, 4]. Thus, the computation efficiency of the developed local policy iteration algorithm is high and the computation burden of local policy iteration algorithm is relaxed. On the other hand, letting $\mathscr{L}_x^i \equiv \Omega_x$, $\forall\, i = 0, 1, \ldots$, the local policy iteration algorithm is reduced to the traditional global one. Hence, the global policy iteration algorithm is a special case of the local policy iteration algorithm.

4.3 Properties of the Local Iterative ADP Algorithm

In this section, the properties of the local policy iteration algorithm are discussed. First, the monotonicity property of the iterative value

function is presented and the admissibility of the iterative control law is discussed. Then, the convergence properties are developed. Under different constraints, it is shown that the iterative value function will converge to local and global optimum, with the corresponding local and global optimality proofs.

Before the main theorems, the following lemma is necessary.

Lemma 4.1. *Let $\mu(x_k)$ be an admissible control law. If Assumption* 4.1 *holds, then there exists an positive definite function $\mathcal{P}(x_k)$ that satisfies the following GHJB equation:*

$$\mathcal{P}(x_k) = U(x_k, \mu(x_k)) + \mathcal{P}(x_{k+1}). \tag{4.12}$$

Proof. According to the definition of the admissible control law, we can obtain that

$$\mathcal{P}(x_k) = \sum_{j=0}^{\infty} U(x_{k+j}, \mu(x_{k+j})) < \infty. \tag{4.13}$$

Letting $\mathcal{P}(x_{k+1}) = \sum_{j=1}^{\infty} U(x_{k+j}, \mu(x_{k+j}))$, we can obtain (4.12). On the other hand, according to Assumption 4.1, we have $\mathcal{P}(x_k) = 0$ for $x_k = 0$ and $\mathcal{P}(x_k) > 0$ for $x_k \neq 0$, which means $\mathcal{P}(x_k)$ is a positive definite function. The proof is complete. □

According to Lemma 4.1, for the admissible control law $v_0(x_k)$, we know that there exits a positive definite function $V_0(x_k)$ that satisfies (4.7). Thus, the iterative value function $V_0(x_k)$ can be defined. Initialized by an admissible control law $v_0(x_k)$, the monotonicity of the iterative value function and the admissibility of the iterative control law can be developed in the following theorem.

Theorem 4.1. *For $i = 0, 1, \ldots$, let $V_i(x_k)$ and $v_i(x_k)$ be obtained by* (4.7)–(4.11)*. If Assumption* 4.1 *holds, then we have*

(i) *for any $i = 0, 1, \ldots$, the iterative function $V_i(x_k)$ can be defined for all $x_k \in \Omega_x$;*

(ii) *for all $x_k \in \Omega_x$, the iterative value function $V_i(x_k)$ is monotonically non-increasing for any $i = 0, 1, \ldots$, i.e.,*

$$V_{i+1}(x_k) \leq V_i(x_k); \tag{4.14}$$

(iii) *for any $i = 0, 1, \ldots$, the iterative control law $v_i(x_k)$ is an admissible control law.*

Proof. The statement is proven by mathematical induction. First, we consider $i = 0$. We know that the iterative control law $v_0(x_k)$ is admissible. Let $\mathcal{V}_1^j(x_k)$, $j = 0, 1, \ldots$, be an iterative value function that satisfies

$$\mathcal{V}_1^{j+1}(x_k) = U(x_k, v_1(x_k)) + \mathcal{V}_1^j(x_{k+1}), \tag{4.15}$$

where $x_{k+1} = F(x_k, v_1(x_k))$ and we let $\mathcal{V}_1^0(x_k) = V_0(x_k)$. The iterative control law $v_1(x_k)$ is obtained by (4.8). From Lemma 4.1, we know that the iterative value function $V_0(x_k)$ can be defined for all $x_k \in \Omega_x$. Next, we prove that the inequality

$$\mathcal{V}_1^{j+1}(x_k) \leq V_0(x_k) \tag{4.16}$$

holds for any $j = 0, 1, \ldots$.

The inequality (4.16) is also proven by mathematical induction. First, for $j = 0$ and for all $x_k \in \mathscr{L}_x^0$, according to (4.8), we can get

$$\begin{aligned}
\mathcal{V}_1^1(x_k) &= U(x_k, v_1(x_k)) + \mathcal{V}_1^0(F(x_k, v_1(x_k))) \\
&= \min_{u_k}\{U(x_k, u_k) + V_0(x_{k+1})\} \\
&\leq U(x_k, v_0(x_k)) + V_0(F(x_k, v_0(x_k))) \\
&= V_0(x_k).
\end{aligned} \tag{4.17}$$

For $j = 0$ and for all $x_k \in \Omega_x \backslash \mathscr{L}_x^0$, we have

$$\begin{aligned}
\mathcal{V}_1^1(x_k) &= U(x_k, v_1(x_k)) + \mathcal{V}_1^0(F(x_k, v_1(x_k))) \\
&= U(x_k, v_0(x_k)) + V_0(F(x_k, v_0(x_k))) \\
&= V_0(x_k).
\end{aligned} \tag{4.18}$$

According to (4.17) and (4.18), for all $x_k \in \Omega_x$, we have $\mathcal{V}_1^1(x_k) \leq V_0(x_k)$. As $V_0(x_k)$ and $U(x_k, v_0(x_k))$ are both positive definite functions of x_k, we have $\mathcal{V}_1^1(x_k)$ which is a positive definite function of x_k.

Assume that (4.16) holds for $j = l - 1$, $l = 1, 2, \ldots$, i.e., $\mathcal{V}_1^l(x_k) \leq V_0(x_k)$, $\forall x_k \in \Omega_x$. Then, for $j = l$ and $x_k \in \mathscr{L}_x^0$, we have

$$\begin{aligned}
\mathcal{V}_1^{l+1}(x_k) &= U(x_k, v_1(x_k)) + \mathcal{V}_1^l(F(x_k, v_1(x_k))) \\
&\leq U(x_k, v_1(x_k)) + V_0(F(x_k, v_1(x_k)))
\end{aligned}$$

$$\begin{aligned}
&= \min_{u_k}\{U(x_k, u_k) + V_0(x_{k+1})\} \\
&\leq U(x_k, v_0(x_k)) + V_0(F(x_k, v_0(x_k))) \\
&= V_0(x_k). \qquad (4.19)
\end{aligned}$$

For $x_k \in \Omega_x \backslash \mathscr{L}_x^0$, we have

$$\begin{aligned}
\mathcal{V}_1^{l+1}(x_k) &= U(x_k, v_1(x_k)) + \mathcal{V}_1^l(F(x_k, v_1(x_k))) \\
&= U(x_k, v_0(x_k)) + \mathcal{V}_1^l(F(x_k, v_0(x_k))) \\
&\leq U(x_k, v_0(x_k)) + V_0(F(x_k, v_0(x_k))) \\
&= V_0(x_k). \qquad (4.20)
\end{aligned}$$

Thus, (4.16) holds for any $j = 0, 1, \ldots$. The mathematical induction for (4.16) is complete.

According to (4.15) and (4.16), for any $j = 0, 1, \ldots$, we can obtain

$$\begin{aligned}
\mathcal{V}_1^{j+1}(x_k) &= \sum_{l=0}^{j} U(x_{k+l}, v_1(x_{k+l})) + \mathcal{V}_1^0(x_{k+j+1}) \\
&\leq V_0(x_k). \qquad (4.21)
\end{aligned}$$

According to Assumption 4.1, we can derive that for any $j = 0, 1, \ldots, \mathcal{V}_1^{j+1}(x_k)$ in (4.21) is a positive definite function of x_k.

Let $j \to \infty$. According to (4.21), for all $x_k \in \Omega_x$, we have

$$\begin{aligned}
\lim_{j\to\infty} \mathcal{V}_1^{j+1}(x_k) &= \lim_{j\to\infty} \left(\sum_{l=0}^{j} U(x_{k+l}, v_1(x_{k+l})) \right) \\
&\quad + \lim_{j\to\infty} V_0(x_{k+j+1}) \\
&\leq V_0(x_k) \\
&< \infty. \qquad (4.22)
\end{aligned}$$

From (4.22), we know that the positive series $\sum_{l=0}^{\infty} U(x_{k+l}, v_1(x_{k+l})) < \infty$. Thus, for all $x_k \in \Omega_x$, we can derive that $x_{k+l} \to 0$ as $l \to \infty$ and $\lim_{j\to\infty} V_0(x_{k+j+1}) = 0$, which means that

the iterative control law $v_1(x_k)$ is an admissible control law. Letting

$$\mathcal{V}_1^{\infty}(x_k) = \lim_{j\to\infty} \mathcal{V}_1^{j+1}(x_k), \tag{4.23}$$

we can obtain that

$$\mathcal{V}_1^{\infty}(x_k) = U(x_k, v_1(x_k)) + \mathcal{V}_1^{\infty}(x_{k+1}), \tag{4.24}$$

where $x_{k+1} = F(x_k, v_1(x_k))$. According to the definition of $V_1(x_k)$ in (4.9), we can obtain that

$$\lim_{j\to\infty} \mathcal{V}_1^{j+1}(x_k) = \mathcal{V}_1^{\infty} = V_1(x_k). \tag{4.25}$$

According to (4.21) and (4.25), we can obtain that $V_1(x_k) \le V_0(x_k)$, $\forall x_k \in \Omega_x$. Thus, the iterative value function $V_1(x_k)$ in (4.9) can be defined for all $x_k \in \Omega_x$.

Assume the conclusion holds for $i = l-1$, $l = 1, 2, \ldots$, i.e., $V_l(x_k) \le V_{l-1}(x_k)$ and $v_l(x_k)$ is an admissible control law. For $i = l$, let $\mathcal{V}_{l+1}^{j}(x_k)$, $j = 0, 1, \ldots$, be an iterative value function that satisfies

$$\mathcal{V}_{l+1}^{j+1}(x_k) = U(x_k, v_{l+1}(x_k)) + \mathcal{V}_{l+1}^{j}(x_{k+1}), \tag{4.26}$$

where $x_{k+1} = F(x_k, v_{l+1}(x_k))$ and let $\mathcal{V}_{l+1}^{0}(x_k) = V_l(x_k)$. According to the idea of the mathematical induction from (4.17)–(4.20), for $j = 0, 1, \ldots$ and all $x_k \in \Omega_x$, we can obtain that

$$\mathcal{V}_{l+1}^{j+1}(x_k) \le V_l(x_k). \tag{4.27}$$

Letting $\mathcal{V}_{l+1}^{\infty}(x_k) = \lim_{j\to\infty} \mathcal{V}_{l+1}^{j+1}(x_k)$, we can obtain $\mathcal{V}_{l+1}^{\infty}(x_k) \le V_l(x_k)$, $\forall x_k \in \Omega_x$. According to (4.26) and (4.27), for all $x_k \in \Omega_x$, we have

$$\begin{aligned}
\mathcal{V}_{l+1}^{\infty}(x_k) &= \lim_{j\to\infty} \left(\sum_{l=0}^{j} U(x_{k+l}, v_{l+1}(x_{k+l})) \right) \\
&\quad + \lim_{j\to\infty} V_l(x_{k+j+1}) \\
&\le V_l(x_k) \\
&< \infty.
\end{aligned} \tag{4.28}$$

According to (4.22)–(4.24), we can obtain that

$$\sum_{l=0}^{\infty} U(x_{k+l}, v_{l+1}(x_{k+l})) < \infty \tag{4.29}$$

and

$$\mathcal{V}_{l+1}^{\infty}(x_k) = U(x_k, v_{l+1}(x_k)) + \mathcal{V}_{l+1}^{\infty}(x_{k+1}). \tag{4.30}$$

Thus, we can obtain that the iterative control law $v_{l+1}(x_k)$ is admissible control law, and according to (4.10), we have $V_{l+1}(x_k) = \mathcal{V}_{l+1}^{\infty}(x_k)$. Thus, the iterative value function $V_{l+1}(x_k)$ in (4.10) can be defined for all $x_k \in \Omega_x$ and $V_{l+1}(x_k) = \mathcal{V}_{l+1}^{\infty}(x_k) \leq V_l(x_k)$. The mathematical induction is complete. □

In each iteration of the local policy iteration algorithm (4.7)–(4.11), the iterative control law $v_i(x_k)$, $i = 0, 1, \ldots$, is updated in a given subset of the state space, $\mathscr{L}_x^i \subseteq \Omega_x$, while the iterative control law in $\mathscr{L}_x^i \backslash \Omega_x$ is kept unchanged. In this situation, the iterative value function may also be updated in a subset of the state space. This property is shown in the following theorem.

Theorem 4.2 (Local iteration property). *For $i = 0, 1, \ldots$, let $V_i(x_k)$ and $v_{i+1}(x_k)$ be obtained by (4.7)–(4.11). For $i = 0, 1, \ldots$, let $\mathscr{G}_x^i \subseteq \Omega_x$ be the state set, such that for all $x_k \in \mathscr{G}_x^i$, the system state satisfies*

$$F(x_k, v_{i+1}(x_k)) \in \mathscr{L}_x^i. \tag{4.31}$$

Then, we have

$$\begin{cases} V_{i+1}(x_k) \leq V_i(x_k), \forall x_k \in (\mathscr{L}_x^i \cup \mathscr{G}_x^i), \\ V_{i+1}(x_k) = V_i(x_k), \forall x_k \in \Omega_x \backslash (\mathscr{L}_x^i \cup \mathscr{G}_x^i). \end{cases} \tag{4.32}$$

Proof. For $i = 0, 1, \ldots$, define the iterative value function $\mathcal{V}_{i+1}^{j+1}(x_k)$ as (4.26), $j = 0, 1, \ldots$, where $\mathcal{V}_{i+1}^{0}(x_k) = V_i(x_k)$ and the iterative value functions $V_i(x_k)$ satisfy the GHJB equations (4.7) and (4.10).

First, for $j = 0$, we have

$$\mathcal{V}_{i+1}^{1}(x_k) = U(x_k, v_{i+1}(x_k)) + \mathcal{V}_{i+1}^{0}(x_{k+1}), \tag{4.33}$$

where $v_{i+1}(x_k)$ is the iterative control law. If $x_k \in \mathscr{L}_x^i$, we have

$$\begin{aligned}\mathcal{V}_{i+1}^1(x_k) &= U(x_k, v_{i+1}(x_k)) + \mathcal{V}_{i+1}^0(x_{k+1}) \\ &= U(x_k, v_{i+1}(x_k)) + V_i(x_{k+1}) \\ &= \min_{u_k}\{U(x_k, u_k) + V_i(x_{k+1})\} \\ &\le U(x_k, v_i(x_k)) + V_i(x_{k+1}) \\ &= V_i(x_k). \end{aligned} \tag{4.34}$$

If $x_k \in \Omega_x \backslash \mathscr{L}_x^i$, according to the definition of $v_{i+1}(x_k)$ in (4.11), we can get

$$\begin{aligned}\mathcal{V}_{i+1}^1(x_k) &= U(x_k, v_{i+1}(x_k)) + \mathcal{V}_{i+1}^0(x_{k+1}) \\ &= U(x_k, v_i(x_k)) + V_i(x_{k+1}) \\ &= V_i(x_k). \end{aligned} \tag{4.35}$$

Thus, (4.32) holds for $j = 0$. According to (4.34) and (4.35), we can obtain $\mathcal{V}_{i+1}^1(x_k) \le V_i(x_k), \forall\, x_k \in \Omega_x$. Consider $j = 1$. For all $x_k \in \Omega_x$, there are four cases which can be displayed as

$$\begin{aligned}&\text{Case 1}: \ x_k \in \mathscr{L}_x^i \cap \mathscr{G}_x^i; \\ &\text{Case 2}: \ x_k \in \mathscr{L}_x^i \cap (\Omega_x \backslash \mathscr{G}_x^i); \\ &\text{Case 3}: \ x_k \in (\Omega_x \backslash \mathscr{L}_x^i) \cap \mathscr{G}_x^i; \\ &\text{Case 4}: \ x_k \in (\Omega_x \backslash \mathscr{L}_x^i) \cap (\,\Omega_x \backslash \mathscr{G}_x^i).\end{aligned}$$

Considering Case 1, according to (4.34), we have

$$\begin{aligned}\mathcal{V}_{i+1}^2(x_k) &= U(x_k, v_{i+1}(x_k)) + \mathcal{V}_{i+1}^1(x_{k+1}) \\ &\le U(x_k, v_{i+1}(x_k)) + V_i(x_{k+1}) \\ &\le V_i(x_k). \end{aligned} \tag{4.36}$$

Considering Case 2, we have $F(x_k, v_{i+1}(x_k)) \in \mathscr{L}_x^i$. According to (4.35), we can get

$$\begin{aligned}\mathcal{V}_{i+1}^2(x_k) &= U(x_k, v_{i+1}(x_k)) + \mathcal{V}_{i+1}^1(x_{k+1}) \\ &= U(x_k, v_{i+1}(x_k)) + V_i(x_{k+1}) \\ &\le V_i(x_k). \end{aligned} \tag{4.37}$$

Considering Case 3, according to (4.34), we have

$$\begin{aligned}\mathcal{V}_{i+1}^{2}(x_k) &= U(x_k, v_{i+1}(x_k)) + \mathcal{V}_{i+1}^{1}(x_{k+1}) \\ &= U(x_k, v_i(x_k)) + \mathcal{V}_{i+1}^{1}(x_{k+1}) \\ &\leq U(x_k, v_i(x_k)) + V_i(x_{k+1}) \\ &\leq V_i(x_k). \end{aligned} \tag{4.38}$$

Considering Case 4, according to (4.35), we can get

$$\begin{aligned}\mathcal{V}_{i+1}^{2}(x_k) &= U(x_k, v_{i+1}(x_k)) + \mathcal{V}_{i+1}^{1}(x_{k+1}) \\ &= U(x_k, v_i(x_k)) + V_i^{1}(x_{k+1}) \\ &= V_i(x_k). \end{aligned} \tag{4.39}$$

According to (4.36)–(4.39), we can get that (4.32) holds for $j = 1$.

According to mathematical induction, for any $j = 0, 1, \ldots,$ we can obtain

(i) $\mathcal{V}_{i+1}^{j+1}(x_k) \leq V_i(x_k), \quad \forall x_k \in \mathscr{L}_x^i \cap \mathscr{G}_x^i;$
(ii) $\mathcal{V}_{i+1}^{j+1}(x_k) \leq V_i(x_k), \quad \forall x_k \in \mathscr{L}_x^i \cap (\Omega_x \backslash \mathscr{G}_x^i);$
(iii) $\mathcal{V}_{i+1}^{j+1}(x_k) \leq V_i(x_k), \quad \forall x_k \in (\Omega_x \backslash \mathscr{L}_x^i) \cap \mathscr{G}_x^i;$
(iv) $\mathcal{V}_{i+1}^{j+1}(x_k) = V_i(x_k), \quad \forall x_k \in (\Omega_x \backslash \mathscr{L}_x^i) \cap (\Omega_x \backslash \mathscr{G}_x^i).$

Thus, we can obtain

$$\begin{cases} \mathcal{V}_{i+1}^{j+1}(x_k) \leq V_i(x_k), & \forall x_k \in (\mathscr{L}_x^i \cup \mathscr{G}_x^i), \\ \mathcal{V}_{i+1}^{j+1}(x_k) = V_i(x_k), & \forall x_k \in \Omega_x \backslash (\mathscr{L}_x^i \cup \mathscr{G}_x^i). \end{cases} \tag{4.40}$$

According to Theorem 4.1, we have $V_{i+1}(x_k) = \mathcal{V}_{i+1}^{\infty}(x_k), \forall\, i = 0, \ldots.$ Letting $j \to \infty$, we can obtain (4.32). The proof is complete. □

From Theorem 4.2, the iterative value function $V_i(x_k)$ is local updated for any $i = 0, 1, \ldots.$ From Theorem 4.1, we know that for any $i = 0, 1, \ldots,$ the iterative value function $V_i(x_k)$ is positive definite for x_k, which means $V_i(x_k)$ is lower bounded for all $x_k \in \Omega_x$. As for all $x_k \in \Omega_x$, the iterative value function $V_i(x_k)$ is monotonically non-increasing as i increases. Hence, the convergence property of the iterative value function can be analyzed.

As for $i = 0, 1, \ldots$, the iterative control law is updated in $\mathscr{L}_x^i$, for an state $x_k \in \Omega_x$, it can be located in several state sets, i.e., $x_k \in \{\mathscr{L}_x^{i_0} \cap \mathscr{L}_x^{i_1} \cap \cdots\}$, $i_\eta \in \{0, 1, \ldots\}, \eta = 0, 1, \ldots$. Let

$$\mathcal{T}_x = \{i_\eta | x_k \in \mathscr{L}_x^{i_\eta}, i_\eta \in \{0, 1, \ldots\}, \eta = 0, 1, \ldots\}. \tag{4.41}$$

Let π_x denote the number of the elements in $\mathcal{T}_x$, which is expressed as

$$\pi_x = |\mathcal{T}_x|. \tag{4.42}$$

Then, we can derive the following theorem.

Theorem 4.3 (Local optimality property). *For $i = 0, 1, \ldots$, let $V_i(x_k)$ and $v_{i+1}(x_k)$ be obtained by (4.7)–(4.11). If π_x in (4.42) satisfies $\pi_x \to \infty$, that is,*

$$x_k \in \bigcap_{\eta=0}^{\infty} \mathscr{L}_x^{i_\eta}, \quad i_\eta \in \{0, 1, \ldots\}, \tag{4.43}$$

then the iterative value function will converge to a solution of the HJB equation (4.4) for all x_k that satisfies (4.43).

Proof. According to (4.41), without loss of generality, we assume that $i_0 \leq i_1 \leq \cdots \leq i_\eta \leq \cdots$. From (4.42), for $\pi_x \to \infty$, we know that $\eta \to \infty$. According to Theorem 4.1, as the iterative value function $V_i(x_k)$ is a non-increasing and lower-bounded sequence, the limit of the iterative value function $V_{i_\eta}(x_k)$ exists as $\eta \to \infty$. For all $x_k \in \Omega_x$, define the value function $V_\infty^{\mathcal{L}}(x_k)$ as the limit of the iterative value function $V_i(x_k)$, i.e.,

$$V_\infty^{\mathcal{L}}(x_k) = \lim_{\eta \to \infty} V_{i_\eta}(x_k). \tag{4.44}$$

For any $\eta = 0, 1, \ldots$ and $x_k \in \Omega_x$, according to (4.26), the iterative value function $\mathcal{V}_{i_\eta+1}^{j+1}(x_k)$ is

$$\mathcal{V}_{i_\eta+1}^{j+1}(x_k) = U(x_k, v_{i_\eta+1}(x_k)) + \mathcal{V}_{i_\eta+1}^{j}(x_{k+1}), \tag{4.45}$$

where $x_{k+1} = F(x_k, v_{i_\eta+1}(x_k))$, $\mathcal{V}_{i_\eta+1}^{0}(x_k) = V_{i_\eta}(x_k)$, and $v_{i_\eta+1}$ is defined in (4.11).

For $j = 0$, according to (4.34) and (4.35), we can obtain that

$$\begin{cases} \mathcal{V}^1_{i_\eta+1}(x_k) \leq V_{i_\eta}(x_k), & \forall x_k \in \mathscr{L}_x^{i_\eta}, \\ \mathcal{V}^1_{i_\eta+1}(x_k) = V_{i_\eta}(x_k), & \forall x_k \in \Omega_x \backslash \mathscr{L}_x^{i_\eta}, \end{cases} \tag{4.46}$$

which means

$$\mathcal{V}^1_{i_\eta+1}(x_k) \leq V_{i_\eta}(x_k), \tag{4.47}$$

$\forall x_k \in \Omega_x$. For $j = 1$, for all $x_k \in \Omega_x$, we can get

$$\begin{aligned} \mathcal{V}^2_{i_\eta+1}(x_k) &= U(x_k, v_{i_\eta+1}(x_k)) + \mathcal{V}^1_{i_\eta+1}(x_{k+1}) \\ &\leq U(x_k, v_{i_\eta+1}(x_k)) + \mathcal{V}^0_{i_\eta+1}(x_{k+1}) \\ &= \mathcal{V}^1_{i_\eta+1}(x_k). \end{aligned} \tag{4.48}$$

Using mathematical induction, for all $x_k \in \Omega_x$ and $j = 0, 1, \ldots$, we can obtain

$$\mathcal{V}^{j+1}_{i_\eta+1}(x_k) \leq \mathcal{V}^j_{i_\eta+1}(x_k). \tag{4.49}$$

Letting $j \to \infty$, according to Theorem 4.1, for any $\eta = 0, 1, \ldots$ and $\forall x_k \in \Omega_x$, we can obtain

$$\mathcal{V}^1_{i_\eta+1}(x_k) \geq \lim_{j\to\infty} \mathcal{V}^j_{i_\eta+1}(x_k) = V_{i_\eta+1}(x_k). \tag{4.50}$$

According to (4.45) and (4.47), for all $x_k \in \mathscr{L}_x^{i_\eta}$, we have

$$\begin{aligned} \mathcal{V}^1_{i_\eta+1}(x_k) &= U(x_k, v_{i_\eta+1}(x_k)) + \mathcal{V}^0_{i_\eta+1}(x_{k+1}) \\ &= \min_{u_k}\{U(x_k, u_k) + \mathcal{V}^0_{i_\eta+1}(x_{k+1})\} \\ &= \min_{u_k}\{U(x_k, u_k) + V_{i_\eta}(x_{k+1})\} \\ &\leq V_{i_\eta}(x_k). \end{aligned} \tag{4.51}$$

Letting $\eta \to \infty$, we can get that

$$\min_{u_k}\{U(x_k,u_k)+V_\infty^{\mathcal{L}}(x_{k+1})\} \le V_\infty^{\mathcal{L}}(x_k) \tag{4.52}$$

holds for all $x_k \in \cap_{\eta=0}^{\infty}\mathscr{L}_x^{i_\eta}$. On the other hand, according to (4.50), for $x_k \in \mathscr{L}_x^{i_\eta}$, we can obtain

$$\begin{aligned}\mathcal{V}_{i_\eta+1}^{1}(x_k) &= \min_{u_k}\{U(x_k,u_k)+V_{i_\eta}(x_{k+1})\}\\ &\ge V_{i_\eta+1}(x_k).\end{aligned} \tag{4.53}$$

Letting $\eta \to \infty$, for all $x_k \in \cap_{\eta=0}^{\infty}\mathscr{L}_x^{i_\eta}$, we can get

$$\min_{u_k}\{U(x_k,u_k)+V_\infty^{\mathcal{L}}(x_{k+1})\} \ge V_\infty^{\mathcal{L}}(x_k). \tag{4.54}$$

Combining (4.52) and (4.54), we can obtain that for all $x_k \in \cap_{\eta=0}^{\infty}\mathscr{L}_x^{i_\eta}$, the limiting iterative value function $V_\infty^{\mathcal{L}}(x_k)$ is a solution of the following HJB equation:

$$V_\infty^{\mathcal{L}}(x_k) = \min_{u_k}\{U(x_k,u_k)+V_\infty^{\mathcal{L}}(x_{k+1})\}. \tag{4.55}$$

The proof is complete. □

Theorem 4.4 (Non-global optimal property). *For $i=0,1,\ldots$ and all $x_k \in \Omega_x$, let $V_i(x_k)$ and $v_{i+1}(x_k)$ be obtained by* (4.7)–(4.11). *Let $\pi_x \to \infty$, which makes the iterative value function converge to $\mathcal{V}_\infty^{\mathcal{L}}(x_k)$ for all x_k that satisfies* (4.43), *where $\mathcal{V}_\infty^{\mathcal{L}}(x_k)$ satisfies* (4.55). *Define*

$$\mathcal{T}_x' = \{i_\eta' | x_k \in \mathscr{L}_x^{i_\eta'}, i_\eta' \in \{0,1,\ldots\}, \eta = 0,1,\ldots\}, \tag{4.56}$$

and

$$\pi_x' = |\mathcal{T}_x'|. \tag{4.57}$$

If there exists a state $x_k \in \Omega_x$ which makes $\pi_x' < \infty$, then the converged iterative value function $V_\infty^{\mathcal{L}}(x_k)$ is not guaranteed to converge to the optimal performance index function for all $x_k \in \Omega_x$.

Proof. As $\pi_x \to \infty$, for all $x_k \in \Omega_x$, defining

$$v_\infty^{\mathcal{L}}(x_k) = \lim_{\eta \to \infty} v_{i_\eta+1}(x_k), \tag{4.58}$$

we can get

$$V_\infty^{\mathcal{L}}(x_k) = U(x_k, v_\infty^{\mathcal{L}}(x_k)) + V_\infty^{\mathcal{L}}(x_{k+1}), \tag{4.59}$$

where $x_{k+1} = F(x_k, v_\infty^{\mathcal{L}}(x_k))$. According to (4.57), if there exists a state $x_k \in \Omega_x$ which satisfies $\pi_x' < \infty$, then there exists a finite non-negative integer $\vartheta \geq 0$, such that

$$x_k \in \bigcap_{\eta=0}^{\vartheta} \mathscr{L}_x^{i_\eta'}. \tag{4.60}$$

Without loss of generality, we assume that $i_0' \leq i_1' \leq \cdots \leq i_\vartheta'$.

Define $\mathfrak{L}_x^{i_\vartheta'} = \cap_{\eta=0}^{\vartheta} \mathscr{L}_x^{i_\eta'}$. Give a state set $\mathscr{L}_x^{i_{\vartheta+1}'}$, such that $\mathscr{L}_x^{i_{\vartheta+1}'} \cap \mathfrak{L}_x^{i_\vartheta'} \neq \emptyset$. Let $\mathfrak{L}_x^{i_{\vartheta+1}'} = \mathscr{L}_x^{i_{\vartheta+1}'} \cap \mathfrak{L}_x^{i_\vartheta'}$ and let x_k satisfy $x_k \in \mathfrak{L}_x^{i_{\vartheta+1}'}$. Define a new iterative function $\Gamma_j(x_k) = V_\infty^{\mathcal{L}}(x_k)$, $j = 0, 1, \ldots$, where $\Gamma_0(x_k) = V_\infty^{\mathcal{L}}(x_k)$, $\forall x_k \in \Omega_x$. Define a new iterative control law $\lambda_j(x_k)$, $j = 0, 1, \ldots$, where $\lambda_0(x_k) = v_\infty^{\mathcal{L}}(x_k)$, $\forall x_k \in \Omega_x$. For $j = 1, 2, \ldots$ and for all $x_k \in \mathfrak{L}_x^{i_{\vartheta+1}'}$, we let

$$\lambda_{j+1}(x_k) = \arg\min_{u_k} \{U(x_k, u_k) + \Gamma_j(x_{k+1})\}, \tag{4.61}$$

and $\lambda_{j+1}(x_k) = \lambda_j(x_k)$ for all $x_k \in \Omega_x \backslash \mathfrak{L}_x^{i_{\vartheta+1}'}$. For $j = 0, 1, \ldots$ and all $x_k \in \Omega_x$, let

$$\Gamma_{j+1}(x_k)(x_k) = U(x_k, \lambda_{j+1}(x_k)) + \Gamma_{j+1}(F(x_k, \lambda_{j+1}(x_k))). \tag{4.62}$$

Considering $j = 0$, for all $x_k \in \Omega_x$ and $l = 0, 1, \ldots$, we let

$$\Upsilon_1^{l+1}(x_k) = U(x_k, \lambda_1(x_k)) + \Upsilon_1^l(x_{k+1}), \tag{4.63}$$

where $x_{k+1} = F(x_k, \lambda_1(x_k))$ and $\Upsilon_1^0(x_k) = \Gamma_0(x_k)$. Let $l = 0$. For all $x_k \in \mathfrak{L}_x^{i'_{\vartheta+1}}$, we can get

$$\begin{aligned}\Upsilon_1^1(x_k) &= U(x_k, \lambda_1(x_k)) + \Upsilon_1^0(x_{k+1}) \\ &= \min_{u_k}\{U(x_k, u_k) + \Upsilon_1^0(x_{k+1})\} \\ &= \min_{u_k}\{U(x_k, u_k) + V_\infty^{\mathcal{L}}(x_{k+1})\}. \end{aligned} \tag{4.64}$$

According to Theorem 4.3, for all $x_k \in \cap_{\eta=0}^{\infty} \mathscr{L}_x^{i_\eta}$, the iterative value function $V_\infty^{\mathcal{L}}(x_k)$ satisfies the HJB equation (4.55). For $x_k \in \mathfrak{L}_x^{i'_{\vartheta+1}}$, where $\pi'_x < \infty$, we know that $x_k \in \mathfrak{L}_x^{i'_{\vartheta+1}} \backslash \big(\cap_{\eta=0}^{\infty} \mathscr{L}_x^{i_\eta}\big)$. In this situation, the iterative value function $V_\infty^{\mathcal{L}}(x_k)$ may not satisfy the HJB equation (4.55) under the iterative control law $v_\infty^{\mathcal{L}}(x_k)$. Thus, for all $x_k \in \mathfrak{L}_x^{i'_{\vartheta+1}}$, we can obtain

$$\begin{aligned}\Upsilon_1^1(x_k) &= \min_{u_k}\{U(x_k, u_k) + V_\infty^{\mathcal{L}}(x_{k+1})\} \\ &\le U(x_k, v_\infty^{\mathcal{L}}(x_k)) + V_\infty^{\mathcal{L}}(F(x_k, v_\infty^{\mathcal{L}}(x_k))) \\ &= V_\infty^{\mathcal{L}}(x_k). \end{aligned} \tag{4.65}$$

For all $x_k \in \Omega_x \backslash \mathfrak{L}_x^{i'_{\vartheta+1}}$, as $\lambda_1(x_k) = \lambda_0(x_k)$, we can get

$$\begin{aligned}\Upsilon_1^1(x_k) &= U(x_k, \lambda_1(x_k)) + \Upsilon_1^0(x_{k+1}) \\ &= U(x_k, \lambda_0(x_k)) + \Upsilon_1^0(x_{k+1}) \\ &= V_\infty^{\mathcal{L}}(x_k). \end{aligned} \tag{4.66}$$

Thus, we have $\Upsilon_1^1(x_k) \le V_\infty^{\mathcal{L}}(x_k)$, $\forall x_k \in \Omega_x$. According to the idea (4.15)–(4.25), we can prove $\lim_{l\to\infty} \Upsilon_1^{l+1}(x_k) = \Gamma_1(x_k)$ and $\Gamma_1(x_k) \le V_\infty^{\mathcal{L}}(x_k)$, $\forall x_k$. According to (4.26)–(4.30), for $j = 0, 1, \ldots$ and $\forall x_k \in \Omega_x$, we can obtain that

$$\Gamma_{j+1}(x_k) \le \Gamma_j(x_k). \tag{4.67}$$

Letting $j \to \infty$, we can obtain

$$\Gamma_\infty(x_k) \leq V_\infty^{\mathcal{L}}(x_k), \tag{4.68}$$

which shows that $V_\infty^{\mathcal{L}}(x_k)$ is not guaranteed to the optimal performance index function on Ω_x. The proof is complete. □

Remark 4.2. From Theorem 4.4, we can see that if there exists a state $x_k \in \Omega_x$, such that the iterative control law $v_i(x_k)$ is not updated for infinite times as $i \to \infty$, then we say that the iterative value function cannot converge to the optimal performance index function. Thus, to guarantee that the iterative value function converges to the optimum, it is required that all the states in Ω_x should be chosen infinite times to update the iterative control law. In this situation, we say the iterative performance index function can converge to the optimal performance index function. The detail analysis is shown in the following theorem.

Theorem 4.5 (Global optimal property). *For $i = 0, 1, \ldots$ and all $x_k \in \Omega_x$, let $V_i(x_k)$ and $v_{i+1}(x_k)$ be obtained by (4.7)–(4.11). Let $\mathcal{N}_j$, $j = 0, 1, \ldots$, be non-negative integers, which satisfy $\mathcal{N}_0 \leq \mathcal{N}_1 \leq \cdots$. Assume that $i_{\mathcal{N}_j} \leq i_{\mathcal{N}_{j+1}}$, $\forall j = 0, 1, \ldots$. If for any $j = 0, 1, \ldots$, the state set $\mathscr{L}_x^i$, $i = 0, 1, \ldots$, satisfies*

$$\bigcup_{l=i_{\mathcal{N}_{j-1}}+1}^{i_{\mathcal{N}_j}} \mathscr{L}_x^l = \Omega_x, \tag{4.69}$$

where we let $i_{\mathcal{N}_{j-1}} = -1$ for $j = 0$, then the iterative value function $V_{i_{\mathcal{N}_j}}(x_k) \to J^*(x_k)$, $\forall x_k \in \Omega_x$ as $j \to \infty$.

Proof. The statement can be proven by the following three steps.

First, as $i_{\mathcal{N}_j} \leq i_{\mathcal{N}_{j+1}}$, $\forall j = 0, 1, \ldots$, we have $i_{\mathcal{N}_j} \to \infty$ as $j \to \infty$. According to Theorem 4.1, for all $x_k \in \Omega_x$, the limit of the iterative value function exits as $j \to \infty$, i.e.,

$$V_\infty(x_k) = \lim_{j \to \infty} V_{i_{\mathcal{N}_j}}(x_k). \tag{4.70}$$

According to (4.69), for any $x_k \in \Omega_x$, we can find i_{ρ_j}, where $i_{\rho_j} \in \{0, 1, \ldots\}$, $i_{\mathcal{N}_{j-1}+1} \leq i_{\rho_j} \leq i_{\mathcal{N}_j}$, $j = 0, 1, \ldots$, which satisfies $x_k \in \mathscr{L}_x^{i_{\rho_j}}$. Letting $j \to \infty$, we can get

$$x_k \in \bigcap_{j=0}^{\infty} \mathscr{L}_x^{i_{\rho_j}}. \tag{4.71}$$

According to Theorem 4.2, for $x_k \in \cap_{\rho=0}^{\infty} \mathscr{L}_x^{i_{\rho_j}}$, the limit of the iterative value function $V_{i_{\mathcal{N}_j}}(x_k)$ satisfies the HJB equation

$$V_\infty(x_k) = \min_{u_k}\{U(x_k, u_k) + V_\infty(x_{k+1})\}. \tag{4.72}$$

As x_k is arbitrary in Ω_x, we have (4.72) which holds for all $x_k \in \Omega_x$.

Next, let $\mu(x_k)$ be an arbitrary admissible control law, and define a new value function $\mathcal{P}_{i+1}(x_k)$, which satisfies

$$\mathcal{P}_{i+1}(x_k) = U(x_k, \mu(x_k)) + \mathcal{P}_i(x_{k+1}), \tag{4.73}$$

where $\mathcal{P}_0(x_k) = V_\infty(x_k)$. For $i = 0$, we have

$$\begin{aligned}
\mathcal{P}_1(x_k) &= U(x_k, \mu(x_k)) + \mathcal{P}_0(F(x_k, \mu(x_k))) \\
&\geq U(x_k, \mu(x_k)) + V_\infty(F(x_k, \mu(x_k))) \\
&\geq \min_{u_k}\{U(x_k, u_k) + V_\infty(x_{k+1})\} \\
&= V_\infty(x_k).
\end{aligned} \tag{4.74}$$

Assume

$$\mathcal{P}_l(x_k) \geq V_\infty(x_k) \tag{4.75}$$

holds for $i = l - 1$, $l = 1, 2, \ldots$. For $j = l$, we have

$$\begin{aligned}
\mathcal{P}_{l+1}(x_k) &= U(x_k, \mu(x_k)) + \mathcal{P}_l(F(x_k, \mu(x_k))) \\
&\geq U(x_k, \mu(x_k)) + V_\infty(F(x_k, \mu(x_k))) \\
&\geq \min_{u_k}\{U(x_k, u_k) + V_\infty(x_{k+1})\} \\
&= V_\infty(x_k).
\end{aligned} \tag{4.76}$$

Thus, for $i = 0, 1, \ldots$, and for all $x_k \in \Omega_x$, we can derive that

$$\mathcal{P}_{i+1}(x_k) \geq V_\infty(x_k). \tag{4.77}$$

The mathematical induction is complete.

Next, according to the definitions of $V_i(x_k)$ and $J^*(x_k)$ in (4.9) and (4.3), respectively, for $\forall i = 0, 1, \ldots$, we have

$$V_i(x_k) \geq J^*(x_k). \tag{4.78}$$

Let $i \to \infty$, and then we can obtain

$$V_\infty(x_k) \geq J^*(x_k). \tag{4.79}$$

According to (4.77), we have

$$\begin{aligned} \mathcal{P}_{i+1}(x_k) &= \sum_{j=0}^{N} U(x_{k+j}, \mu(x_{k+j})) + P_0(x_{k+N+1}) \\ &\geq V_\infty(x_k). \end{aligned} \tag{4.80}$$

Letting $i \to \infty$, we can obtain

$$\sum_{j=0}^{\infty} U(x_{k+j}, \mu(x_{k+j})) \geq V_\infty(x_k) \tag{4.81}$$

which holds for an arbitrary control law $\mu(x_k)$. Let $\mu(x_k) = u^*(x_k)$, where $u^*(x_k)$ is an optimal control law. Then, we can get

$$V_\infty(x_k) \leq J^*(x_k). \tag{4.82}$$

According to (4.79) and (4.82), we can obtain $V_\infty(x_k) = J^*(x_k)$, for all $x_k \in \Omega_x$. The proof is complete. □

4.4 Simulation Example

To evaluate the performance of our local iterative ADP algorithm, an example is given for numerical experiment to solve the approximate optimal control problems. We consider the following non-affine nonlinear system [2]:

$$\begin{aligned} \dot{x}_1 &= -x_1 + x_2 u, \\ \dot{x}_2 &= -x_2 + (1 + x_2^2)u + u^3. \end{aligned} \tag{4.83}$$

Discretizing the system with the sampling interval $\Delta T = 0.1$s leads to

$$\begin{aligned} x_{1(k+1)} &= (1 - \Delta T)x_{1k} + \Delta T x_{2k} u_k, \\ x_{2(k+1)} &= (1 - \Delta T)x_{2k} + \Delta T(1 + x_{1k}^2)u_k + \Delta T u_k^3. \end{aligned} \tag{4.84}$$

The utility function is chosen as $U(x_k, u_k) = x_k^{\mathsf{T}} Q x_k + u_k^{\mathsf{T}} R u_k$, where $Q = 2I_1$, $R = 2I_2$, and I_1, I_2 are identity matrices with suitable dimensions. We choose the state space as Ω_x. Let the initial state be $x_0 = [1, -1]^{\mathsf{T}}$. According to Algorithm 1 in Ref. [2], we can obtain the initial admissible control law by action network, where the initial admissible control law can be expressed as $v_0(x_k) = W_{a,\text{initial}}\sigma(Y_{a,\text{initial}}x_k + b_{a,\text{initial}})$. The weight matrices of the action network for the admissible control law are obtained as

$$\begin{aligned} W_{a,\text{initial}} = &\big[-1.70\ -0.18\ 0.00\ -0.93\ -0.00 \\ &\quad 0.02\ -0.02\ -0.01\ 0.07\ 0.03\big], \\ Y_{a,\text{initial}} = &\begin{bmatrix} -0.05 & -0.33 & -2.44 & -0.45 & -10.66 \\ 0.52 & 4.32 & -2.73 & 0.03 & -8.63 \\ -0.17 & -3.31 & 4.04 & 0.29 & 5.31 \\ -4.07 & -0.19 & 5.58 & -3.41 & -0.01 \end{bmatrix}^{\mathsf{T}}, \end{aligned}$$

and

$$\begin{aligned} b_{a,\text{initial}} = &\big[-0.28\ 5.12\ 0.82\ 0.28\ 1.51 \\ &\quad -0.52\ -0.98\ 4.92\ -2.15\ 4.16\big]^{\mathsf{T}}. \end{aligned}$$

For $i = 0, 1, \ldots$, we update the iterative value function and iterative control law for all $x_k \in \mathscr{L}_x^{\text{rem}(i,4)}$. According to the initial admissible control law $v_0(x_k)$, we implement the local policy iteration algorithm for 30 iterations to reach the computation precision $\varepsilon = 0.01$. The plots of the iterative value function are shown in Fig. 4.1(a). The corresponding trajectories of the states and controls are shown in Figs. 4.1(b) and 4.1(c), respectively. From Fig. 4.1(a), we can see that the iterative value function is monotonically non-increasing and converges to the optimal performance index function. The corresponding states and controls also converge to their optimums. The optimal trajectories of the states and control are shown in Fig. 4.1(d).

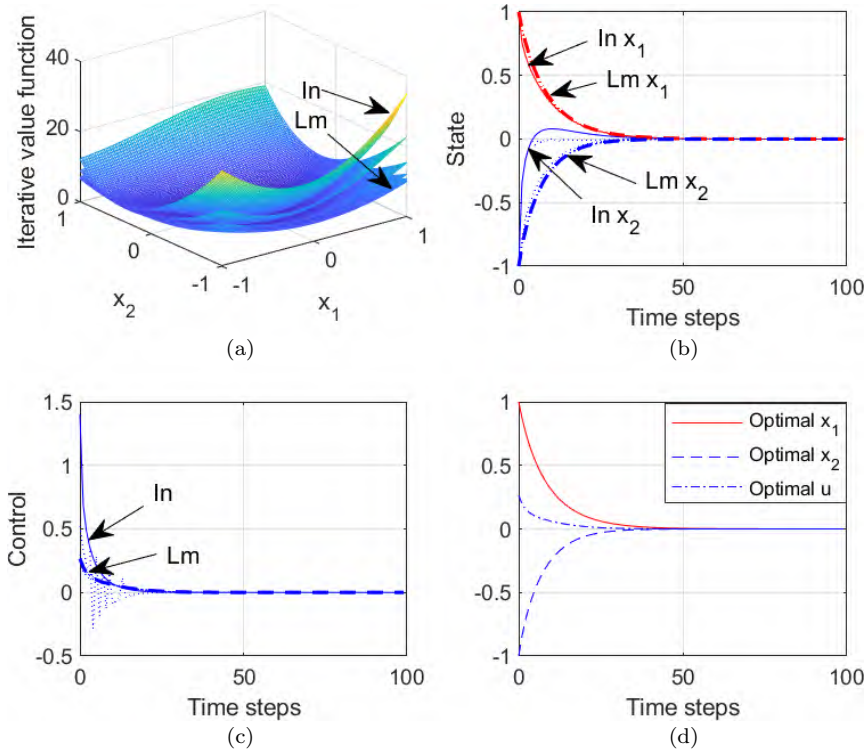

Fig. 4.1. Simulation results for local policy iteration algorithm. (a) Iterative value function. (b) State trajectories. (c) Control trajectories. (d) Optimal state and control trajectories.

Now, we implement the traditional global policy iteration algorithm [2] for 30 iterations. The plots of the iterative value functions are shown in Fig. 4.2(a). The iterative state and control trajectories of the global policy iteration are shown in Figs. 4.2(b) and 4.2(c), respectively. The optimal states and control by the traditional global policy iteration algorithm [2] are shown in 4.2(d), which is the same as the optimal states and control by the developed local policy iteration algorithm in this chapter. From Figs. 4.1 and 4.2, we can see that both of the local and global can make the iterative value function converge to the optimum and obtain the optimal control law of the system, although the convergence processes between the global

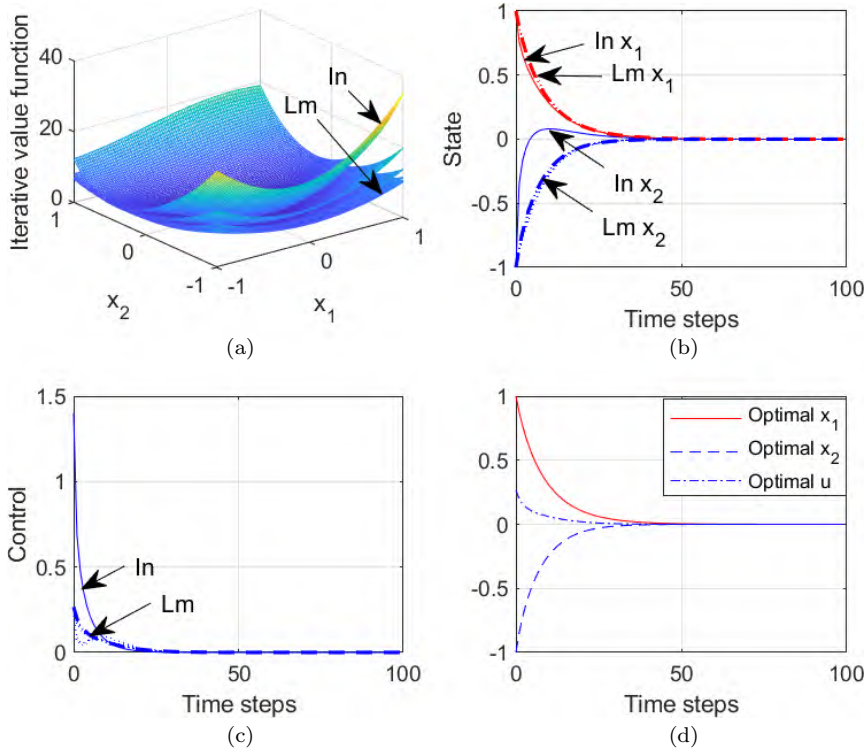

Fig. 4.2. Simulation results for traditional global policy iteration algorithm [2]. (a) Iterative value function. (b) State trajectories. (c) Control trajectories. (d) Optimal state and control trajectories.

and local policy iteration algorithms are different. This shows the effectiveness of the developed algorithm.

Now, we let $\mathscr{L}_x^i \equiv \mathscr{L}_x^0$, $\forall\, i = 0, 1, \ldots$. Implementing the developed local policy iteration algorithm for 30 iterations, the iterative value functions are shown in Fig. 4.3(a). We can see that the iterative value function $V_i(x_k)$ is monotonically non-increasing and convergent as i increases. The convergence trajectories of the iterative states and controls are shown in Figs. 4.3(b) and 4.3(c), respectively. Although the convergence of the iterative value function and the corresponding state and control can be guaranteed for $\mathscr{L}_x^i \equiv \mathscr{L}_x^0$, $\forall\, i = 0, 1, \ldots,$

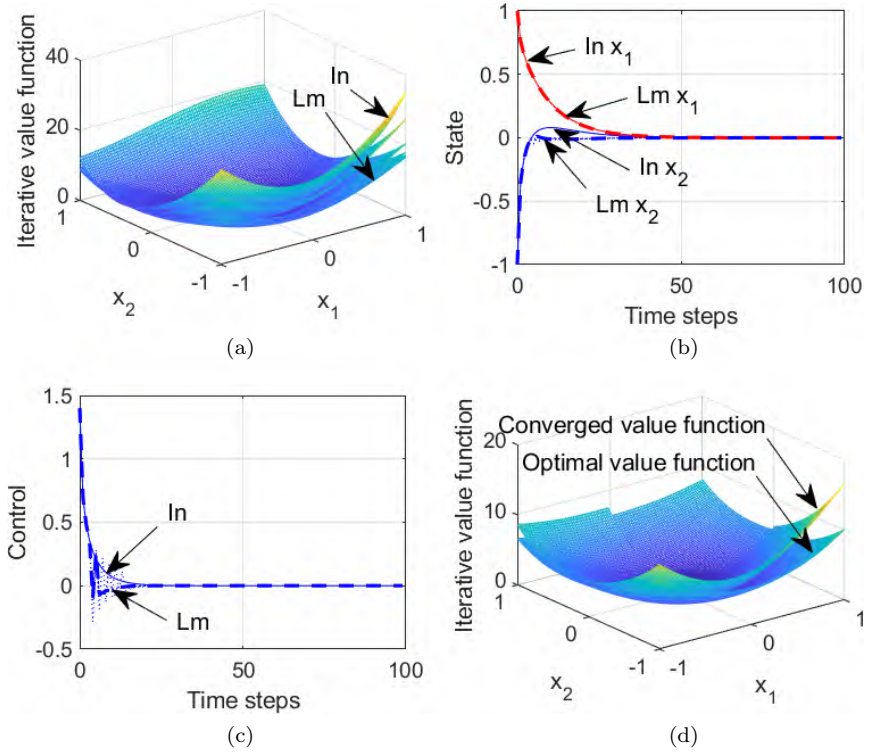

Fig. 4.3. Simulation results for local policy iteration algorithm for $\mathscr{L}_x^i \equiv \mathscr{L}_x^0$. (a) Iterative value function. (b) State trajectories. (c) Control trajectories. (d) The converged and optimal performance index functions.

from Fig. 4.3(d), we can see that the iterative value function for $\mathscr{L}_x^i \equiv \mathscr{L}_x^0$ does not converge to the optimum. The trajectories of the converged states for $\mathscr{L}_x^i \equiv \mathscr{L}_x^0$ and the optimal states are shown in Fig. 4.4(a). The trajectories of the converged control for $\mathscr{L}_x^i \equiv \mathscr{L}_x^0$ and the optimal control are shown in Fig. 4.4(b). We can see that the converged states and control for $\mathscr{L}_x^i \equiv \mathscr{L}_x^0$ are different from the optimal states and control. Thus, we say that if there exits a state $x_k \in \Omega_x$, such that the iterative control law $v_i(x_k)$ is not updated for infinite times as $i \to \infty$, then the optimality of the converged iterative value function cannot be guaranteed.

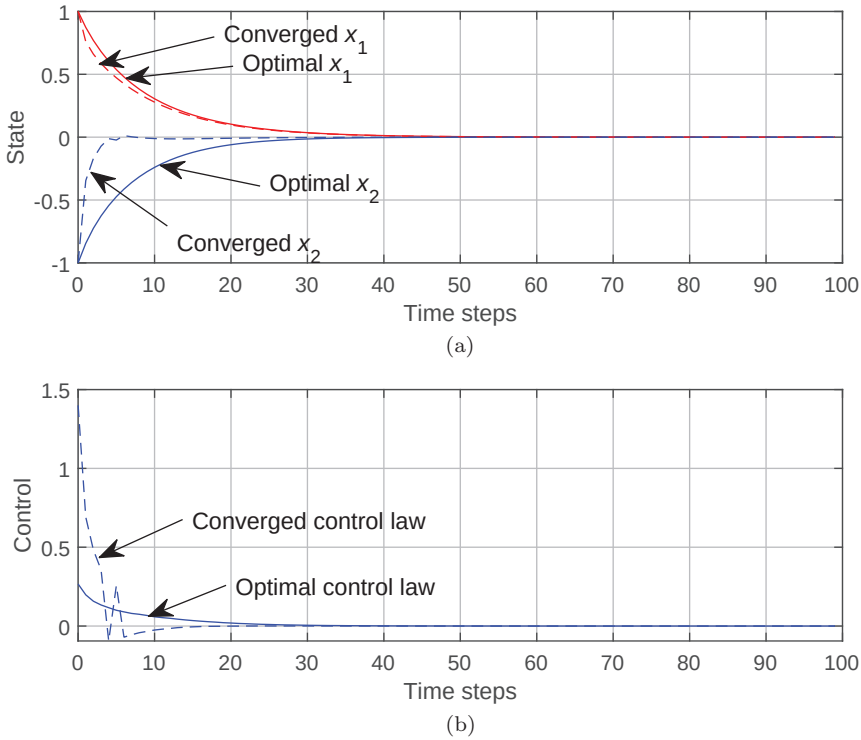

Fig. 4.4. State and control trajectories. (a) The converged and optimal states. (b) The converged and optimal controls.

4.5 Conclusion

In this chapter, a new local policy iteration ADP algorithm is developed to solve optimal control problems for infinite horizon discrete-time nonlinear systems. In each iteration of the developed discrete-time local policy iteration algorithm, the iterative value function and iterative control law can be updated in a given subset of the state space, instead of updated in the whole state space. It has been shown that the iterative value function is a monotonically non-increasing and convergent sequence as the iteration index increases. Under different constraints of the state set $\mathscr{L}_x^i$, $i = 0, 1, \ldots$, it has been proven that the iterative value function may converge to local or global optimum. The admissibility of the iterative control law is proven, where the stability of the control system can be guaranteed

under the iterative control law. Comparisons between the local and global policy iteration algorithms have been displayed in the simulation studies and the effectiveness of the developed local policy iteration algorithm has been justified by the simulation results.

References

[1] Bertsekas, D.P. and Tsitsiklis, J.N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

[2] Liu, D. and Wei, Q. (2014). Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems* 25(3), 621–634.

[3] Si, J. and Wang, Y.T. (2001). On-line learning control by association and reinforcement. *IEEE Transactions on Neural Networks* 12(2), 264–276.

[4] Sutton, R.S. and Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

[5] Wei, Q., Liu, D., and Yang, X. (2015). Infinite horizon self-learning optimal control of nonaffine discrete-time nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems* 26(4), 866–879.

This page intentionally left blank

Chapter 5

Continuous-Time Time-Varying Policy Iteration

5.1 Introduction

Most existing researches about policy iteration ADP algorithms focused on time-invariant nonlinear systems [1–6]. The main difficulty in investigating the policy iteration of time-invariant nonlinear systems lies in the complexity of solving the Hamilton-Jacobi-Bellman (HJB) equation. For the time-invariant systems, the performance index function and optimal control laws can be derived by the states of systems. In this situation, the HJB equation of time-invariant systems is an ordinary differential equation, which can approximated by some time-invariant iterative functions. However, time-varying systems are ubiquitous in real world. For time-varying systems, the optimal control law and performance index function are described by time-varying functions, which are very different from time-invariant systems. Furthermore, the HJB equation is described by a partial differential equation in time-varying systems, which is not applicable to approximate by a sequence of time-invariant iterative functions. Hence, the policy iteration and the property analysis for time-invariant systems may not be available for time-invariant systems. As far as we know, there are still no results on the policy iteration algorithms to obtain the infinite horizon optimal control laws for time-varying nonlinear systems in continuous-time cases.

In this chapter, a novel policy iteration algorithm, called "continuous-time time-varying (CTTV) policy iteration algorithm",

is presented to obtain the optimal control laws for infinite horizon continuous-time time-varying nonlinear systems. The ADP technique is utilized to get the iterative control laws for the optimization of the performance index function. The properties of the CTTV policy iteration algorithm are analyzed. Monotonicity, convergence, and optimality of the iterative value function have been analyzed, and the iterative value function can be proven to monotonically converge to the optimal solution of the HJB equation. Furthermore, the iterative control law is guaranteed to be admissible to stabilize the nonlinear systems. In the implementation of the presented CTTV policy algorithm, the approximate iterative control laws and iterative value function are obtained by neural networks. Finally, numerical results are given to verify the effectiveness of the presented method.

5.2 Problem Formulations

In this chapter, the following continuous-time time-varying (CTTV) nonlinear systems are investigated

$$\dot{x} = F(x, u, t). \tag{5.1}$$

Let $x = x(t) \in \mathbb{R}^n$ denote the system state, and $u = u(t) \in \mathbb{R}^m$ represent the control input. Let $F(x, u, t)$ represent the system function. Let x_0 be the initial state. To analyze the dynamics of system (5.1), the following assumption is given.

Assumption 5.1. System (5.1) is controllable on a compact set $\Omega \subset \mathbb{R}^n$ including the origin; the system function $F(x, u, t)$ satisfies Lipschitz continuous conditions for x and u; equilibrium point of system (5.1) with the control inputs $u = 0$ is $x = 0$, i.e., $F(0, 0, t) = 0$, $\forall t \geq 0$; the control law $u(x, t)$ can be continuous on Ω and $u = u(x, t) = 0$ is satisfied for $x = 0$.

To analyze the optimal control laws of system (5.1), the performance index function can be regarded as

$$J(x, t) = \int_t^\infty U(x(\tau), u(\tau), \tau) \mathrm{d}\tau, \tag{5.2}$$

where $U(x, u, t)$, $\forall t \geq 0$, represents the utility function of the system, and it is defined as a positive definite function. The following definition is exhibited to describe the policy iteration algorithm.

Definition 5.1. A control law $u(x,t)$ is called an admissible control law for system (5.1) on Ω, if $u(x,t)$ is continuous on Ω with $u(0,t)=0$, and $u(x,t)$ stabilizes system (5.1), which simultaneously guarantees $J(x,0)$ to be finite under $u(x,t)$, $\forall\, x_0 \in \Omega$.

Assuming that admissible control laws on Ω are denoted by $\Psi(\Omega)$ and the admissible control input satisfying $\mu \in \Psi(\Omega)$, if the following value function

$$V(x,t) = \int_t^\infty U(x(\tau), \mu(\tau), \tau)\mathrm{d}\tau \tag{5.3}$$

is a continuously differentiable function, the nonlinear Lyapunov function can be derived into the following infinitesimal version

$$0 = U(x,\mu,t) + \left(\frac{\partial V(x,t)}{\partial x}\right)^{\mathsf{T}} F(x,\mu,t) + \frac{\partial V(x,t)}{\partial t}, \tag{5.4}$$

where $V(0,t)=0$ holds for all $t \geq 0$. Thus, according to principle of optimality, the optimal performance index function can be considered as

$$J^*(x,t) = \min_{\mu\in\Psi(\Omega)} \left\{ \int_t^\infty U(x(\tau), \mu(x(\tau),\tau), \tau)\mathrm{d}\tau \right\}, \tag{5.5}$$

which satisfies the Hamilton–Jacobi–Bellman (HJB) equation

$$\begin{aligned} &-\frac{\partial J^*(x,t)}{\partial t} \\ &\quad = \min_u \left\{ U(x,u,t) + \left(\frac{\partial J^*(x,t)}{\partial x}\right)^{\mathsf{T}} F(x,u,t) \right\} \\ &\quad = U(x,u^*(x,t),t) + \left(\frac{\partial J^*(x,t)}{\partial x}\right)^{\mathsf{T}} F(x,u^*(x,t),t), \end{aligned} \tag{5.6}$$

with $J^*(0,t) = 0$, $\forall\, t \geq 0$, where $u^*(x,t)$ represents the optimal control law.

5.3 Continuous-Time Time-Varying Policy Iteration: Derivations and Properties

The detailed derivations of the developed CTTV policy iteration algorithm are introduced in this section. The admissibility of the

proposed iterative method and the corresponding convergence analysis are also discussed, which aim to show that the iterative value function converges to the optimum.

5.3.1 *Derivations of the CTTV policy iteration algorithm*

In the developed policy iteration methods, it is necessary to update the value function and control law at every iteration and the iteration index i increases from 0 to ∞.

The algorithm is initialized by $v_0(x,t) \in \Psi(\Omega)$, $\forall x \in \mathbb{R}^n$ and $\forall t \geq 0$, which denotes an arbitrary initial admissible control law. The function $V_0(x,t)$ is defined as the iterative value function, such that

$$U(x, v_0(x,t), t) + \left(\frac{\partial V_0(x,t)}{\partial x}\right)^{\mathsf{T}} F(x, v_0(x,t), t) + \frac{\partial V_0(x,t)}{\partial t} = 0. \tag{5.7}$$

For $i = 0, 1, \ldots,$ we can calculate the iterative control law $v_{i+1}(x,t)$ by

$$v_{i+1}(x,t) = \arg\min_u \left\{ U(x,u,t) + \left(\frac{\partial V_i(x,t)}{\partial x}\right)^{\mathsf{T}} F(x,u,t) \right\}. \tag{5.8}$$

According to $v_{i+1}(x,t)$, $i = 0, 1, \ldots,$ the update laws of iterative value function $V_{i+1}(x,t)$ is

$$U(x, v_{i+1}(x,t), t) + \left(\frac{\partial V_{i+1}(x,t)}{\partial x}\right)^{\mathsf{T}} F(x, v_{i+1}(x,t), t) + \frac{\partial V_{i+1}(x,t)}{\partial t} = 0. \tag{5.9}$$

The iterative laws of the CTTV policy iteration algorithm are introduced in (5.7)–(5.9), which are different from the traditional updated laws of policy iteration for time-invariant nonlinear systems.

Remark 5.1. The iterative laws of the CTTV policy iteration algorithm are introduced in (5.7)–(5.9), which are different from the traditional update laws of policy iteration for time-invariant nonlinear

systems. First, for the traditional polity iterations, the iterative value function can be defined as the function of the system state x, which means that the term $\frac{\mathrm{d}V_i(x)}{\mathrm{d}x}$ is used for the achievement of the iterative control law. In CTTV policy iteration algorithm (5.7)–(5.9), iterative value function is related with both the system state x and the time t such that the term $\frac{\partial V_i(x,t)}{\partial x}$ is adopted to derive the iterative control law. Second, for the traditional polity iterations, the iterative value function $V_i(x)$ can be calculated by solving the following ordinary differential equation

$$U(x, v_i(x)) + \left(\frac{\mathrm{d}V_i(x)}{\mathrm{d}x}\right)^{\mathsf{T}} F(x, v_i(x)) = 0. \tag{5.10}$$

For the developed algorithm (5.7)–(5.9), the iterative value function $V_i(x,t)$ is obtained by solving a partial differential equation (5.7) or (5.9), which is more difficult to obtain the solution. When the system is time-invariant, the partial differential equation (5.9) is reduced to (5.10). Hence, the traditional policy iteration algorithms can be regarded as special case of the proposed CTTV policy iteration algorithm. In addition, considering the parameter t, the traditional property analysis methods may be not applicable for the developed CTTV policy iteration algorithm. Thus, some novel methods for property analyses are established in the following.

5.3.2 *Property analysis*

In this subsection, the characteristics of the CTTV policy iteration algorithm containing admissibility, convergence, and optimality are analyzed. In Ref. [1], considering the continuous-time time-invariant affine nonlinear systems, it was proven for the policy iteration algorithm that the iterative value function will monotonically converge to the optimum. Many subsequent discussions on policy iteration algorithm followed the proven idea. Different from the results in Ref. [1] obviously, the control system (5.1), investigated throughout this paper, is a time-varying system. Furthermore, the control system (5.1) is not necessary an affine nonlinear system. Thus, the property analysis in Ref. [1] may be not available for the proposed CTTV policy iteration algorithm and new analysis method will be

established. For further discussion, it is necessary to give the following lemmas.

Lemma 5.1. *For $i = 0, 1, \ldots$, let $V_i(x,t)$ and $v_i(x,t)$ be updated by* (5.7)–(5.9). *If the iterative control law $v_i(x,t)$ is admissible for system* (5.1), *there exists an iterative value function $V_i(x,t)$ which satisfies* (5.7) *and* (5.9).

Proof. If the control law $v_i(x,t)$, $i = 0, 1, \ldots$, is admissible control for system (5.1), we have

$$\int_t^\infty U(x(\tau), v_i(x(\tau),\tau),\tau)d\tau < \infty. \tag{5.11}$$

Define

$$V_i(x,t) = \int_t^\infty U(x(\tau), v_i(x(\tau),\tau),\tau)\mathrm{d}\tau. \tag{5.12}$$

For $i = 0, 1, \ldots$, taking the derivative of $V_i(x,t)$ along t, it is easy to derive

$$\begin{aligned}&\left(\frac{\partial V_i(x,t)}{\partial x}\right)^{\mathsf{T}} F(x, v_i(x,t),t) + \frac{\partial V_i(x,t)}{\partial t}\\ &\quad = U(x(\infty), v_i(x(\infty),\infty),\infty) - U(x(t), v_i(x(t),t),t),\end{aligned} \tag{5.13}$$

which are (5.7) for $i = 0$ and (5.9) for $i = 1, 2, \ldots$, as $U(x(\infty), v_i(x(\infty),\infty),\infty) = 0$ for the admissible control law $v_i(x,t)$. □

Lemma 5.2. *Assume that $V_i(x,t)$ and $v_i(x,t)$ are updated by* (5.7)–(5.9) *for $i = 0, 1, \ldots$. If $v_i(x,t)$ is admissible for system* (5.1) *and Assumption* 5.1 *holds, $V_i(x,t)$ is always positive for any $i = 0, 1, \ldots$.*

Proof. If $v_i(x,t)$ represents the admissible control law, $i = 0, 1, \ldots$, according to Lemma 5.1, $V_i(x,t)$ satisfies (5.12). According to Assumption 5.1, for $x = 0$, we know $v_i(x,t) = 0$ and $F(x, v_i(x,t),t) = 0$. For the positive definite utility function $U(x,u,t)$, it implies $V_i(x,t) = 0$. For any $x \neq 0$, due to the positive definite characteristic of $U(x,u,t)$, we have $U(x, v_i(x,t),t) > 0$. According to (5.12), it shows $V_i(x,t) > 0$. The proof is complete. □

First, the relationship between the performance index function and the iterative value function will be discussed.

Theorem 5.1. *$V_i(x,t)$ and $v_i(x,t)$ are updated based on* (5.7)–(5.9) *for $i = 0, 1, \ldots$ If control input $v_i(x,t)$ is admissible and there is an iterative value function $V_j(x,t)$, for $j = 0, 1, \ldots,$ satisfying*

$$\begin{aligned} &U(x, v_{i+1}(x), t) + \left(\frac{\partial V_i(x,t)}{\partial s}\right)^{\mathsf{T}} F(x, v_{i+1}(x), t) + \frac{\partial V_i(x,t)}{\partial t} \\ &\quad \leq U(x, v_{j+1}(x), t) + \left(\frac{\partial V_j(x,t)}{\partial x}\right)^{\mathsf{T}} F(x, v_{j+1}(x), t) + \frac{\partial V_j(x,t)}{\partial t} \end{aligned} \tag{5.14}$$

then the following conclusion

$$V_j(x,t) \leq V_i(x,t) \tag{5.15}$$

can be obtained.

Proof. Taking derivatives of $V_i(x,t)$ and $V_j(x,t)$ along $v_{i+1}(x,t)$, we have

$$\begin{aligned} \frac{\mathrm{d}V_i(x,t)}{\mathrm{d}t} - \frac{\mathrm{d}V_j(x,t)}{\mathrm{d}t} &= \left(\frac{\partial V_i(x,t)}{\partial x}\right)^{\mathsf{T}} F(x, v_{i+1}(x,t), t) \\ &\quad - \left(\frac{\partial V_j(x,t)}{\partial x}\right)^{\mathsf{T}} F(x, v_{i+1}(x,t), t) \\ &\quad + \frac{\partial V_i(x,t)}{\partial t} - \frac{\partial V_j(x,t)}{\partial t}. \end{aligned} \tag{5.16}$$

According to (5.8), for $j = 0, 1, \ldots,$ we can get

$$v_{j+1}(x,t) = \arg\min_u \left\{ U(x,u,t) + \left(\frac{\partial V_j(x,t)}{\partial x}\right)^{\mathsf{T}} F(x,u,t) \right\}. \tag{5.17}$$

Then, according to (5.16) and (5.17), we can derive (5.18).

$$\begin{aligned}
&\frac{\mathrm{d}V_j(x,t)}{\mathrm{d}t} - \frac{\mathrm{d}V_i(x,t)}{\mathrm{d}t} \\
&= \left(\frac{\partial V_i(x,t)}{\partial x}\right)^T F(x, v_{i+1}(x,t), t) + \frac{\partial V_i(x,t)}{\partial t} \\
&\quad + U(x, v_{i+1}(x,t), t) \\
&\quad - \left(\frac{\partial V_j(x,t)}{\partial x}\right)^T F(x, v_{i+1}(x,t), t) - \frac{\partial V_j(x,t)}{\partial t} \\
&\quad - U(x, v_{i+1}(x,t), t) \\
&= \min_u \left\{ \left(\frac{\partial V_i(x,t)}{\partial x}\right)^T F(x,u,t) + \frac{\partial V_i(x,t)}{\partial t} + U(x,u,t) \right\} \\
&\quad - \min_u \left\{ \left(\frac{\partial V_j(x,t)}{\partial x}\right)^T F(x,u,t) + \frac{\partial V_j(x,t)}{\partial t} + U(x,u,t) \right\} \\
&\quad + \left(\min_u \left\{ \left(\frac{\partial V_j(x,t)}{\partial x}\right)^T F(x,u,t) + \frac{\partial V_j(x,t)}{\partial t} + U(x,u,t) \right\} \right. \\
&\quad - \left(\left(\frac{\partial V_j(x,t)}{\partial x}\right)^T F(x, v_{i+1}(x,t), t) + \frac{\partial V_j(x,t)}{\partial t} \right. \\
&\quad \left. \left. + U(x, v_{i+1}(x,t), t) \right) \right). \qquad (5.18)
\end{aligned}$$

Defining $V_i(x,\infty) = \lim_{t\to\infty} V_i(x,t)$, as $v_i(x,t)$ is an admissible control law, according to (5.9), for $i = 0, 1, \ldots$, we can derive

$$\begin{aligned}
\int_t^\infty \frac{\mathrm{d}V_i(x,\tau)}{\mathrm{d}\tau}\mathrm{d}\tau &= V_i(x,\infty) - V_i(x,t) \\
&= -\int_t^\infty U(x, v_i(x), \tau)\mathrm{d}\tau. \qquad (5.19)
\end{aligned}$$

Then, we get

$$V_i(x,t) = \int_t^\infty U(x, v_i(x), \tau)\mathrm{d}\tau, \qquad (5.20)$$

and $V_i(x,\infty)=0$. According to (5.18), we can obtain

$$\int_t^{\infty}\frac{\mathrm{d}V_i(x,\tau)}{\mathrm{d}\tau}\mathrm{d}\tau\le\int_t^{\infty}\frac{\mathrm{d}V_j(x,t)}{\mathrm{d}\tau}\mathrm{d}\tau. \tag{5.21}$$

According to (5.18)–(5.20), it is shown that

$$V_i(x,\infty)-V_i(x,t)\le V_j(x,\infty)-V_j(x,t), \tag{5.22}$$

which can obtain (5.15) immediately. The proof is complete. □

Corollary 1. *Let $v_i(x,t)$ represent the iterative control law in* (5.8), $i=0,1,\ldots$. *If $v_i(x,t)$ in* (5.8) *is admissible, then the iterative value function $V_i(x,t)$ satisfies*

$$V_i(x,t)\ge J^*(x,t). \tag{5.23}$$

Proof. Let $J^*(x,t)$ denote the optimal performance index function, which make (5.6) hold. For an admissible control law $v_i(x,t)$, the iterative value function $V_i(x,t)$ always satisfies (5.7) or (5.9). Thus, we can derive

$$\min_u\left\{U(x,u,t)+\left(\frac{\partial V_i(x,t)}{\partial x}\right)^{\mathsf{T}}F(x,u,t)\right\}+\frac{\partial V_i(x,t)}{\partial t}\le 0, \tag{5.24}$$

According to the HJB equation (5.6) and (5.24), it shows

$$\begin{aligned}&\min_u\left\{U(x,u,t)+\left(\frac{\partial V_i(x,t)}{\partial x}\right)^{\mathsf{T}}F(x,u,t)\right\}+\frac{\partial V_i(x,t)}{\partial t}\\&\quad\le\min_u\left\{U(x,u,t)+\left(\frac{\partial J^*(x,t)}{\partial x}\right)^{\mathsf{T}}F(x,u,t)\right\}+\frac{\partial J^*(x,t)}{\partial t}.\end{aligned} \tag{5.25}$$

The conclusion of this corollary can be drawn based on Theorem 5.1. □

Based on Corollary 1, it can include that $V_i(x,t)$ will never be smaller than the optimal performance index function with the admissible control laws. Hence, the optimal value of performance index function can be considered as the infimum of the iterative value function under this situation. This is an important property for CTTV policy iteration algorithm. According to the above properties of the iterative value function, the admissibility of the iterative control law in the CTTV policy iteration algorithm will be analyzed.

Theorem 5.2. *For $i = 0, 1, \ldots$, assume that $V_i(x,t)$ and $v_i(x,t)$ are updated by (5.7)–(5.9). If the iterative control law $v_i(x,t)$ is admissible, then $v_{i+1}(x,t)$ is admissible.*

Proof. If iterative control law $v_i(x,t)$ is admissible, the iterative value function $V_i(x,t)$ satisfies (5.12). Defining $v_{i+1}(x,t)$ in (5.8), we have

$$\frac{\partial V_i(x,t)}{\partial t} + \left(\frac{\partial V_i(x,t)}{\partial x}\right)^{\mathsf{T}} F(x, v_{i+1}(x,t), t) + U(x, v_{i+1}(x,t), t) \leq 0. \tag{5.26}$$

From Lemma 5.2, it is obvious to draw the conclusion that $V_i(x,t)$ is always a positive definite function. Choose $V_i(x,t)$ as the Lyapunov function candidate. Calculating the derivative of $V_i(x,t)$ along $v_{i+1}(x,t)$, according to (5.26), we have

$$\begin{aligned}\frac{\mathrm{d}V_i(x,t)}{\mathrm{d}t} &= \frac{\partial V_i(x,t)}{\partial t} + \left(\frac{\partial V_i(x,t)}{\partial x}\right)^{\mathsf{T}} F(x, v_{i+1}(x,t), t) \\ &\leq -U(x, v_{i+1}(x,t), t).\end{aligned} \tag{5.27}$$

Thus, it is shown that the iterative control law $v_{i+1}(x,t)$ is a stable control law for system (5.1) and it is easy to get the derivations $\lim_{t\to\infty} U(x, v_{i+1}(x,t), t) = 0$.

Let $W(x,t)$ stand for a value function, such that

$$W(x,t) = \int_t^\infty U(x, v_{i+1}(x,\tau), \tau)\mathrm{d}\tau. \tag{5.28}$$

Next, we will give the proof that $W(x,t)$ is finite, $\forall x \in \mathbb{R}^n$. By calculating the derivative of $W(x,t)$, we have

$$\frac{\mathrm{d}W(x,t)}{\mathrm{d}t} = -U(x, v_{i+1}(x,t), t). \tag{5.29}$$

According to (5.26), we obtain

$$\frac{\mathrm{d}V_i(x,t)}{\mathrm{d}t} \leq \frac{\mathrm{d}W(x,t)}{\mathrm{d}t}, \quad \forall x \in \mathbb{R}^n. \tag{5.30}$$

Based on Theorem 5.1, we have

$$V_i(x,t) \geq W(x,t), \tag{5.31}$$

which shows $W(x,t)$ is finite. Hence, we can get the conclusion that $v_{i+1}(x,t)$ is admissible, which completes the proof. □

Then, we analyze characteristics of the proposed CTTV policy iteration algorithm such as convergence and optimality.

Theorem 5.3. *Assume that $V_i(x,t)$ and $v_i(x,t)$ are updated by* (5.7)–(5.9). *If $v_0(x,t)$ is admissible, then the iterative value function $V_i(x,t)$ is monotonically non-increasing, i.e.,*

$$V_{i+1}(x,t) \leq V_i(x,t), \quad \forall x \in \mathbb{R}^n, \tag{5.32}$$

and it will converge to the optimum $J^*(x,t)$ as $i \to \infty$, i.e.,

$$\lim_{i\to\infty} V_i(x,t) = J^*(x,t). \tag{5.33}$$

Proof. Consider $i = 0$. If the initial control law $v_0(x,t)$ is admissible, based on Theorem 5.2, $v_1(x,t)$ is also an admissible control law. Taking the derivative of $V_0(x,t)$ along $v_1(x,t)$, according to (5.26), we have

$$\begin{aligned}\frac{\mathrm{d}V_0(x,t)}{\mathrm{d}t} &= \frac{\partial V_0(x,t)}{\partial t} + \left(\frac{\partial V_0(x,t)}{\partial x}\right)^{\mathsf{T}} F(x, v_1(x,t), t) \\ &\leq -U(x, v_1(x,t), t),\end{aligned} \tag{5.34}$$

which obtains

$$\frac{\mathrm{d}V_0(x,t)}{\mathrm{d}t} \leq \frac{\mathrm{d}V_1(x,t)}{\mathrm{d}t}, \quad \forall x \in \mathbb{R}^n. \tag{5.35}$$

Based on Theorem 5.1, we can draw the conclusion that $V_0(x,t) \geq V_1(x,t)$, $\forall x \in \mathbb{R}^n$. Assume that the inequality (5.32) holds for $i = l-1$, $l = 1, 2, \ldots$. Then, we consider the situation for $i = l$. As $v_0(x,t)$ is an admissible control law, according to Theorem 5.2, it can

derived that the iterative control law $v_l(x,t)$ is admissible. According to Lemma 5.1, there exists an iterative value function $V_l(x,t)$, such that

$$U(x,v_l(x,t),t)+\left(\frac{\partial V_l(x,t)}{\partial x}\right)^{\mathsf{T}}F(x,v_l(x,t),t)+\frac{\partial V_l(x,t)}{\partial t}=0. \tag{5.36}$$

According to (5.8), we know that

$$v_{l+1}(x,t)=\arg\min_u\left\{U(x,u,t)+\left(\frac{\partial V_l(x,t)}{\partial x}\right)^{\mathsf{T}}F(x,u,t)\right\}, \tag{5.37}$$

which implies

$$\begin{aligned}\frac{\mathrm{d}V_l(x,t)}{\mathrm{d}t}&=\frac{\partial V_l(x,t)}{\partial t}+\left(\frac{\partial V_l(x,t)}{\partial x}\right)^{\mathsf{T}}F(x,v_{l+1}(x,t),t)\\&\le -U(x,v_{1+1}(x,t),t)\\&=\frac{\partial V_{l+1}(x,t)}{\partial t}+\left(\frac{\partial V_{l+1}(x,t)}{\partial x}\right)^{\mathsf{T}}F(x,v_{l+1}(x,t),t)\\&=\frac{\mathrm{d}V_{l+1}(x,t)}{\mathrm{d}t}.\end{aligned} \tag{5.38}$$

Then, we can derive

$$\frac{\mathrm{d}V_l(x,t)}{\mathrm{d}t}\le\frac{\mathrm{d}V_{l+1}(x,t)}{\mathrm{d}t},\quad\forall x\in\mathbb{R}^n, \tag{5.39}$$

which obtains $V_l(x,t)\ge V_{l+1}(x,t)$. By the mathematical induction, we can get (5.32) for any $i=0,1,\ldots$. From (5.32), it shows that the iterative value function $V_i(x,t)$ is monotonically non-increasing. As $V_i(x,t)$ is positive definite for x, which means that $V_i(x,t)$ is lower bounded for all x, we can define

$$\lim_{i\to\infty}V_i(x,t)=V_\infty(x,t). \tag{5.40}$$

According to (5.8) and (5.9), we have

$$\min_u\left\{U(x,u,t)+\left(\frac{\partial V_\infty(x,t)}{\partial x}\right)^{\mathsf{T}}F(x,u,t)\right\}+\frac{\partial V_\infty(x,t)}{\partial t}=0. \tag{5.41}$$

Then, it can be derived as

$$\min_u \left\{ U(x,u,t) + \frac{\mathrm{d}V_\infty(x,t)}{\mathrm{d}t} \right\} = 0, \tag{5.42}$$

which shows

$$\min_u \left\{ \int_t^\infty U(x,u,\tau)d\tau + \int_t^\infty \frac{\mathrm{d}V_\infty(x,\tau)}{\mathrm{d}\tau}\mathrm{d}\tau \right\} = 0. \tag{5.43}$$

According to Theorem 5.2, it can be derived that $v_\infty(x,t) = \lim_{i\to\infty} v_i(x,t)$ is an admissible control law, which implies

$$\int_t^\infty \frac{\mathrm{d}V_\infty(x,\tau)}{\mathrm{d}\tau}d\tau = V_\infty(x,\infty) - V_\infty(x,t) = -V_\infty(x,t). \tag{5.44}$$

Substituting (5.44) into (5.43), we can get

$$V_\infty(x,t) = \min_{\mu\in\Psi(\Omega)} \left\{ \int_t^\infty U(x(\tau),\mu(x(\tau),\tau),\tau)\mathrm{d}\tau \right\}, \tag{5.45}$$

which shows $V_\infty(x,t) = J^*(x,t)$. The proof is complete. □

5.4 Neural Networks Implementation of the CTTV Policy Iteration

For CTTV policy algorithms of nonlinear system (5.1), $v_i(x,t)$ and $V_i(x,t)$ are both time-varying functions, which are generally very hard to obtain by directly solving (5.7) and (5.9). In this section, we choose neural networks to implement the proposed algorithm, whose aim is to get the approximate laws of optimal solutions for the nonlinear system (5.1). Two neural networks are adopted in the developed CTTV policy iteration algorithms such as critic and action networks. Three-layer back-propagation (BP) algorithm is applied in the neural networks to update the weights. The whole structure diagram of the developed algorithm is shown in Fig. 5.1, where $t_j \geq 0$ is the time parameter which will be explained in the latter content.

The parameter ℓ describes the number of hidden layer neurons, and matrix Y represents the weights between the input and hidden

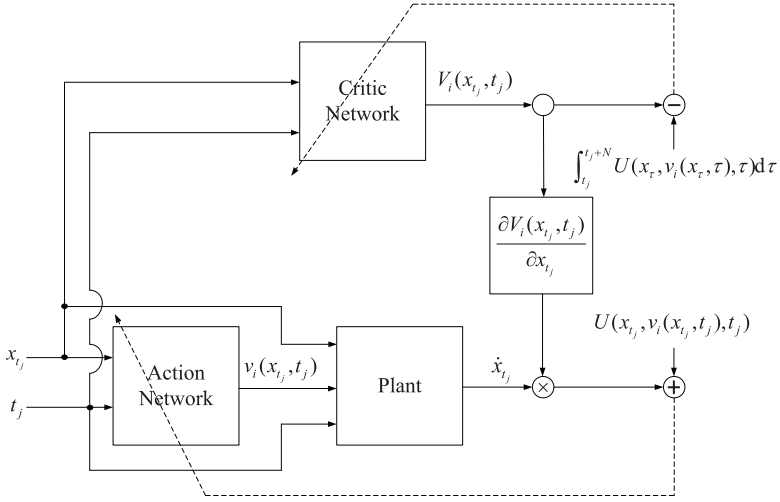

Fig. 5.1. The structure diagram of the algorithm.

layers. Similarly, the weight matrix between the hidden and output layers is reflected by W. Then the output can be represented as

$$\hat{F}(S, Y, W) = W^{\mathsf{T}} \sigma(Y^{\mathsf{T}} S + b), \tag{5.46}$$

where $\sigma(Y^{\mathsf{T}} S) \in \mathbb{R}^{\ell}, [\sigma(z)]_i = \frac{e^{z_i} - e^{-z_i}}{e^{z_i} + e^{-z_i}}, i = 1, \ldots, \ell$, denote the activation functions of neural networks and b is considered as the threshold value.

5.4.1 *The critic network*

In this subsection, the critic network is introduced, which can approximate the performance index function $V_i(x, t)$, $i = 1, 2, \ldots$. It is worth pointing out that the neural network approximation for traditional iterative ADP algorithms of time-invariant systems is invalid for the approximation of time-varying systems. First, the iterative value function in traditional ADP algorithm is a time-invariant function which is only related to $x(t)$, i.e., $V_i(x)$. For the time-varying policy iteration algorithm, the iterative value function is a function of not only $x(t)$ but also t, i.e., $V_i(x, t)$. This may cause the invalidity of the neural network approximation for traditional iterative ADP algorithms of time-invariant systems. Second, for traditional iterative ADP algorithms of time-invariant systems, the iterative control law

$v_i(x)$ and value function $V_i(x)$ can be updated based on the ordinary differential equation (5.10), which is generally easy to solve. For the time-varying policy iteration algorithm, however, it is necessary to solve the partial differential equation (5.9) to get the iterative control law $v_i(x,t)$ and iterative value function $V_i(x,t)$, which is very difficult. This implies that the target function $V_i(x,t)$ cannot be solved, even though the iterative control law can be derived. Thus, in this subsection, neural network approximation must be improved.

If the iterative control law $v_i(x(t),t)$, $i=0,1,\ldots$ is admissible, the iterative value function $V_i(x,t)$ will converge to zero as $t\to\infty$, which has been proven in this paper. So, the iterative value function $V_i(x,t)$ for $0\le t<\infty$ can be approximated by the function $V_i(x,t)$ for $0\le t\le T_f$, where T_f is a large positive number and we regard $V_i(x,t)=0$ for all $t>T_f$.

Let $\mathcal{M}$ be a positive integer. Let t_j, $j=0,1,\ldots,\mathcal{M}$, be time instants, which satisfy $0\le t_j\le T_f$. For an arbitrary time instant $t_j\ge 0$, it is declared that the input of the critic network includes $x(t_j)$ and t_j. Letting $X(t_j)=[x^{\mathsf{T}}(t_j),t_j]^{\mathsf{T}}$, $j=0,1,\ldots,\mathcal{M}$, where $x(t_j)\in\mathbb{R}^n$ and $0\le t_j\le T_f$, for $l=0,1,\ldots$, the output of the critic network is given by the following equations

$$\hat{V}_i^l(x(t_j),t_j)=W_{ci}^{l\mathsf{T}}\sigma(Y_{ci}^{\mathsf{T}}X(t_j)+b_{ci}). \tag{5.47}$$

The target functions are $V_i(x(t_j),t_j)$, $0\le t_j\le T_f$, which are expressed in (5.7) and (5.9). However, it is difficult to solve $V_i(x(t_j),t_j)$ directly using (5.7) and (5.9). Hence, numerical methods are employed.

According to (5.7) and (5.9), the iterative value function $V_i(x(t_j),t_j)$ satisfies the following conditions for any $i=0,1,\ldots,$

$$V_i(x(t_j),t_j)=\int_{t_j}^{\infty}U(x(\tau),v_i(x(\tau),\tau),\tau)\mathrm{d}\tau. \tag{5.48}$$

Because of the admissibility of control input $v_i(x(t),t)$, $U(x(t),v_i(x(t),t),t)$ converges to zero as $t\to\infty$. Choosing a large time terminal $N>0$, for any t_j, we regard $U(x(t),v_i(x(t),t),t)=0$ for all $t>t_j+N$. Therefore, $V_i(x(t_j),t_j)$ can be approximated by

$$V_i(x(t_j),t_j)\doteq\int_{t_j}^{t_j+N}U(x(\tau),v_i(x(\tau),\tau),\tau)\mathrm{d}\tau. \tag{5.49}$$

Then, we define the error function in critic network as

$$e_{ci}^{l}(t_j) = \hat{V}_i^{l}(x(t_j), t_j) - V_i(x(t_j), t_j). \tag{5.50}$$

The objective function is given in the following, which should be minimized.

$$E_{ci}^{l}(t_j) = \frac{1}{2}(e_{ci}^{l}(t_j))^2. \tag{5.51}$$

The updated rules of the weights in critic network are given as the following gradient-based form.

$$\begin{aligned} W_{ci}^{l+1} &= W_{ci}^{l} + \Delta W_{ci}^{l} \\ &= W_{ci}^{l} - \alpha_c \left[\frac{\partial E_{ci}^{l}(t_j)}{\partial \hat{V}_i^{l}(x(t_j), t_j)} \frac{\partial \hat{V}_i^{l}(x(t_j), t_j)}{\partial W_{ci}^{l}} \right] \\ &= W_{ci}^{l} - \alpha_c e_{ci}^{l}(t_j)\sigma(Z_c(t_j)), \end{aligned} \tag{5.52}$$

where $\alpha_c > 0$ is regarded as the learning rate. The same training method is also used to update the weights Y_{ci} and b_{ci} and omitted here. If the desired goal of training precision can be achieved for all $X(t_j)$, $j = 0, 1, \ldots, \mathcal{M}$, then we say that critic networks can approximate $V_i(x(t), t)$ effectively.

Remark 5.2. For $0 \leq t_j \leq T_f$ and $j = 0, 1, \ldots, \mathcal{M}$, we arbitrarily choose the time t_j and the state $x(t_j)$ to construct the vector $X(t_j)$. For any $X(t_j)$, we can use (5.49) to get the desired iterative value function approximately in the critic network. Choosing different t_j and $x(t_j)$, $j = 0, 1, \ldots, \mathcal{M}$, we can obtain an array of $X(t_j)$. Given the array of $X(t_j)$, the array of the target values of $V_i(x(t_j), t_j)$ can be obtained and the critic network can be trained. In this situation, it is unnecessary to update the iterative value function $V_i(x, t)$ by solving the partial differential equation (5.7) or (5.9). This is a merit of the critic approximation. Moreover, the input of the critic neural network for the CTTV policy iteration algorithm is $X(t_j) = [x^{\mathsf{T}}(t_j), t_j]^{\mathsf{T}}$ instead of $x(t_j)$ in the traditional policy iteration algorithms. The present CTTV policy iteration algorithm requires to collect the data both in the state space and time space, which implies that the CTTV policy iteration algorithm needs more data than the one in traditional policy iteration algorithms.

5.4.2 *The action network*

The action network is adopted to obtain the approximation of the iterative control law $v_i(x,t)$. For $0 \le t_j \le T_f$, the input of the action network is expressed as $X(t_j) = [x^{\mathsf{T}}(t_j), t_j]^{\mathsf{T}}$, which includes the state and the time instant. For $l = 0, 1, \ldots$, the action network outputs the following formulation.

$$\hat{v}_i^l(x(t_j), t_j) = W_{ai}^{l\mathsf{T}} \sigma(Y_{ai}^{\mathsf{T}} X(t_j) + b_{ai}). \tag{5.53}$$

For $i = 1, 2, \ldots$ and $0 \le t_j \le T_f$, the desired output of the action network is introduced as

$$v_i(x(t_j), t_j) = \arg\min_{u(t_j)} \left\{ U(x(t_j), u(t_j), t_j) + \left(\frac{\partial \hat{V}_{i-1}(x(t_j), t_j)}{\partial x(t_j)} \right)^{\mathsf{T}} F(x(t_j), u(t_j), t_j) \right\}, \tag{5.54}$$

where $\hat{V}_{i-1}(x(t_j), t_j)$ is obtained by the critic network.

Choosing different t_j and x_{t_j} to construct an array of $X(t_j)$, $j = 0, 1, \ldots, \mathcal{M}$, we can obtain an array of $v_i(x_{t_j}, t_j)$. For any $0 \le t_j \le T_f$, the errors of the output in the action neural networks are defined as

$$e_{ai}^l(t_j) = \hat{v}_i^l(x(t_j), t_j) - v_i(x(t_j), t_j). \tag{5.55}$$

The updated laws of weights in neural networks are designed to achieve the minimization of the following error performance

$$E_{ai}^l(t_j) = \frac{1}{2} (e_{ai}^l(t_j))^{\mathsf{T}} (e_{ai}^l(t_j)). \tag{5.56}$$

The updated rules of weights in the action network can be expressed as the following gradient-based form

$$\begin{aligned} W_{ai}^{l+1} &= W_{ai}^l + \Delta W_{ai}^l \\ &= W_{ai}^l - \beta_a \left[\frac{\partial E_{ai}^l(t_j)}{\partial e_{ai}^l(t_j)} \frac{\partial e_{ai}^l(t_j)}{\partial \hat{v}_i^l(x(t_j), t_j)} \frac{\partial \hat{v}_i^l(x(t_j), t_j)}{\partial W_{ai}^l} \right] \\ &= W_{ai}^l - \beta_a \sigma(Z_a(t_j)) (e_{ai}^l(t_j))^{\mathsf{T}}, \end{aligned} \tag{5.57}$$

where $\beta_a > 0$ represents the learning rate. The same training method is also used to update the weights Y_{ai} and b_{ai} and omitted here. When the desired goal of training precision is achieved, the conclusion can be drawn that $v_i(x,t)$ can be approximated by the action network.

5.4.3 *Training phase of the CTTV policy iteration algorithm*

The training of the CTTV policy iteration algorithm by neural networks is summarized in Algorithm 5.1.

Algorithm 5.1 The CTTV policy iteration algorithm

Initialization:

Let $\mathcal{M}$ be a large positive integer.
Let T_f be a large positive number.
Choose randomly an array of $X(t_j)$, $0 \leq t_j \leq T_f$ and $j = 0, 1, \ldots, \mathcal{M}$, where $X(t_j) = [x^{\mathsf{T}}(t_j), t_j]^{\mathsf{T}}$, $x(t_j) \in \mathbb{R}^n$;
Choose a neural network training precision ε_n;
Define the computation precision ε;
Give the initial control law $v_0(s,t)$ which is admissible for the system;

Iteration:

1: Initialize the iteration index as $i = 0$;
2: For $X(t_j)$, $0 \leq t_j \leq T_f$ and $j = 0, 1, \ldots, \mathcal{M}$, compute the target iterative value function $V_i(x(t_j), t_j)$ by (5.49).
3: Build the critic network by (5.47) and train the critic network according to (5.50)–(5.52) until the training precision ε_n is achieved.
4: If $|V_i(x(t_j), t_j) - V_{i-1}(x(t_j), t_j)| \leq \varepsilon$ for all $X(t_j)$, $0 \leq t_j \leq T_f$ and $j = 0, 1, \ldots, \mathcal{M}$, where $V_{i-1}(\cdot) \equiv 0$ for $i = 0$, then goto Step 8. Otherwise, goto next step.
5: For $X(t_j)$, $0 \leq t_j \leq T_f$ and $j = 0, 1, \ldots, \mathcal{M}$, compute the iterative control law $v_i(x,t)$ by (5.54).
6: Build the action network by (5.53) and train the action network according to (5.55)–(5.57) until the training precision ε_n is achieved.
7: Let $i = i + 1$ and goto Step 2.
8: **return** $v_i(x,t)$ and $V_i(x,t)$.

Remark 5.3. From Algorithm 5.1, we choose an array of $X(t_j)$, $0 \leq t_j \leq T_f$ and $j = 0, 1, \ldots, \mathcal{M}$ to train the weights in the critic and action networks. According to the approximation by (5.49), it

is not necessary to derive the iterative value function $V_i(x(k), k)$ by solving the partial differential equation (5.9). It can be considered as the main advantage of the time-varying policy iteration algorithm. Moreover, for any $i = 0, 1, \ldots$, it requires to search both the state space and all possible time instants to derive the approximation of the time-varying function $v_i(x, t)$ and $V_i(x, t)$. Thus, the computation quantity for the time-varying policy iteration algorithm is obviously higher than one for the traditional time-invariant policy iteration algorithms, which is regarded as the disadvantage for the time-varying policy iteration algorithm. Furthermore, for large T_f and $\mathcal{M}$, it is recommended normalizing the time parameter t before using it to train the neural networks.

5.5 Simulation Example

In this section, a numerical simulation example is conducted to evaluate the performance of our developed CTTV policy iteration algorithm. The dynamics of the system are formalized as

$$\dot{x} = A(x, t)x + Bu, \tag{5.58}$$

where $A(x,t) = \begin{pmatrix} \cos(0.5tx_1) & \sin(0.5x_1) \\ \cos(0.4x_2^2) & \sin(0.4tx_2) \end{pmatrix}$ and $B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Let $x(0) = [1, 1]^\mathsf{T}$ be chosen as the initial system states. The performance index function is defined as

$$J(x, t) = \int_t^\infty (x^\mathsf{T}(\tau)Qx(\tau) + u^\mathsf{T}(\tau)Ru(\tau))\mathrm{d}\tau, \tag{5.59}$$

where matrices $Q = 0.1I_1$, $R = 0.1I_2$ and the identity matrices with appropriate dimensions are denoted by I_1 and I_2, respectively.

To implement the developed algorithm, two neural networks with three-layer structures of 3–8–1 and 3–8–2, respectively, are adopted to approximate the critic and action networks, respectively. In addition, BP algorithm is adopted for updating the weights of neural networks. Let $T_f^{(2)} = 5$. For each iteration, we choose $\mathcal{M}_2 = 50000$ data pairs $X(t_p)$, $p = 1, 2, \ldots, \mathcal{M}_2$, to train the two neural networks, where $-1 \leq x_1(t_p) \leq 1$, $-1 \leq x_2(t_p) \leq 1$ and $0 \leq t_p \leq T_f^{(2)}$, $p = 0, 1, \ldots, \mathcal{M}_2$. The initial control law is given by the action network in

(5.53) with the expression $v_0^{(2)}(x,t) = W_{a,\text{initial}}^{(2)}\sigma(Y_{a,\text{initial}}^{(2)}X + b_{a,\text{initial}}^{(2)})$, where the weights are given by

$$Y_{a,\text{initial}}^{(2)} = \begin{bmatrix} 1.2924 & -3.2481 & -0.0378 \\ -0.1564 & -0.3440 & 0.9568 \\ -1.3969 & -2.4661 & 0.0476 \\ -1.1021 & -0.5115 & 0.0068 \\ 0.1254 & 0.3428 & 1.0669 \\ 0.4877 & -0.1813 & 0.0335 \\ 0.5626 & -0.0873 & 0.0087 \\ 0.0615 & 0.6226 & -0.2972 \end{bmatrix},$$

$$W_{a,\text{initial}}^{(2)} = \begin{bmatrix} -0.6351 & -0.0844 & 0.3953 & -1.7069 & -0.4272 & -9.4428 & 9.1576 & -3.5649 \\ 0.1328 & 2.1255 & -0.0034 & -0.6295 & -1.8743 & 1.2545 & -2.0151 & 0.8724 \end{bmatrix},$$

and

$$b_{a,\text{initial}}^{(2)} = [-2.5873\ 0.0756\ 0.4082\ 0.4899\ -0.2749\ 0.4218\ 0.7759\ 0.7830]^{\mathsf{T}}.$$

In each iteration, the training error 10^{-5} and learning rate $\alpha_c = \beta_a = 0.01$ are chosen for the training of neural networks. The algorithm is implemented for $i = 50$ iterations to approximate the optimal solutions. The trajectory of $V_i(x,t)$ at the initial state $x = x_0$ with 50 iterations is shown in Fig. 5.2, which can prove the characteristics of monotone and convergence in the proposed policy iteration algorithm. The varying curve of system states and control laws are given in Figs. 5.3 and 5.4, respectively. After $i = 50$ iterations, the optimal sequences of control laws can be derived, which similarly the optimal value function is also achieved. It can be seen that system (5.58) can achieve the stabilization by the presented algorithm, and the validity of the proposed methods is also proven. The optimum of the system states and control laws can be illustrated in Fig. 5.5(a) and (b), respectively.

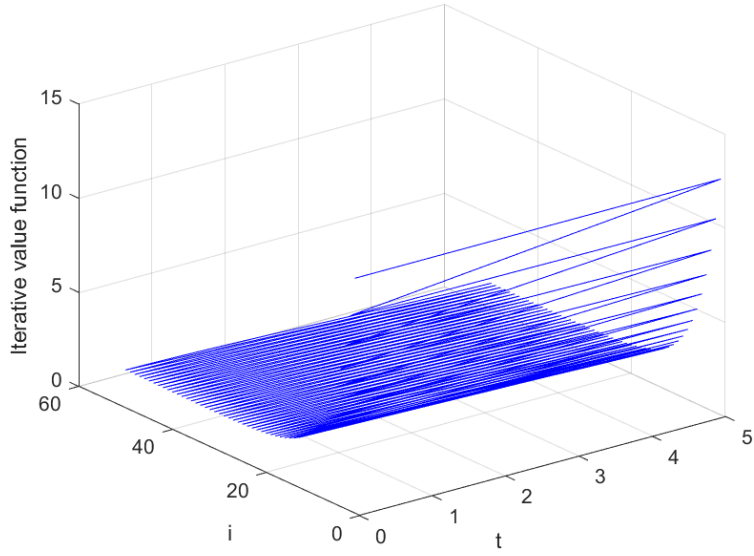

Fig. 5.2. Iterative value function.

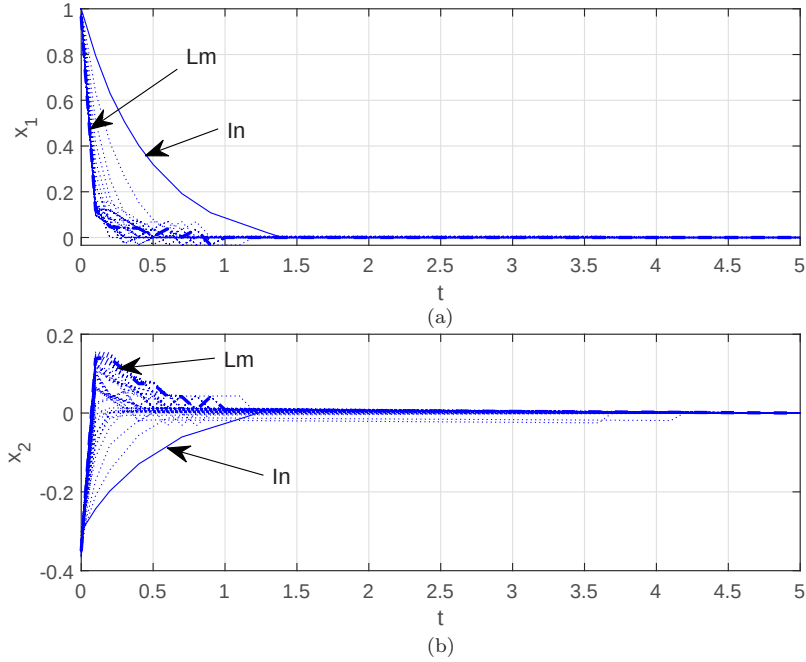

Fig. 5.3. State trajectories.

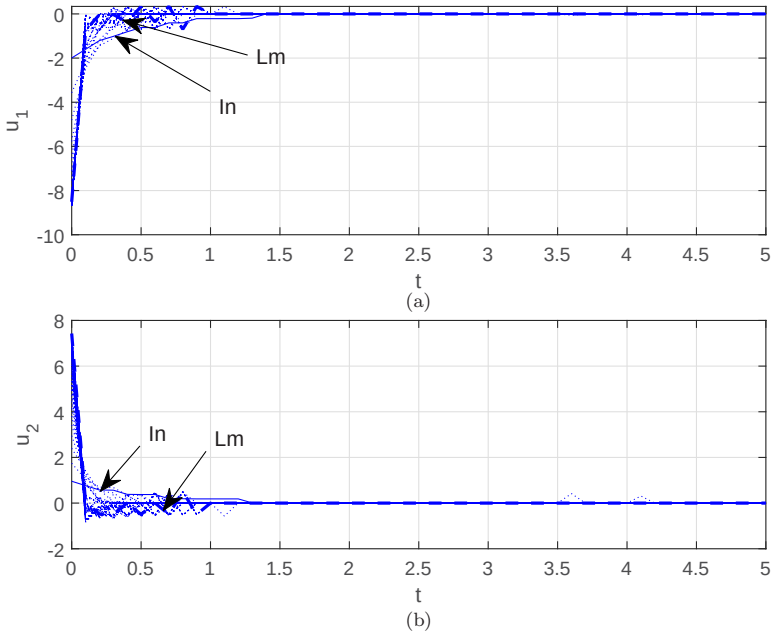

Fig. 5.4. Control trajectories.

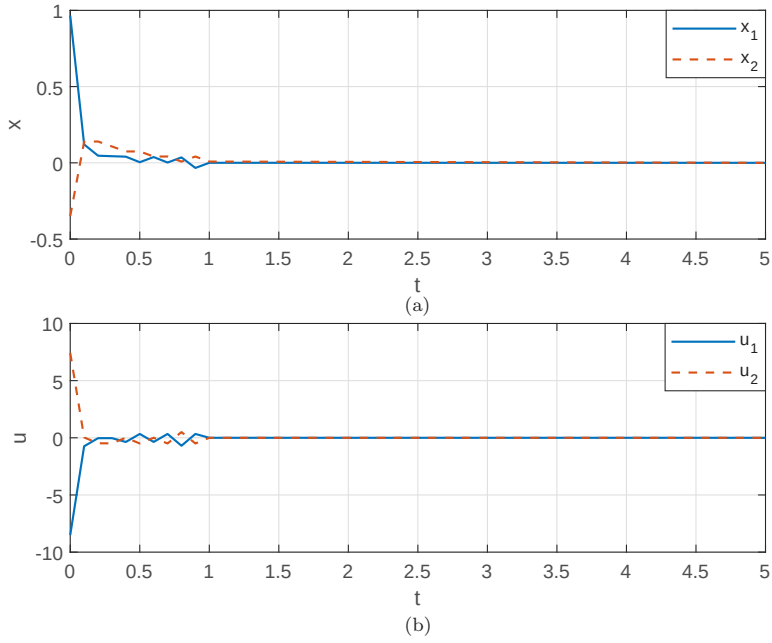

Fig. 5.5. Optimal state and control trajectories.

5.6 Conclusion

To deal with the optimal control problems in continuous-time time-varying nonlinear systems, a new CTTV policy iteration algorithm is developed in this chapter. First, the detailed procedure of the developed algorithm has been introduced. Second, the monotonicity, convergence, and optimality of the iterative value function have been analyzed. Furthermore, the admissibility of the iterative control laws can be guaranteed by the proposed methods. Two neural networks are employed to achieve the approximation of the iterative value function and control law to implement the action and critic networks in the developed algorithm. Via the approximation of the neural networks, it is unnecessary to calculate the iterative value function by solving the partial differential equation (5.9). Finally, a simulation experiment is conducted to illustrate the validity of the present method.

References

[1] Murray, J., Cox, C., Lendaris, G. and Saeks, R. (2002). Adaptive dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews* 32(2), 140–153.

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

[3] Bertsekas, D. (2017). Value and policy iterations in optimal control and adaptive dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems* 28(3), 500–509.

[4] Zhu, Y., Zhao, D., Yang, X. and Zhang, Q. (2018). Policy iteration for H_∞ optimal control of polynomial nonlinear systems via sum of squares programming. *IEEE Transactions on Cybernetics* 48(2), 500–509.

[5] Song, R., Lewis, F. and Wei, Q. (2017). Off-policy integral reinforcement learning method to solve nonlinear continuous-time multiplayer nonzero-sum games. *IEEE Transactions on Neural Networks and Learning Systems* 28(3), 704–713.

[6] Zhang, H., Jiang, H., Luo, C. and Xiao, G. (2017). Discrete-time nonzero-sum games for multiplayer using policy-iteration-based adaptive dynamic programming algorithms. *IEEE Transactions on Cybernetics* 47(10), 3331–3340.

This page intentionally left blank

Chapter 6

Adaptive Dynamic Programming for Discrete-Time Zero-Sum Games

6.1 Introduction

A large class of real systems are controlled by more than one controller or decision maker with each using an individual strategy. These controllers often operate in a group with a general performance index function as a game [1–5]. Two-player zero-sum games, capturing two players' behaviors in which the success of a player in selecting strategies depends strictly on the choices of the other player [6], have been widely applied to decision making problems. In these situations, many control schemes are presented in order to reach some form of optimality. Traditional approaches to deal with zero-sum games are to find out the optimal solution or the saddle-point equilibrium of the games. So many interests are developed to discuss the existence criteria of the saddle-point equilibrium of the zero-sum games. In real-world applications, however, the existence criteria of the saddle-point equilibrium for zero-sum games are so difficult to satisfy that many applications of the zero-sum games are limited to linear systems. For many zero-sum games of nonlinear systems, it is generally assumed that the saddle-point equilibrium exists, which is guaranteed by the assumption of the L_2 gain, and then the optimal control of the zero-sum games for the nonlinear systems is obtained. Unfortunately, for real-world zero-sum games, especially for nonlinear systems, the L_2 gain cannot generally be guaranteed,

which means the existence assumption of the saddle-point equilibrium for the zero-sum games cannot be realized. Thus, traditional optimal control for zero-sum games of nonlinear systems is actually difficult to apply. Therefore, a new method is necessary to obtain the saddle-point equilibrium of the zero-sum game without the complex existence criteria.

In this chapter, a new iterative zero-sum ADP algorithm is developed to solve infinite horizon optimal control problems for discrete-time two-player zero-sum games of nonlinear systems. Initialized by arbitrary positive semi-definite functions for the upper and lower iterations, the upper and lower iterative value functions using the iterative zero-sum ADP algorithm can reach the upper and lower optimums of the zero-sum games, which satisfy the upper and lower Isaacs equations, respectively. A novel convergence analysis method is developed to show that the upper and lower iterative value functions converge to the upper and lower optimums, respectively. Optimality of the iterative zero-sum ADP algorithm will be presented. If the saddle-point equilibrium of the zero-sum game exists, we emphasize that both the upper and lower iterative value functions will converge to the optimal solution of the zero-sum game, where the existence criteria of the saddle-point equilibrium in the traditional zero-sum ADP algorithms are not required. Monotonicity of the upper and lower iterative value functions by the iterative zero-sum ADP algorithm is also presented. It is shown that under some mild constraints of the initial functions, the upper and lower iterative value functions can be monotonically non-increasing, monotonically non-decreasing, non-monotonous and converge to their optimums. If the saddle-point equilibrium does not exist, the upper and lower optimal performance index function are obtained, respectively, where it is proven that the converged upper and lower performance index functions cannot be equivalent. Finally, a simulation example is given to illustrate the performance of the present method.

6.2 Problem Formulations

In this chapter, we will study the following discrete-time nonlinear system

$$x_{k+1} = F(x_k, u_k, w_k), \quad k = 0, 1, 2, \ldots, \tag{6.1}$$

where $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^m$ and $w_k \in \mathbb{R}^l$ are the control vectors of Players I and II, respectively. Let x_0 be the initial state and let $F(x_k, u_k, w_k)$ be the system function. For convenience of analysis, results of system (6.1) are based on the following assumption.

Assumption 6.1. System (6.1) is controllable on a compact set $\Omega_x \subset \mathbb{R}^n$ containing the origin; $x_k = 0$ is an equilibrium state of system (6.1) under the control $u_k = 0$ and $w_k = 0$, i.e., $F(0, 0, 0) = 0$; the feedback controls $u_k = u(x_k)$ and $w_k = w(x_k)$ satisfy $u_k = u(x_k) = 0$ and $w_k = w(x_k) = 0$, respectively, for $x_k = 0$.

Let $\mathcal{U}$ and $\mathcal{W}$ denote policy spaces of Players I and II, respectively. Let $u \in \mathcal{U}$ and $w \in \mathcal{W}$ be the control laws of Players I and II, respectively. Then, the infinite-horizon performance index function $J : \mathcal{U} \times \mathcal{W} \to \mathbb{R}$ for state x_0 can be defined as

$$J(x_0, u, w) = \sum_{k=0}^{\infty} U(x_k, u_k, w_k), \tag{6.2}$$

where we let the utility function $U(x_k, u_k, w_k)$ be positive definite for x_k and u_k, and negative definite for w_k. The triple $\{J; \mathcal{U}, \mathcal{W}\}$ constitutes the *normal form* of the zero-sum dynamic game (zero-sum game in brief), in the context of which we can introduce the notion of a saddle-point equilibrium.

Definition 6.1. Given a zero-sum dynamic game $\{J; \mathcal{U}, \mathcal{W}\}$ in a normal form, a pair of control laws $(u^*, w^*) \in \mathcal{U} \times \mathcal{W}$ constitutes a saddle-point solution if, for all $(u, w) \in \mathcal{U} \times \mathcal{W}$,

$$J(x_k, u^*, w) \le J^*(x_k) := J(x_k, u^*, w^*) \le J(x_k, u, w^*). \tag{6.3}$$

The quantity $J^*(x_k)$ is the optimal performance index function of the game.

Given a zero-sum game $\{J; \mathcal{U}, \mathcal{W}\}$ in a normal form, we define the upper optimal performance index function as

$$\overline{J}^*(x_k) = \min_{u \in \mathcal{U}} \max_{w \in \mathcal{W}} J(x_k, u, w), \tag{6.4}$$

and the lower optimal performance index function can be defined as

$$\underline{J}^*(x_k) = \max_{w \in \mathcal{W}} \min_{u \in \mathcal{U}} J(x_k, u, w), \tag{6.5}$$

with the obvious inequality

$$\underline{J}^*(x_k) \leq \overline{J}^*(x_k). \tag{6.6}$$

If the upper and lower optimal performance index functions are equal, i.e.,

$$\overline{J}^*(x_k) = \underline{J}^*(x_k) = J^*(x_k), \tag{6.7}$$

then optimal performance index function $J^*(x_k)$ for the zero-sum game is defined.

According to the principle of optimality, the upper optimal performance index function $\overline{J}^*(x_k)$ satisfies the following discrete-time Isaacs equation

$$\overline{J}^*(x_k) = \min_{u_k} \max_{w_k} \{U(x_k, u_k, w_k) + \overline{J}^*(F(x_k, u_k, w_k))\}. \tag{6.8}$$

The lower optimal performance index function $\underline{J}^*(x_k)$ satisfies the following discrete-time Isaacs equation

$$\underline{J}^*(x_k) = \max_{w_k} \min_{u_k} \{U(x_k, u_k, w_k) + \underline{J}^*(F(x_k, u_k, w_k))\}. \tag{6.9}$$

Generally, the existence of the saddle-point equilibrium of the zero-sum game $\{J; \mathcal{U}, \mathcal{W}\}$ is difficult to justify, especially for nonlinear systems. Thus, it is nearly impossible to obtain the saddle-point equilibrium by directly solving the inequality (6.3). In this situation, obtaining the upper and lower optimal performance index functions is a necessary method to achieve the saddle-point equilibrium. Unfortunately, the upper and lower optimal performance index functions $\overline{J}^*(x_k)$ and $\underline{J}^*(x_k)$ are also difficult to obtain, due to the difficulty of solving the discrete-time Isaacs equations (6.8) and (6.9). To overcome this difficulty, a new iterative algorithm based on ADP will be developed.

6.3 Iterative ADP Algorithm for Discrete-Time Zero-Sum Games

In this section, a new iterative zero-sum ADP algorithm is developed to solve the discrete-time zero-sum game for system (6.1). New convergence and monotonicity analysis methods will be established in

this section. Optimality analysis will be presented to show that the upper and lower iterative value functions will converge to the optimum, if the saddle-point equilibrium of the zero-sum game exists, where the existence criteria of the saddle-point equilibrium can effectively be avoided.

6.3.1 *Derivations of the Iterative Zero-Sum ADP Algorithm*

In the developed iterative zero-sum ADP algorithm, the value function and control law are updated at every iteration, with the iteration index i increasing from 0 to infinity. For $x_k \in \mathbb{R}^n$, let the initial function $\overline{\Psi}(x_k) \geq 0$ be an arbitrary positive semi-definite function. Then, let the upper initial value function be expressed as

$$\overline{V}_0(x_k) = \overline{\Psi}(x_k). \tag{6.10}$$

For $i = 0, 1, \ldots$, the iterative control law $\overline{\omega}_i(x_k, u_k)$ for the upper iterative value function can be computed as

$$\begin{aligned}\overline{\omega}_i(x_k, u_k) &= \arg\max_{w_k} \left\{U(x_k, u_k, w_k) + \overline{V}_i(x_{k+1})\right\} \\ &= \arg\max_{w_k} \left\{U(x_k, u_k, w_k) + \overline{V}_i(F(x_k, u_k, w_k))\right\}, \end{aligned} \tag{6.11}$$

where $\overline{V}_0(x_{k+1}) = \overline{\Psi}(x_{k+1})$. Then, the iterative control law $\overline{v}_i(x_k)$ can be obtained by

$$\begin{aligned}\overline{v}_i(x_k) = \arg\min_{u_k} \{&U(x_k, u_k, \overline{\omega}_i(x_k, u_k)) \\ &+ V_i(F(x_k, u_k, \overline{\omega}_i(x_k, u_k)))\}. \end{aligned} \tag{6.12}$$

Letting $\overline{\omega}_i(x_k) = \overline{\omega}_i(x_k, \overline{v}_i(x_k))$, the upper iterative control pair $(\overline{v}_i(x_k), \overline{\omega}_i(x_k))$ can be expressed as

$$(\overline{v}_i(x_k), \overline{\omega}_i(x_k)) = \arg(\min_{u_k} \max_{w_k})\left\{U(x_k, u_k, w_k) + \overline{V}_i(x_{k+1})\right\}. \tag{6.13}$$

The upper iterative value function can be updated as

$$\begin{aligned}\overline{V}_{i+1}(x_k) &= \min_{u_k} \max_{w_k} \{U(x_k, u_k, w_k) + \overline{V}_i(F(x_k, u_k, w_k))\} \\ &= U(x_k, \overline{v}_i(x_k), \overline{\omega}_i(x_k)) + \overline{V}_i(F(x_k, \overline{v}_i(x_k), \overline{\omega}_i(x_k))). \end{aligned} \tag{6.14}$$

For $x_k \in \mathbb{R}^n$, let the initial function $\underline{\Psi}(x_k) \geq 0$ be an arbitrary positive semi-definite function. Let the lower initial value function be expressed as

$$\underline{V}_0(x_k) = \underline{\Psi}(x_k). \tag{6.15}$$

For $i = 0, 1, \ldots,$ the iterative control law $\underline{v}_i(x_k, w_k)$ for lower iterative value function can be computed as

$$\begin{aligned} \underline{v}_i(x_k, w_k) &= \arg\min_{u_k} \{U(x_k, u_k, w_k) + \underline{V}_i(x_{k+1})\} \\ &= \arg\min_{u_k} \{U(x_k, u_k, w_k) + \underline{V}_i(F(x_k, u_k, w_k))\}, \end{aligned} \tag{6.16}$$

where $\underline{V}_0(x_{k+1}) = \underline{\Psi}(x_{k+1})$. Then, the iterative control law $\underline{\omega}_i(x_k)$ can be obtained by

$$\begin{aligned} \underline{\omega}_i(x_k) = \arg\max_{u_k} \{&U(x_k, \underline{v}_i(x_k, w_k), w_k) \\ &+ V_i(F(x_k, \underline{v}_i(x_k, w_k), w_k))\}. \end{aligned} \tag{6.17}$$

Letting $\underline{v}_i(x_k) = \underline{v}_i(x_k, \underline{\omega}_i(x_k))$, the lower iterative control pair $(\underline{v}_i(x_k), \underline{\omega}_i(x_k))$ can be expressed as

$$(\underline{v}_i(x_k), \underline{\omega}_i(x_k)) = \arg(\max_{w_k} \min_{u_k}) \big\{U(x_k, u_k, w_k) + \underline{V}_i(x_{k+1})\big\}. \tag{6.18}$$

The lower iterative value function can be updated as

$$\begin{aligned} \underline{V}_{i+1}(x_k) &= \max_{w_k} \min_{u_k} \{U(x_k, u_k, w_k) + \underline{V}_i(F(x_k, u_k, w_k))\} \\ &= U(x_k, \underline{v}_i(x_k), \underline{\omega}_i(x_k)) + \underline{V}_i(F(x_k, \underline{v}_i(x_k), \underline{\omega}_i(x_k))). \end{aligned} \tag{6.19}$$

From the iterative zero-sum ADP algorithm (6.10)–(6.19), we can see that the upper iterative value function $\overline{V}_i(x_k)$ is used to approximate the upper optimal performance index function $\overline{J}^*(x_k)$ and the lower iterative value function $\underline{V}_i(x_k)$ is used to approximate the lower optimal performance index function $\underline{J}^*(x_k)$. Thus, the iteration (6.10)–(6.14) can be called "upper iteration" of the iterative zero-sum ADP algorithm. The iteration (6.15)–(6.19) can be called "lower iteration" of the iterative zero-sum ADP algorithm. Therefore, when $i \to \infty$, the developed iterative zero-sum ADP algorithm

should be convergent which makes $\overline{V}_i(x_k)$ and $\underline{V}_i(x_k)$ converge to their optimal ones. If the saddle-point equilibrium exists of the zero-sum game, both the upper and lower iterative value functions are expected to converge to the saddle-point equilibrium of the zero-sum game. In the next subsection, we will show such properties of the developed iterative zero-sum ADP algorithm.

6.3.2 *Property analysis*

New convergence analysis methods for the value iteration algorithm are developed in this subsection.

Theorem 6.1. *For the zero-sum game* $\{J;\mathcal{U},\mathcal{W}\}$ *of system* (6.1), *the upper and lower optimal performance index functions* $\overline{J}^*(x_k)$ *and* $\underline{J}^*(x_k)$ *in* (6.4) *and* (6.5) *are positive definite for* x_k.

Proof. According to Assumption 6.1, we have

$$\underline{J}^*(x_k) = \overline{J}^*(x_k) = 0 \tag{6.20}$$

for $x_k = 0$. Letting $w_k \equiv 0$, $k = 0, 1, \ldots$, we can get

$$J(x_k, u, 0) = \sum_{i=k}^{\infty} U(x_i, u_i, 0) > 0 \tag{6.21}$$

for all $x_k \neq 0$ and all $u \in \mathcal{U}$, as the utility function is positive definite for x_k and u_k.

According to (6.4), for all $x_k \neq 0$, the optimal upper performance index function satisfies

$$\begin{aligned} \overline{J}^*(x_k) &= \min_{u\in\mathcal{U}} \max_{w\in\mathcal{W}} J(x_k, u, w) \\ &\geq \min_{u\in\mathcal{U}} J(x_k, u, 0) > 0. \end{aligned} \tag{6.22}$$

According to (6.5), for all $x_k \neq 0$, the optimal lower performance index function satisfies

$$\begin{aligned} \underline{J}^*(x_k) &= \max_{w\in\mathcal{W}} \min_{u\in\mathcal{U}} J(x_k, u, w) \\ &= \max_{w\in\mathcal{W}} J(x_k, u^*, w) \\ &\geq J(x_k, u^*, 0) > 0. \end{aligned} \tag{6.23}$$

The proof is complete. □

Lemma 6.1. *For $i = 0, 1, \ldots,$ let the upper and lower iterative value functions $\overline{V}_i(x_k)$ and $\underline{V}_i(x_k)$ be updated by* (6.14) *and* (6.19)*, respectively, where $\overline{V}_0(x_k)$ and $\underline{V}_0(x_k)$ satisfy* (6.10) *and* (6.15)*. Then, for any $i = 0, 1, \ldots,$ the upper and lower iterative value functions $\overline{V}_{i+1}(x_k)$ and $\underline{V}_{i+1}(x_k)$ are positive definite functions for x_k.*

Proof. The statement can be proven by mathematical induction. We first consider the upper iterative value function. For $i = 0$, according to (6.14), we have

$$\overline{V}_1(x_k) = \min_{u_k} \max_{w_k} \{U(x_k, u_k, w_k) + \overline{V}_0(x_{k+1})\}. \tag{6.24}$$

According to Assumption 6.1, it is easy to know $\overline{V_1}(x_k) = 0$ for $x_k = 0$. For any $x_k \neq 0$, we have

$$\begin{aligned} \overline{V}_1(x_k) &= \min_{u_k} \max_{w_k} \{U(x_k, u_k, w_k) + \overline{V}_0(x_{k+1})\} \\ &\geq \min_{u_k} \{U(x_k, u_k, 0) + \overline{V}_0(F(x_k, u_k, 0))\} > 0. \end{aligned} \tag{6.25}$$

Assume that the statement holds for $i = l - 1$, $l = 1, 2, \ldots,$ i.e., $\overline{V}_{l-1}(x_k) > 0$, $\forall x_k \neq 0$ and $\overline{V}_{l-1}(x_k) = 0$, for $x_k = 0$.

Then for $i = l$, according to Assumption 6.1, we have $\overline{V}_{l+1}(x_k) = 0$ and $x_k = 0$. For all $x_k \neq 0$, we can get

$$\begin{aligned} \overline{V}_{l+1}(x_k) &= \min_{u_k} \max_{w_k} \{U(x_k, u_k, w_k) + \overline{V}_l(x_{k+1})\} \\ &\geq \min_{u_k} \{U(x_k, u_k, 0) + \overline{V}_l(F(x_k, u_k, 0))\} > 0. \end{aligned} \tag{6.26}$$

Thus, we have $\overline{V}_{i+1}(x_k)$ is positive definite for any $i = 0, 1, \ldots$. Next, we consider the lower iterative value function. For $i = 0$, according to (6.19), we have

$$\underline{V}_1(x_k) = \max_{w_k} \min_{u_k} \{U(x_k, u_k, w_k) + \underline{V}_0(x_{k+1})\}. \tag{6.27}$$

According to Assumption 6.1, it is easy to know $\underline{V}_1(x_k) = 0$ for $x_k = 0$. For any $x_k \neq 0$, we have

$$\begin{aligned} \underline{V}_1(x_k) &= \max_{w_k} \{U(x_k, v_1(x_k, w_k), w_k) + \underline{V}_0(F(x_k, v_1(x_k, w_k), w_k))\} \\ &\geq U(x_k, v_1(x_k, 0), 0) + \underline{V}_0(F(x_k, v_1(x_k, 0), 0)) > 0. \end{aligned} \tag{6.28}$$

Assume that the statement holds for $i = l - 1$, $l = 1, 2, \ldots,$ i.e., $\underline{V}_{l-1}(x_k) > 0$ $\forall x_k \neq 0$. Then for $i = l$, according to Assumption 6.1, we have $\underline{V}_{l+1}(x_k) = 0$ and $x_k = 0$.

For all $x_k \neq 0$, we can get

$$\begin{aligned}\underline{V}_{l+1}(x_k) &= \max_{w_k}\left\{U(x_k, v_l(x_k, w_k), w_k) + \underline{V}_0(F(x_k, v_l(x_k, w_k), w_k))\right\} \\ &\geq U(x_k, v_l(x_k, 0), 0) + \underline{V}_l(F(x_k, v_l(x_k, 0), 0)) > 0. \quad (6.29)\end{aligned}$$

Thus, we have $\underline{V}_{i+1}(x_k)$ is positive definite for any $i = 0, 1, \ldots$. The mathematical induction is complete. □

Theorem 6.2. *For $i = 0, 1, \ldots$, let the upper and lower iterative value functions $\overline{V}_i(x_k)$ and $\underline{V}_i(x_k)$ be updated by* (6.14) *and* (6.19), *respectively, where $\overline{V}_0(x_k)$ and $\underline{V}_0(x_k)$ satisfy* (6.10) *and* (6.15), *respectively. If for all $x_k \in \Omega_x$, $\overline{\Psi}(x_k) \geq \underline{\Psi}(x_k)$, then for any $i = 0, 1, \ldots$ we have*

$$\underline{V}_i(x_k) \leq \overline{V}_i(x_k). \quad (6.30)$$

Proof. First, the conclusion obviously holds for $i = 0$. For $i = 1$, we have

$$\begin{aligned}\overline{V}_1(x_k) &= \min_{u_k}\max_{w_k}\{U(x_k, u_k, w_k) + \overline{V}_0(x_{k+1})\} \\ &\geq \max_{w_k}\min_{u_k}\{U(x_k, u_k, w_k) + \overline{V}_0(x_{k+1})\} \\ &\geq \max_{w_k}\min_{u_k}\{U(x_k, u_k, w_k) + \underline{V}_0(x_{k+1})\} \\ &= \underline{V}_1(x_k). \quad (6.31)\end{aligned}$$

The inequality (6.30) holds for $i = 1$. Using the mathematical induction, we can derive that the inequality (6.30) holds for any $i = 0, 1, \ldots$. The proof is complete. □

From Theorem 6.1, we know that the upper and lower optimal performance index functions, i.e., $\overline{J}^*(x_k)$ and $\underline{J}^*(x_k)$, are both positive definite functions for x_k, which satisfy (6.8) and (6.9), respectively. On the other hand, according to Lemma 6.1, we can derive that the upper and lower iterative value functions, i.e., $\overline{V}_{i+1}(x_k)$ and $\underline{V}_{i+1}(x_k)$, $i = 0, 1, \ldots$, in the iterative ADP algorithm (6.10)–(6.19) are also positive definite for x_k.

Theorem 6.3. *For $i = 0, 1, \ldots$, let $\overline{V}_i(x_k)$ and $(\overline{v}_i(x_k), \overline{\omega}_i(x_k))$ be obtained by the upper iteration* (6.10)–(6.14). *Then, if the optimal*

upper performance index function $\overline{J}^*(x_k)$ *can be defined for all* $x_k \in \Omega_x$, *then the upper iterative value function* $\overline{V}_i(x_k)$ *converges to the upper optimal performance index function* $\overline{J}^*(x_k)$ *in* (6.8) *for all* $x_k \in \Omega_x$, *as* $i \to \infty$.

Proof. First, if for all $x_k \in \Omega_x$, we choose the control laws $\mu_k = \mu(x_k)$ and $\chi(x_k) = \chi_k$ that satisfy $U(x_k, \mu_k, \chi_k) = 0$, then we can see that the performance index function

$$J(x_k) = \sum_{k=i}^{\infty} U(x_i, \mu_i, \chi_i) = 0 \tag{6.32}$$

for all $x_k \in \Omega_x$. According to Theorem 6.1, we know that the upper optimal performance index function $\overline{J}^*(x_k)$ is positive definite for x_k. Thus, we know that the control law pair (μ, χ) is not the optimal control law for the upper performance index function. Then, all the control law pairs that make $U(x_k, u_k, w_k) = 0$, $\forall x_k \in \Omega_x$ are not considered to be the optimal ones.

On the other hand, we define $\Omega_\epsilon = \{x_k \,|\, x_k \in \Omega_x, \|x_k\| < \epsilon\}$. According to Theorem 6.1 and Lemma 6.1, we have $\overline{J}^*(x_k) = \overline{V}_i(x_k) = 0$ for any $i = 0, 1, \ldots$, as $\epsilon \to 0$. Hence, the conclusion is obvious for $x_k = 0$.

Next, for any $\epsilon > 0$, and $x_k \in \Omega_x \backslash \Omega_\epsilon$, if $U(x_k, u_k, w_k) \neq 0$, then there exist positive constants α and β, which satisfy

$$\alpha \overline{J}^*(x_k) \leq \overline{\Psi}(x_k) \leq \beta \overline{J}^*(x_k), \tag{6.33}$$

where $0 \leq \alpha \leq 1 \leq \beta < \infty$. If the utility function $U(x_k, u_k, w_k) > 0$, then there exists a $\gamma > 0$ that satisfies

$$\overline{J}^*(F(x_k, u_k, w_k)) \leq \gamma U(x_k, u_k, w_k). \tag{6.34}$$

For $i = 0, 1, \ldots$, we will prove the following inequality

$$\left(1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^i}\right) \overline{J}^*(x_k) \leq \overline{V}_i(x_k) \leq \left(1 + \frac{\beta - 1}{(1 + \gamma^{-1})^i}\right) \overline{J}^*(x_k) \tag{6.35}$$

holds for all $x_k \in \Omega_x \backslash \Omega_\epsilon$.

(1) *Prove the left hand side of the inequality* (6.35).

Mathematical induction is employed to prove the conclusion. According to (6.10) and (6.33), the inequality (6.35) obviously holds for $i = 0$. Now, let $i = 1$. We have

$$\begin{aligned}
\overline{V}_1(x_k) &= \min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)+\overline{V}_0(x_{k+1})\right\}\\
&\geq \min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)+\alpha\overline{J}^*(x_{k+1})\right\}\\
&\geq \min_{u_k}\max_{w_k}\left\{\left(1+\gamma\frac{\alpha-1}{1+\gamma}\right)U(x_k,u_k,w_k)\right.\\
&\quad\left.+\left(\alpha-\frac{\alpha-1}{1+\gamma}\right)\overline{J}^*(x_{k+1})\right\}\\
&= \left(1+\frac{\alpha-1}{(1+\gamma^{-1})}\right)\min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)+\overline{J}^*(x_{k+1})\right\}\\
&= \left(1+\frac{\alpha-1}{(1+\gamma^{-1})}\right)\overline{J}^*(x_k). \qquad (6.36)
\end{aligned}$$

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

$$\begin{aligned}
\overline{V}_l(x_k) &= \min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)+\overline{V}_{l-1}(F(x_k,u_k,w_k))\right\}\\
&\geq \min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)+\left(1+\frac{\alpha-1}{(1+\gamma^{-1})^{l-1}}\right)\right.\\
&\quad\times\overline{J}^*(F(x_k,u_k,w_k))+\frac{\alpha-1}{(1+\gamma)(1+\gamma^{-1})^{l-1}}\\
&\quad\left.\times\left(\gamma U(x_k,u_k,w_k)-\overline{J}^*(F(x_k,u_k,w_k))\right)\right\}\\
&= \left(1+\frac{\alpha-1}{(1+\gamma^{-1})^l}\right)\min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)\right.\\
&\quad\left.+\overline{J}^*(F(x_k,u_k,w_k))\right\}\\
&= \left(1+\frac{\alpha-1}{(1+\gamma^{-1})^l}\right)\overline{J}^*(x_k). \qquad (6.37)
\end{aligned}$$

(2) *Prove the right hand side of the inequality* (6.35).

We also use mathematical induction to prove the conclusion. According to (6.10) and (6.33), the inequality (6.35) obviously holds for $i = 0$. Let $i = 1$. We have

$$\begin{aligned}
\overline{V}_1(x_k) &= \min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)+\overline{V}_0(x_{k+1})\right\}\\
&\le \min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)+\beta\overline{J}^*(x_{k+1})\right\}\\
&\le \min_{u_k}\max_{w_k}\left\{\left(1+\gamma\frac{\beta-1}{1+\gamma}\right)U(x_k,u_k,w_k)\right.\\
&\quad\left.+\left(\beta-\frac{\beta-1}{1+\gamma}\right)\overline{J}^*(x_{k+1})\right\}\\
&= \left(1+\frac{\beta-1}{(1+\gamma^{-1})}\right)\min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)+\overline{J}^*(x_{k+1})\right\}\\
&= \left(1+\frac{\beta-1}{(1+\gamma^{-1})}\right)\overline{J}^*(x_k). \qquad (6.38)
\end{aligned}$$

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

$$\begin{aligned}
\overline{V}_l(x_k) &= \min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)+\overline{V}_{l-1}(x_{k+1})\right\}\\
&\le \min_{u_k}\max_{w_k}\left\{U(x_k,u_k,w_k)+\left(1+\frac{\beta-1}{(1+\gamma^{-1})^{l-1}}\right)\right.\\
&\quad\times\overline{J}^*(F(x_k,u_k,w_k))+\frac{1-\beta}{(1+\gamma)(1+\gamma^{-1})^{l-1}}\\
&\quad\left.\times(\overline{J}^*(F(x_k,u_k,w_k))-\gamma U(x_k,u_k,w_k))\right\}\\
&= \left(1+\frac{\beta-1}{(1+\gamma^{-1})^l}\right)\min_{u_k}\max_{w_k}\{U(x_k,u_k,w_k)+J^*(x_{k+1})\}\\
&= \left(1+\frac{\beta-1}{(1+\gamma^{-1})^l}\right)\overline{J}^*(x_k). \qquad (6.39)
\end{aligned}$$

Now, we consider the situation that the utility function $U(x_k, u_k, w_k) < 0$. If $\overline{J}^*(F(x_k, u_k, w_k)) \leq -U(x_k, u_k, w_k)$, according to (6.8), we can obtain that

$$\begin{aligned}\overline{J}^*(x_k) &= \min_{u_k}\max_{w_k}\{U(x_k, u_k, w_k) + \overline{J}^*(F(x_k, u_k, w_k))\} \\ &\leq \min_{u_k}\max_{w_k}\{U(x_k, u_k, w_k) - U(x_k, u_k, w_k)\} \\ &= 0. \end{aligned} \tag{6.40}$$

According to Theorem 6.1, we know that $\overline{J}^*(x_k)$ is positive definite for x_k, which means that $\overline{J}^*(x_k) > 0$ for all $x_k \in \Omega_x \backslash \Omega_\epsilon$. Thus, we can derive that $\overline{J}^*(F(x_k, u_k, w_k)) > -U(x_k, u_k, w_k)$ for $U(x_k, u_k, w_k) < 0$. In this situation, there exists a constant $\gamma_1 > 1$ that satisfies

$$\overline{J}^*(F(x_k, u_k, w_k)) \geq -\gamma_1 U(x_k, u_k, w_k). \tag{6.41}$$

For $i = 0, 1, \ldots$, we will prove the following inequality

$$\left(1 + \frac{\alpha - 1}{(1 - \gamma_1^{-1})^i}\right)\overline{J}^*(x_k) \leq \overline{V}_i(x_k) \leq \left(1 + \frac{\beta - 1}{(1 - \gamma_1^{-1})^i}\right)\overline{J}^*(x_k) \tag{6.42}$$

holds for all $x_k \in \Omega_x \backslash \Omega_\epsilon$.

(3) *Prove the left hand side of the inequality* (6.42).

Mathematical induction is employed to prove the conclusion. According to (6.10) and (6.33), the inequality (6.42) obviously holds for $i = 0$. Now, let $i = 1$. We have

$$\begin{aligned}\overline{V}_1(x_k) &= \min_{u_k}\max_{w_k}\left\{U(x_k, u_k, w_k) + \overline{V}_0(x_{k+1})\right\} \\ &\geq \min_{u_k}\max_{w_k}\left\{\left(1 + \gamma_1\frac{\alpha - 1}{\gamma_1 - 1}\right)U(x_k, u_k, w_k)\right. \\ &\quad \left. + \left(\alpha + \frac{\alpha - 1}{\gamma_1 - 1}\right)\overline{J}^*(x_{k+1})\right\} \\ &= \left(1 + \frac{\alpha - 1}{(1 - \gamma_1^{-1})}\right)\min_{u_k}\max_{w_k}\left\{U(x_k, u_k, w_k) + \overline{J}^*(x_{k+1})\right\} \\ &= \left(1 + \frac{\alpha - 1}{(1 - \gamma_1^{-1})}\right)\overline{J}^*(x_k). \end{aligned} \tag{6.43}$$

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

$$\begin{aligned}
\overline{V}_l(x_k) =& \min_{u_k} \max_{w_k} \left\{ U(x_k, u_k, w_k) + \overline{V}_{l-1}(F(x_k, u_k, w_k)) \right\} \\
\geq & \min_{u_k} \max_{w_k} \left\{ U(x_k, u_k, w_k) + \left(1 + \frac{\alpha - 1}{(1 - \gamma_1^{-1})^{l-1}} \right) \right. \\
& \times \overline{J}^*(F(x_k, u_k, w_k)) + \frac{\alpha - 1}{(\gamma_1 - 1)(1 - \gamma_1^{-1})^{l-1}} \\
& \left. \times (\gamma_1 U(x_k, u_k, w_k) + \overline{J}^*(F(x_k, u_k, w_k))) \right\} \\
= & \left(1 + \frac{\alpha - 1}{(1 - \gamma_1^{-1})^l} \right) \min_{u_k} \max_{w_k} \left\{ U(x_k, u_k, w_k) \right. \\
& \left. + \overline{J}^*(F(x_k, u_k, w_k)) \right\} \\
= & \left(1 + \frac{\alpha - 1}{(1 - \gamma_1^{-1})^l} \right) \overline{J}^*(x_k). \qquad (6.44)
\end{aligned}$$

(4) *Prove the right hand side of the inequality* (6.42).

Mathematical induction is employed to prove the conclusion. According to (6.10) and (6.33), the inequality (6.42) obviously holds for $i = 0$. Now, let $i = 1$. We have

$$\begin{aligned}
\overline{V}_1(x_k) =& \min_{u_k} \max_{w_k} \left\{ U(x_k, u_k, w_k) + \overline{V}_0(x_{k+1}) \right\} \\
\leq & \min_{u_k} \max_{w_k} \left\{ U(x_k, u_k, w_k) + \beta \overline{J}^*(x_{k+1}) \right\} \\
\leq & \min_{u_k} \max_{w_k} \left\{ \left(1 + \gamma_1 \frac{\beta - 1}{1 - \gamma_1} \right) U(x_k, u_k, w_k) \right. \\
& \left. + \left(\beta - \frac{\beta - 1}{\gamma_1 - 1} \right) \overline{J}^*(x_{k+1}) \right\}
\end{aligned}$$

$$= \left(1 + \frac{\beta - 1}{(1 - \gamma_1^{-1})}\right) \min_{u_k} \max_{w_k} \left\{U(x_k, u_k, w_k) + \overline{J}^*(x_{k+1})\right\}$$

$$= \left(1 + \frac{\beta - 1}{(1 - \gamma_1^{-1})}\right) \overline{J}^*(x_k). \tag{6.45}$$

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

$$\begin{aligned}
\overline{V}_l(x_k) &= \min_{u_k} \max_{w_k} \left\{U(x_k, u_k, w_k) + \overline{V}_{l-1}(x_{k+1})\right\} \\
&\leq \min_{u_k} \max_{w_k} \left\{U(x_k, u_k, w_k) + \left(1 + \frac{\beta - 1}{(1 - \gamma_1^{-1})^{l-1}}\right)\right. \\
&\quad \times \overline{J}^*(F(x_k, u_k, w_k)) + \frac{\beta - 1}{(\gamma_1 - 1)(1 - \gamma_1^{-1})^{l-1}} \\
&\quad \left. \times (\overline{J}^*(F(x_k, u_k, w_k)) + \gamma_1 U(x_k, u_k, w_k))\right\} \\
&= \left(1 + \frac{\beta - 1}{(1 - \gamma_1^{-1})^l}\right) \min_{u_k} \max_{w_k} \left\{U(x_k, u_k, w_k) + J^*(x_{k+1})\right\} \\
&= \left(1 + \frac{\beta - 1}{(1 - \gamma_1^{-1})^l}\right) \overline{J}^*(x_k).
\end{aligned} \tag{6.46}$$

According to (6.35) and (6.42), we can obtain that $\lim_{i \to \infty} \overline{V}_i(x_k) = \overline{J}^*(x_k)$ for all $x_k \in \Omega_x \backslash \Omega_\epsilon$. The proof is complete. □

Theorem 6.4. *For $i = 0, 1, \ldots$, let $\underline{V}_i(x_k)$ and $(\underline{v}_i(x_k), \underline{\omega}_i(x_k))$ be obtained by* (6.15)–(6.19). *Then, if the lower optimal performance index function $\underline{J}^*(x_k)$ can be defined for all $x_k \in \Omega_x$, then the lower iterative value function $\underline{V}_i(x_k)$ converges to the lower optimal performance index function $\underline{J}^*(x_k)$ in* (6.9), *as $i \to \infty$.*

Proof. The theorem can be proven according to the idea of (6.32)–(6.46) and the detail is omitted here. □

Theorem 6.5. *For $i = 0, 1, \ldots$, let $\overline{V}_i(x_k)$ and $(\overline{v}_i(x_k), \overline{\omega}_i(x_k))$ be obtained by* (6.10)–(6.14). *If for all $x_k \in \Omega_x$, the inequality*

$$\overline{V}_1(x_k) \leq \overline{V}_0(x_k) \tag{6.47}$$

holds, then the upper iterative value function $\overline{V}_i(x_k)$ is a monotonically non-increasing sequence for $i = 0, 1, \ldots$, i.e.,

$$\overline{V}_{i+1}(x_k) \leq \overline{V}_i(x_k). \tag{6.48}$$

Proof. We prove this by mathematical induction. First, we let $i = 1$. According to (6.14) and (6.47), for all $x_k \in \Omega_x$, we have

$$\begin{aligned}
\overline{V}_2(x_k) &= \min_{u_k} \max_{w_k} \left\{ U(x_k, u_k, w_k) + \overline{V}_1(x_{k+1}) \right\} \\
&\leq \min_{u_k} \max_{w_k} \left\{ U(x_k, u_k, w_k) + \overline{V}_0(x_{k+1}) \right\} \\
&= \overline{V}_1(x_k).
\end{aligned} \tag{6.49}$$

Thus, the inequality (6.48) holds for $i = 1$. Assume that the conclusion holds for $i = l - 1$, $l = 2, 3, \ldots$. Then for $i = l$, we have

$$\begin{aligned}
\overline{V}_{l+1}(x_k) &= \min_{u_k} \max_{w_k} \left\{ U(x_k, u_k, w_k) + \overline{V}_l(x_{k+1}) \right\} \\
&\leq \min_{u_k} \max_{w_k} \left\{ U(x_k, u_k, w_k) + \overline{V}_{l-1}(x_{k+1}) \right\} \\
&= \overline{V}_l(x_k).
\end{aligned} \tag{6.50}$$

Thus, the inequality (6.48) holds for any $i = 0, 1, \ldots$. The proof is complete. □

Theorem 6.6. *For $i = 0, 1, \ldots$, let $\overline{V}_i(x_k)$ and $(\overline{v}_i(x_k), \overline{\omega}_i(x_k))$ be obtained by* (6.10)–(6.14). *If for all $x_k \in \Omega_x$, the inequality*

$$\overline{V}_1(x_k) \geq \overline{V}_0(x_k) \tag{6.51}$$

holds, then the upper iterative value function $\overline{V}_i(x_k)$ is a monotonically non-decreasing sequence for $i = 0, 1, \ldots$, i.e.,

$$\overline{V}_{i+1}(x_k) \geq \overline{V}_i(x_k). \tag{6.52}$$

Corollary 6.1. *For $i = 0, 1, \ldots,$ let $\underline{V}_i(x_k)$ and $(\underline{v}_i(x_k), \underline{\omega}_i(x_k))$ be obtained by* (6.15)–(6.19)*. If for all $x_k \in \Omega_x$, the inequality*

$$\underline{V}_1(x_k) \leq \underline{V}_0(x_k) \tag{6.53}$$

holds, then the lower iterative value function $\underline{V}_i(x_k)$ is a monotonically non-increasing sequence for $i = 0, 1, \ldots,$ i.e.,

$$\underline{V}_{i+1}(x_k) \leq \underline{V}_i(x_k). \tag{6.54}$$

Corollary 6.2. *For $i = 0, 1, \ldots,$ let $\underline{V}_i(x_k)$ and $(\underline{v}_i(x_k), \underline{\omega}_i(x_k))$ be obtained by* (6.15)–(6.19)*. If for all $x_k \in \Omega_x$, the inequality*

$$\underline{V}_1(x_k) \geq \underline{V}_0(x_k) \tag{6.55}$$

holds, then the lower iterative value function $\underline{V}_i(x_k)$ is a monotonically non-decreasing sequence for $i = 0, 1, \ldots,$ i.e.,

$$\underline{V}_{i+1}(x_k) \geq \underline{V}_i(x_k). \tag{6.56}$$

Theorem 6.7. *For $i = 0, 1, \ldots,$ let $\overline{V}_i(x_k)$ and $(\overline{v}_i(x_k), \overline{\omega}_i(x_k))$ be obtained by* (6.10)–(6.14)*. If for all $x_k \in \Omega_x$, the inequality* (6.47) *holds, then the upper iterative value function $\overline{V}_i(x_k)$ satisfies*

$$\overline{V}_i(x_k) \geq \overline{J}^*(x_k). \tag{6.57}$$

If for all $x_k \in \Omega_x$, the inequality (6.51) *holds, then the upper iterative value function $\overline{V}_i(x_k)$ satisfies*

$$\overline{V}_i(x_k) \leq \overline{J}^*(x_k). \tag{6.58}$$

Proof. According to Theorem 6.5, for any $i = 0, 1, \ldots,$ we have

$$\overline{V}_i(x_k) \geq \overline{V}_{i+1}(x_k) \geq \overline{V}_{i+2}(x_k) \geq \cdots. \tag{6.59}$$

Thus, for any $i = 0, 1, \ldots,$ we can obtain

$$V_i(x_k) \geq \lim_{l \to \infty} V_l(x_k) = J^*(x_k). \tag{6.60}$$

The proof is complete. □

Corollary 6.3. *For $i = 0, 1, \ldots,$ let $\underline{V}_i(x_k)$ and $(\underline{v}_i(x_k), \underline{\omega}_i(x_k))$ be obtained by* (6.16)–(6.19)*. If for all $x_k \in \Omega_x$, the inequality* (6.53) *holds, then for $i = 0, 1, \ldots,$ the lower iterative value function*

satisfies

$$\underline{V}_i(x_k) \geq \underline{J}^*(x_k). \tag{6.61}$$

If for all $x_k \in \Omega_x$, *the inequality* (6.55) *holds, then for* $i = 0, 1, \ldots$, *the lower iterative value function satisfies*

$$\underline{V}_i(x_k) \leq \underline{J}^*(x_k). \tag{6.62}$$

Remark 6.1. According to Theorems 6.3–6.7, the convergence and monotonicity properties of the iterative zero-sum ADP algorithm are analyzed, which guarantee that the upper and lower iterative value functions are convergent to their optimums. If the saddle-point equilibrium of the zero-sum exists, then both the upper and lower iterative value functions are expected to converge to the optimal solution of the zero-sum game. In the following part, the optimality of the iterative zero-sum ADP algorithm will be analyzed.

Theorem 6.8. *For* $i = 0, 1, \ldots$, *let* $\overline{V}_i(x_k)$ *and* $(\overline{v}_i(x_k), \overline{\omega}_i(x_k))$ *be obtained by* (6.10)–(6.14) *and let* $\underline{V}_i(x_k)$ *and* $(\underline{v}_i(x_k), \underline{\omega}_i(x_k))$ *be obtained by* (6.15)–(6.19). *We can derive*:

(i) *If the saddle-point equilibrium of the zero-sum game exists, then the upper and lower iterative value functions will both converge to the saddle-point equilibrium;*
(ii) *If the upper and lower iterative value function converge to the same function, then the converged value function is the saddle-point equilibrium of the zero-sum game.*

Proof. First, if the saddle-point equilibrium of the zero-sum game exists, according to Theorem 6.1, we can derive that the optimal performance index function $J^*(x_k)$ is positive definite for x_k. Second, according to Theorem 6.3, we can derive that the control law pair $(\mu(x_k), \chi(x_k))$, which satisfies $U(x_k, \mu_k, \chi_k) = 0$, $\forall x_k \in \Omega_x$, is not the optimal control law pair. Next, for any $\epsilon > 0$, and $x_k \in \Omega_x \backslash \Omega_\epsilon$, if $U(x_k, u_k, w_k) \neq 0$, then there exist constants α^*, β^*, and γ^*, such that $0 \leq \alpha^* \leq 1 \leq \beta^* < \infty$, which satisfy

$$\alpha^* J^*(x_k) \leq \Psi(x_k) \leq \beta^* J^*(x_k), \tag{6.63}$$

where the optimal performance index function $J^*(x_k)$ satisfies

$$\begin{aligned} J^*(x_k) &= \min_{u_k} \max_{w_k} \{U(x_k, u_k, w_k) + J^*(x_{k+1})\} \\ &= \max_{w_k} \min_{u_k} \{U(x_k, u_k, w_k) + J^*(x_{k+1})\}. \end{aligned} \tag{6.64}$$

According to Theorem 6.3, for $U(x_k, u_k, w_k) > 0$, there exists a $\gamma^* > 0$, that makes

$$\left(1 + \frac{\alpha^* - 1}{(1+\gamma^{*-1})^i}\right) J^*(x_k) \le \overline{V}_i(x_k) \le \left(1 + \frac{\beta^* - 1}{(1+\gamma^{*-1})^i}\right) J^*(x_k) \tag{6.65}$$

hold and there exist $\gamma_1^* > 1$ that make

$$\left(1 + \frac{\alpha^* - 1}{(1-\gamma_1^{*-1})^i}\right) J^*(x_k) \le \overline{V}_i(x_k) \le \left(1 + \frac{\beta^* - 1}{(1-\gamma_1^{*-1})^i}\right) J^*(x_k) \tag{6.66}$$

hold, for $U(x_k, u_k, w_k) < 0$. Thus, for all $x_k \in \Omega_x \backslash \Omega_\epsilon$, we have

$$\lim_{i\to\infty} \overline{V}_i(x_k) = J^*(x_k). \tag{6.67}$$

According to the idea (6.63)–(6.67), we can also obtain that

$$\lim_{i\to\infty} \underline{V}_i(x_k) = J^*(x_k). \tag{6.68}$$

Thus, we have that the upper and lower iterative value functions both converge to the optimal performance index function.

On the other hand, if the upper and lower performance index functions converge to the same function, i.e.,

$$\lim_{i\to\infty} \underline{V}_i(x_k) = \lim_{i\to\infty} \overline{V}_i(x_k) = J^o(x_k). \tag{6.69}$$

Thus, we can get

$$\begin{aligned} J^o(x_k) &= \min_{u_k} \max_{w_k} \{U(x_k, u_k, w_k) + J^o(x_{k+1})\} \\ &= \max_{w_k} \min_{u_k} \{U(x_k, u_k, w_k) + J^o(x_{k+1})\}, \end{aligned} \tag{6.70}$$

which means $J^o(x_k) = J^*(x_k)$, $\forall x_k \in \Omega_x$. The proof is complete. □

Corollary 6.4. *For* $i = 0, 1, \ldots,$ *let* $\overline{V}_i(x_k)$ *and* $(\overline{v}_i(x_k), \overline{\omega}_i(x_k))$ *be obtained by* (6.10)–(6.14) *and let* $\underline{V}_i(x_k)$ *and* $(\underline{v}_i(x_k), \underline{\omega}_i(x_k))$ *be obtained by* (6.15)–(6.19). *We can derive the following.*

(i) *If the saddle-point equilibrium of the zero-sum game does not exist, then the upper and lower iterative value functions cannot converge to the same function;*
(ii) *If the upper and lower iterative value function converge to different functions, then the saddle-point equilibrium of the zero-sum game does not exist.*

Remark 6.2. According to Theorem 6.8, if the saddle-point equilibrium of the zero-sum game exists, it is proven that both the upper and lower iterative value functions converge to the optimal solution of the zero-sum game, where the existence criteria of the saddle-point equilibrium in the traditional zero-sum ADP algorithms are not required. This is a remarkable advantage of the developed algorithm. On the other hand, if the saddle-point equilibrium does not exist, according to Corollary 6.4, we can derive that the upper and lower iterative value function must converge to different functions. Thus, we say that the existence justification of the saddle-point equilibrium for the zero-sum games is not necessary for the developed iterative zero-sum ADP. This is another advantage of the developed algorithm.

6.4 Simulation Example

To evaluate the performance of our iterative zero-sum ADP algorithm for discrete-time zero-sum games, we choose an example with quadratic utility function for numerical experiment. The dynamics of the system is expressed as

$$x_{k+1} = \begin{bmatrix} 1 & 0.25\Delta T \\ \Delta T & 1 \end{bmatrix} x_k + \begin{bmatrix} \Delta T \\ 0 \end{bmatrix} u_k + \begin{bmatrix} \Delta T \\ 0 \end{bmatrix} w_k, \tag{6.71}$$

where the sampling interval $\Delta t = 0.1$s. Let $x_0 = [1, -1]^\mathsf{T}$. Let the performance index function be expressed by (6.2). The utility function is the quadratic form $U_1(x_k, u_k, w_k) = x_k^\mathsf{T} Q_1 x_k + u_k^\mathsf{T} R_1 u_k + w_k^\mathsf{T} S_1 w_k$,

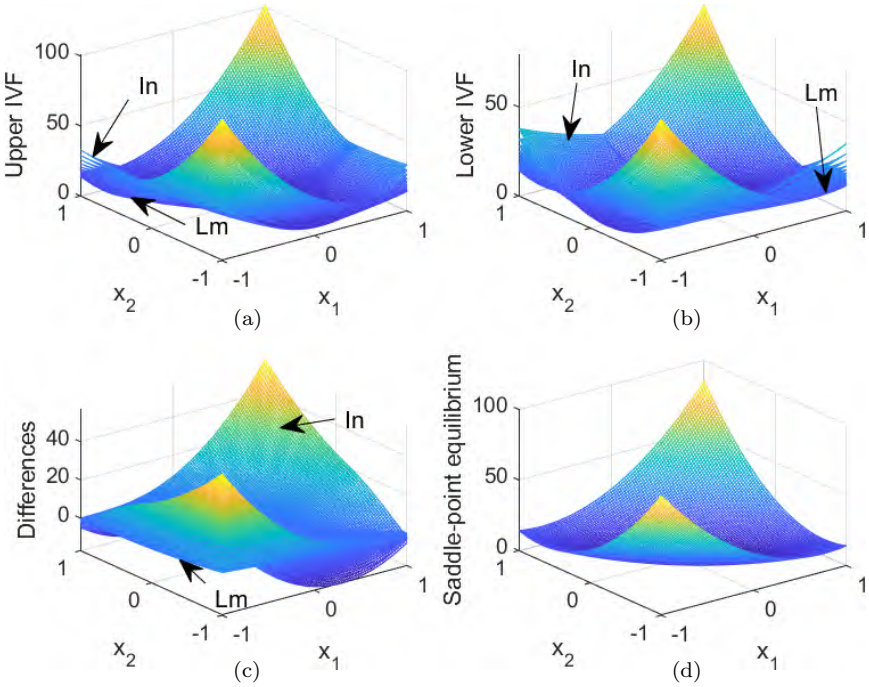

Fig. 6.1. Convergence plots of the iterative value functions. (a) $\overline{V}_i(x_k)$ with $\overline{\Psi}^0(x_k)$. (b) $\underline{V}_i(x_k)$ with $\underline{\Psi}^0(x_k)$. (c) Plots of $\overline{V}_i(x_k) - \underline{V}_i(x_k)$. (d) Saddle-point equilibrium.

where $Q_1 = I_1$, $R_1 = I_2$, $S_1 = -2I_3$ and I_1, I_2, I_3 denote the identity matrices with suitable dimensions. Let the upper initial value function be chosen as $\overline{\Psi}^0(x_k) = x_k^{\mathrm{T}}\overline{P}_0 x_k$, where $\overline{P}_0 = \begin{bmatrix} 6 & 2 \\ 2 & 31 \end{bmatrix}$. Let the lower initial value function be chosen as $\underline{\Psi}^0(x_k) = x_k^{\mathrm{T}}\underline{P}_0 x_k$, where $\underline{P}_0 = \begin{bmatrix} 21 & -5 \\ -5 & 7 \end{bmatrix}$. Implement the iterative zero-sum ADP algorithm for 20 iterations to reach the computation precision $\varepsilon = 0.01$. The convergence plots of the upper and lower iterative value functions, i.e., $\overline{V}_i(x_k)$ and $\underline{V}_i(x_k)$, which are initialized by $\overline{\Psi}^0(x_k)$ and $\underline{\Psi}^0(x_k)$, respectively, are shown in Figs. 6.1(a) and (b), respectively, where "In" denotes "initial iteration", "Lm" denotes "limiting iteration", and "IVF" denotes "iterative value function". The differences between $\overline{V}_i(x_k)$ and $\underline{V}_i(x_k)$ are shown in Fig. 6.1(c), which converges to zero. Thus, we say that the saddle-point equilibrium of the

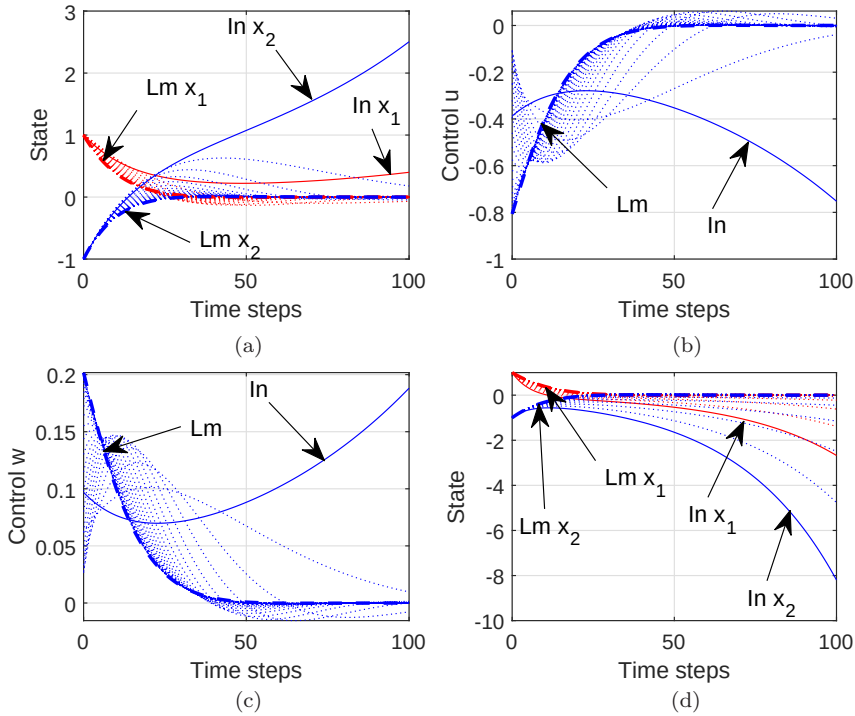

Fig. 6.2. Trajectories of states and controls. (a) States by upper iteration with $\overline{\Psi}^0(x_k)$. (b) Control u by upper iteration with $\overline{\Psi}^0(x_k)$. (c) Control w by upper iteration with $\overline{\Psi}^0(x_k)$. (d) States by the lower iteration with $\underline{\Psi}^0(x_k)$.

zero-sum game exists. Initialized by arbitrary positive semi-definite functions, the upper and lower iterative value iterations can both converge to the saddle-point equilibrium of the zero-sum game. The plot of saddle-point equilibrium of the zero-sum game for system (6.71) is shown in Fig. 6.1(d).

The state and control trajectories of the upper iteration with $\overline{\Psi}^0(x_k)$ are shown in Figs. 6.2(a)–(c), respectively. The state and control trajectories of the lower iteration with $\underline{\Psi}^0(x_k)$ are shown in Fig. 6.2(d) and Figs. 6.3(a)–(b), respectively. Initialized by arbitrary positive semi-definite functions, the states and controls can also converge to their optimums of the zero-sum game. The optimal

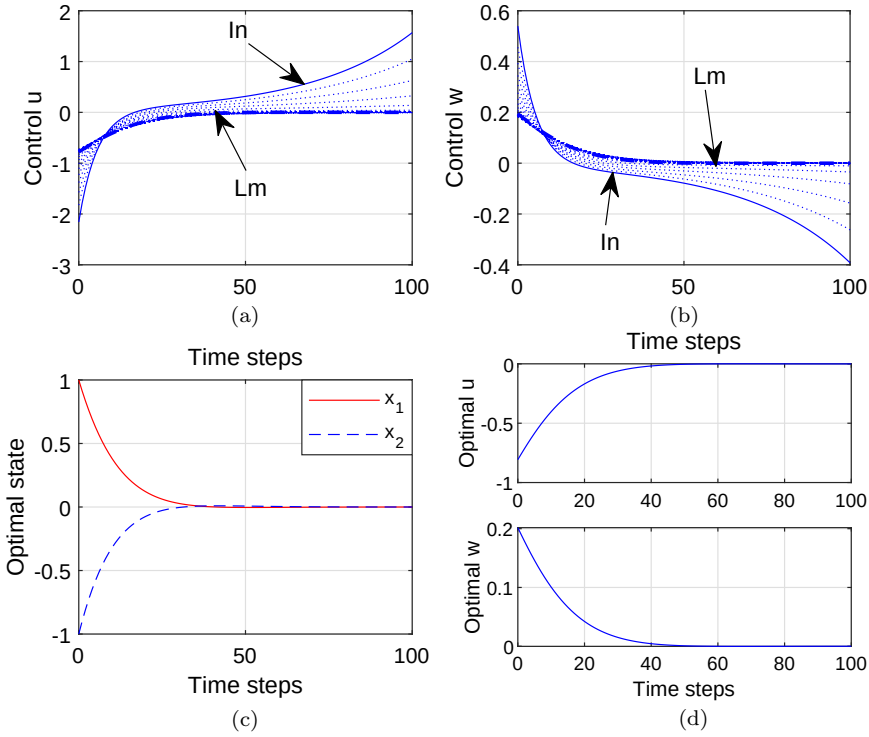

Fig. 6.3. Iterative and optimal trajectories. (a) Control u by the lower iteration with $\underline{\Psi}^0(x_k)$. (b) Control w by the lower iteration with $\underline{\Psi}^0(x_k)$. (c) Optimal states. (d) Optimal controls.

trajectories of the state, controls u and w are shown in Figs. 6.3(c) and (d), respectively.

From Figs. 6.1–6.3, we can see that if the saddle-point equilibrium of the zero-sum game exists, then the upper and lower iterative value function will converge to the saddle-point equilibrium and the corresponding state and controls can also converge to their optimums, where the existence criteria of the saddle-point equilibrium are unnecessary. Thus the effectiveness of the developed algorithm can be justified. According to Lemma 3.1 we know that the optimal performance index function $J^*(x_k)$ satisfies $J^*(x_k) = x_k^{\mathrm{T}} P^* x_k$, where we can derive $P^* = \begin{bmatrix} 20.88 & 14.21 \\ 14.21 & 21.09 \end{bmatrix}$. The optimal control pair

$(u^*(x_k), w^*(x_k)) = (K_1^* x_k, K_2^* x_k)$, where $K_1^* = [-1.85, -1.22]$ and $K_2^* = [0.07, 0.05]$. Hence, the correctness of the developed iterative zero-sum ADP algorithm can be justified.

Now, we choose the utility function as $U_2(x_k, u_k, w_k) = x_k^{\mathrm{T}} Q_2 x_k + u_k^{\mathrm{T}} R_2 u_k + w_k^{\mathrm{T}} S_2 w_k$, where $Q_2 = I_1$, $R_2 = I_2$, $S_2 = -2I_3$. Let the upper initial value function be expressed by $\overline{\Psi}^1(x_k) = x_k^{\mathrm{T}} \overline{P}_1 x_k$, where $\overline{P}_1 = \begin{bmatrix} 19 & -5 \\ -5 & 15 \end{bmatrix}$. Let the lower initial value function be expressed by $\underline{\Psi}^1(x_k) = x_k^{\mathrm{T}} \underline{P}_1 x_k$, where $\underline{P}_1 = 0$.

Implement the iterative zero-sum ADP algorithm for 20 iterations to reach the computation precision $\varepsilon = 0.01$. The convergence plots of the upper and lower iterative value functions, initialized by $\overline{\Psi}^1(x_k)$ and $\underline{\Psi}^1(x_k)$, respectively, are shown in Figs. 6.4(a) and (b), respectively.

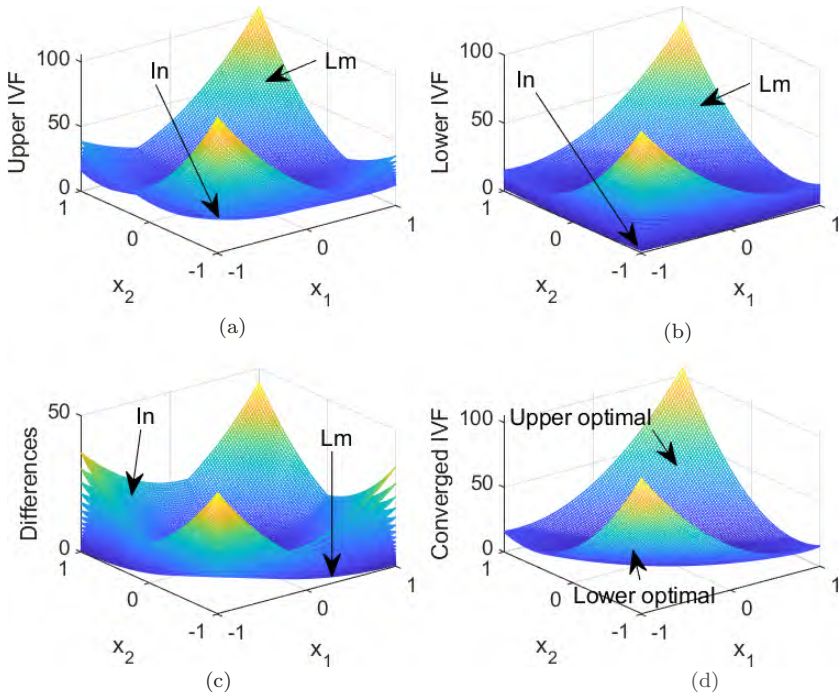

Fig. 6.4. Convergence plots of the iterative value functions. (a) $\overline{V}_i(x_k)$ with $\overline{\Psi}^1(x_k)$. (b) $\underline{V}_i(x_k)$ with $\underline{\Psi}^1(x_k)$. (c) Plots of $\overline{V}_i(x_k) - \underline{V}_i(x_k)$. (d) Upper and lower optimal performance index function.

As $\underline{\Psi}^1(x_k) = 0$, we have $\underline{V}_1(x_k) \leq \underline{V}_0(x_k)$. According to Theorem 6.4 and Corollary 6.1, we know that the lower iterative value function is monotonically non-decreasing and converges to the lower optimum. From Fig. 6.4(b), the monotonicity of the iterative zero-sum ADP algorithm can be justified. On the other hand, as $\overline{\Psi}^1(x_k) \geq \underline{\Psi}^1(x_k)$, from Fig. 6.4(c) we can see that $\overline{V}_i(x_k) \geq \underline{V}_i(x_k)$, $\forall\, i = 0, 1, \ldots$, which justifies the correctness of Theorem 6.2. However, from Fig. 6.4(d), we can see that the saddle-point equilibrium of the zero-sum game does not exist. The corresponding state and control trajectories are shown in Figs. 6.5 and 6.6(a)–(b), respectively. The upper and lower state and control trajectories are shown in Figs. 6.6(c)–(d). We can see that the upper and lower state and control trajectories are different.

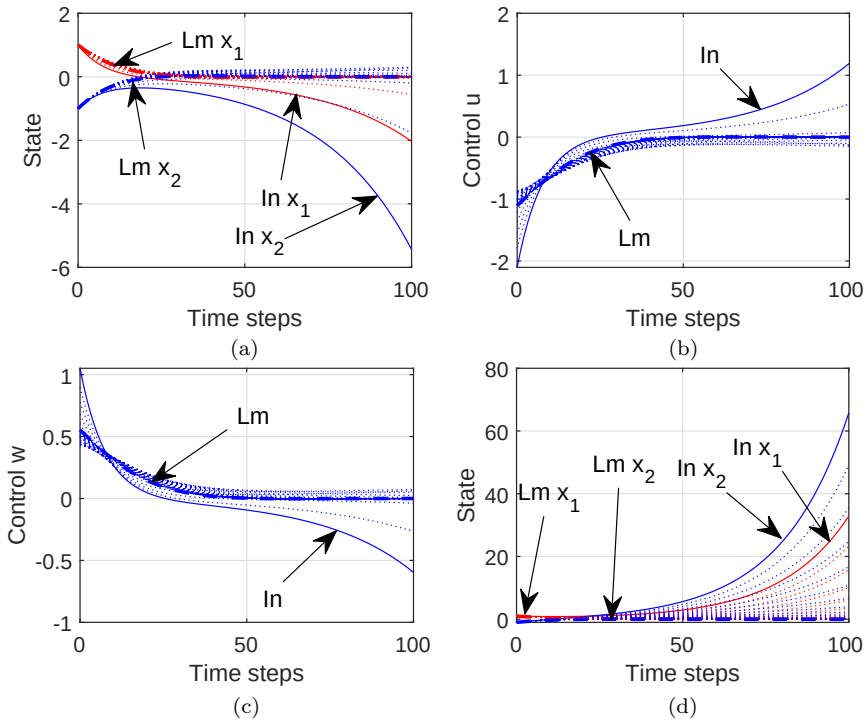

Fig. 6.5. Trajectories of states and controls. (a) States by the upper iteration with $\overline{\Psi}^1(x_k)$. (b) Control u by the upper iteration with $\overline{\Psi}^1(x_k)$. (c) Control w by the upper iteration with $\overline{\Psi}^1(x_k)$. (d) States by the lower iteration with $\underline{\Psi}^1(x_k)$.

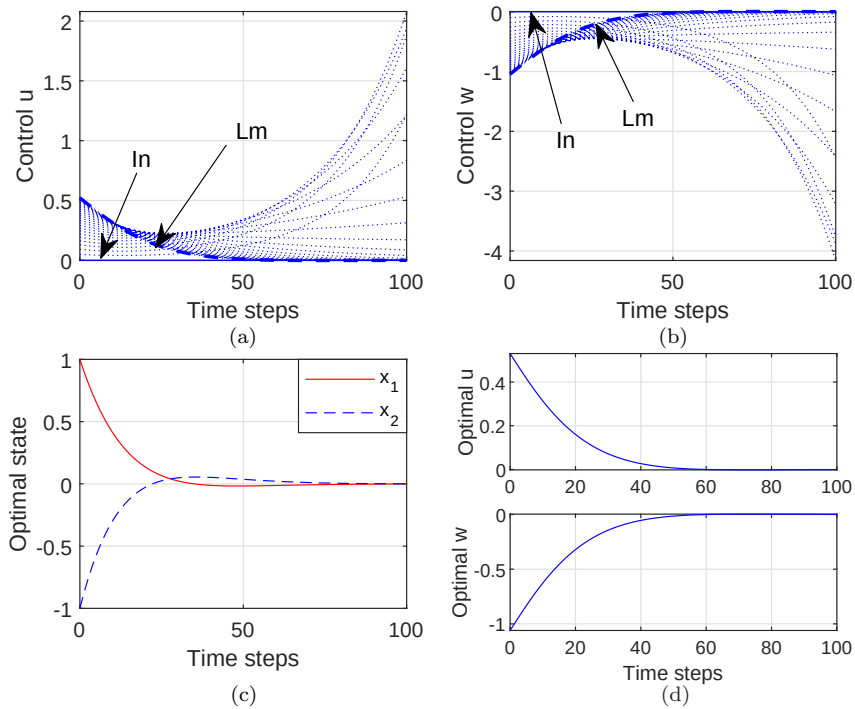

Fig. 6.6. Iterative and converged trajectories. (a) Control u by the lower iteration with $\underline{\Psi}^1(x_k)$. (b) Control w by the lower iteration with $\underline{\Psi}^1(x_k)$. (c) Upper and lower optimal states. (d) Upper and lower optimal controls.

6.5 Conclusion

In this chapter, a new iterative zero-sum ADP algorithm is developed to solve infinite horizon optimal control problems for zero-sum games of discrete-time nonlinear systems. In the developed algorithm, the upper and lower iterations permit arbitrary positive semi-definite functions to initialize the iterations. It is proven that the upper and lower iterative value functions can converge to the upper and lower optimal performance index functions, which satisfy the upper and lower Isaacs equations, respectively. We emphasize that if the saddle-point equilibrium of the zero-sum game exists, it is proven that both the upper and lower iterative value functions converge to the optimal solution of the zero-sum game, where the existence justification of the saddle-point equilibrium is not required. By some

constraints on the initial upper and lower functions, the monotonicity of the upper and lower iterative value functions by the iterative zero-sum ADP algorithm can be guaranteed. If the upper and lower iterative value functions do not converge to the same function, it is proven that the saddle-point equilibrium does not exist. Finally, a simulation example is given to illustrate the performance of the present method.

References

[1] Jamshidi, M. (1982). *Large-Scale Systems-Modeling and Control*. Amsterdam, The Netherlands, North-Holland.

[2] Zhang, H., Qing, C., and Luo, Y. (2014). Neural-network-based constrained optimal control scheme for discrete-time switched nonlinear system using dual heuristic programming. *IEEE Transactions on Automation Science and Engineering* 11(3), 839–849.

[3] Zhang, H., Cui, L., and Luo, Y. (2013). Near-optimal control for nonzero-sum differential games of continuous-time nonlinear systems using single-network ADP. *IEEE Transactions on Cybernetics* 43(1), 206–216.

[4] Chen, W. and Ren, W. (2016). Event-triggered zero-gradient-sum distributed consensus optimization over directed networks. *Automatica* 65, 90–97.

[5] Vamvoudakis, K.G., Lewis, F.L., and Hudas, G.R. (2012). Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality. *Automatica* 48(8), 1598–1611.

[6] Vamvoudakis, K.G. and Lewis, F.L. (2012). Online solution of nonlinear two-player zero-sum games using synchronous policy iteration. *International Journal of Robust Nonlinear Control* 22(13), 1460–1483.

This page intentionally left blank

Chapter 7

Model-Free Adaptive Optimal Control for Unknown Nonlinear Multi-Player Non-Zero-Sum Game

7.1 Introduction

In this chapter, an online adaptive optimal control algorithm based on adaptive dynamic programming is developed to solve the multi-player non-zero-sum game (MP-NZSG) for discrete-time unknown nonlinear systems. First, a model-free coupled globalized dual heuristic dynamic programming (GDHP) structure is designed to solve the MP-NZSG problem, in which there is no model network or identifier. Second, in order to relax the requirement of the systems dynamics, an online adaptive learning algorithm is developed to solve the Hamilton–Jacobi equation using the system states of two adjacent time-steps. Third, a series of critic networks and action networks are used to approximate the value functions and optimal policies for all players, respectively. All the neural network weights are updated online based on real-time system states. Fourth, the uniform ultimate boundedness analysis of the neural network approximation errors is proved based on Lyapunov approach. Finally, simulation results are given to demonstrate the effectiveness of the developed scheme.

7.2 Problem Statement

Consider a class of DT nonlinear systems described by

$$x_{k+1} = f(x_k) + \sum_{j=1}^{N} g_{j,k} u_{j,k}, \quad k = 0, 1, 2, \ldots, \tag{7.1}$$

where $x_k \in \mathbb{R}^n$ is the system state vector with initial state x_0, $u_{j,k} = u_j(x_k) \in \mathbb{R}^{m_j}$ is the control input vector. $f(x_k) \in \mathbb{R}^n$ and $g_{j,k} = g_j(x_k) \in \mathbb{R}^{n \times m_j}$ are system dynamics. N is the number of players.

Assumption 7.1. System (7.1) is controllable on a compact set $\Omega \in \Re^n$ containing the origin. $f + \sum gu$ is Lipschitz continuous for x_k and $u_{j,k}$, $j = 1, 2, \ldots, N$. $x_k = 0$ is an equilibrium state of system (7.1) under the control policies $u_{j,k} = 0$, $j = 1, 2, \ldots, N$.

The performance index function associated with player j is defined by

$$J_j(x_0, u_j, u_{-j}) = \sum_{p=0}^{\infty} \alpha_j^p r_j(x_p, u_{j,p}, u_{-j,p}), \tag{7.2}$$

where

$$\begin{aligned} r_j(x_p, u_{j,p}, u_{-j,p}) = {} & x_p^{\mathsf{T}} Q_j x_p + u_{j,p}^{\mathsf{T}} R_{j,j} u_{j,p} \\ & + \sum_{i=1, i \neq j}^{N} u_{i,p}^{\mathsf{T}} R_{i,j} u_{i,p} \end{aligned} \tag{7.3}$$

is utility function, Q_j, $R_{j,j}$ and $R_{i,j}$ are positive definite symmetric matrices with appropriate dimensions, $u_{-j,p} = \{u_i(x_p) : i = 1, 2, \ldots, N, i \neq j\}$ denotes the control policy set except $u_{j,p}$, α_j is the discount factor of player j. u_j and u_{-j} are the control policies.

Definition 7.1 (Admissible Control). For $j = 1, 2, \ldots, N$, the feedback control policy $u_j(x_k)$ is defined as admissible with respect to (7.2) on a set $\Omega \subset \mathbb{R}^n$, denoted by $u_j(x_k) \in \psi(\Omega)$, if the following conditions hold: 1), $u_j(x_k)$ is continuous on Ω; 2), $u_j(0) = 0$; 3), $u_j(x_k)$ stabilizes (7.1), and (7.2) is finite $\forall x_0 \in \Omega$.

According to Assumption 7.1 and Definition 7.1, the value function of player j can be defined as

$$V_j(x_k) = \sum_{p=k}^{\infty} \alpha_j^{p-k} r_j\left(x_p, u_{j,p}, u_{-j,p}\right), \tag{7.4}$$

where $V_j(x_k)$ represents $V_j(x_k, u_{j,k}, u_{-j,k})$. Then the optimal value function of MP-NZSG is given as

$$V_j^*(x_k) = \min_{u_j} \sum_{p=k}^{\infty} \alpha_j^{p-k} r_j(x_p, u_{j,p}, u_{-j,p}), \tag{7.5}$$

which is the core of this optimal control problem. The following assumption and definition are needed to solve (7.5).

Definition 7.2 (Nash Equilibrium, [1]). In the MP-NZSG problem, the control policy set $\{u_j^*, u_{-j}^*\}$ is said to constitute a Nash equilibrium solution, if the following inequality

$$J_j^* = J_j\left(u_j^*, u_{-j}^*\right) \leq J_j(u_j, u_{-j}^*) \tag{7.6}$$

holds for player j. Then the set $\{J_1^*, J_2^*, \ldots, J_j^*, \ldots, J_N^*\}$ is a Nash equilibrium outcome of the MP-NZSG.

Assumption 7.2. Suppose that in the optimal control problem of MP-NZSG there is only one Nash equilibrium if and only if the Nash condition (7.6) holds.

Review and set forward one step for the definition of the value function (7.5), which can be provided by

$$V_j^*(x_k) = \min_{u_j}\{r_j(x_k, u_{j,k}, u_{-j,k}) + \alpha_j V_j^*(x_{k+1})\}. \tag{7.7}$$

According to the optimal principle, the optimal control policy $u_{j,k}^*$ satisfies the first order necessity condition for the gradient of right hand side of (7.7). Then, $u_{j,k}^*$ can be expressed as

$$u_{j,k}^* = -\frac{\alpha_j}{2} R_{j,j}^{-1} g_{j,k}^{\mathsf{T}} \frac{\partial V_j^*\left(x_{k+1}\right)}{\partial x_{k+1}}. \tag{7.8}$$

Submitting (7.8) and others into (7.7), we can get the DT coupled HJ equation for player j.

$$V_j^*(x_k) = x_k^T Q_j x_k + \frac{\alpha_j^2}{4}\left(\frac{\partial V_j^*(x_{k+1})}{\partial x_{k+1}}\right)^{\mathsf{T}} g_{j,k} R_{j,j}^{-1} g_{j,k}^{\mathsf{T}} \frac{\partial V_j^*(x_{k+1})}{\partial x_{k+1}}$$

$$+ \sum_{i=1, i\neq j}^{N} (u_{i,k}^*)^{\mathsf{T}} R_{i,j} u_{i,k}^* + \alpha_j V_j^*(x_{k+1}). \tag{7.9}$$

Observing (7.9), it is difficult to obtain the optimal value function from the HJ equation (7.9). In the next section, the adaptive dynamic programming is applied to effectively seek the optimal Nash equilibrium solution for this MP-NZSG problem.

7.3 MF-CGDHP Algorithm for Multi-Player Non-Zero-Sum Game

In this section, to solve the optimal control problem of MP-NZSG, an adaptive learning method is developed based on neural networks, which is the MF-CGDHP design. First, GDHP design is expanded to deal with MP-NZSG. Second, a MF-CGDHP algorithm is proposed to obtain the optimal control policies.

7.3.1 *Structural design for MF-CGDHP*

In this subsection, to solve the coupled HJ equation (7.9), a MF-CGDHP design is proposed as shown in Fig. 7.1.

We firstly employ it to the optimal control problem of MP-NZSG. Similarly, both the control policy $\{u_{j,k}, u_{-j,k}\}$ and system state x_k are input into critic network. Then the output of critic network not only estimates the value function $V_j(x_k)$, but also includes the partial derivatives of $V_j(x_k)$, which can be described as

$$\lambda_j(x_k) = \frac{\partial V_j(x_k)}{\partial X_{j,k}}, \tag{7.10}$$

where $X_{j,k} = [x_k^{\mathsf{T}}, u_{1,k}^{\mathsf{T}}, u_{2,k}^{\mathsf{T}}, \ldots, u_{N,k}^{\mathsf{T}}]^{\mathsf{T}}$ is the input of the critic network. This is different from the traditional methods which just contain the system state x_k.

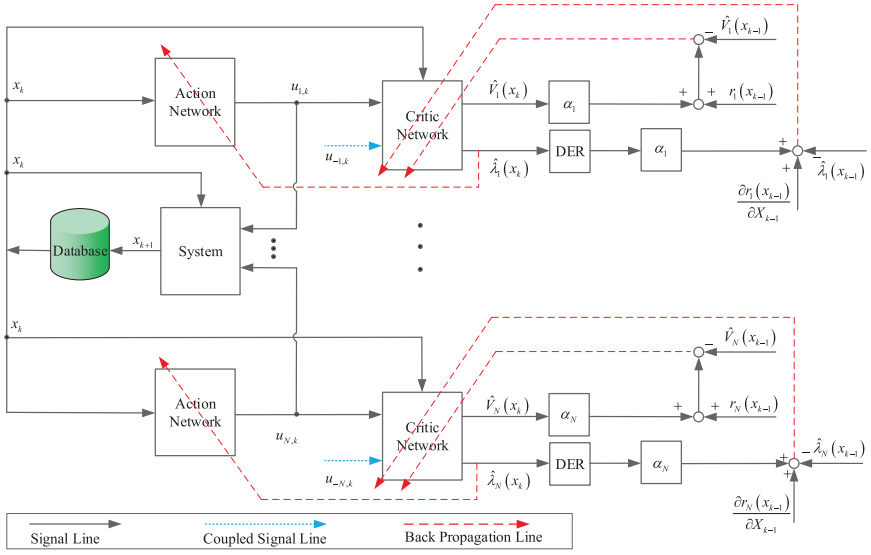

Fig. 7.1. Schematic architecture of the MF-CGDHP design for unknown nonlinear NZSG.

In order to eliminate the requirement for system model information, review and set backward one step for the value function (7.7), we have

$$\alpha_j V_j(x_k) - [V_j(x_{k-1}) - r_j(x_{k-1})] = 0, \tag{7.11}$$

where $r_{j,k-1} = r_j(x_{k-1}, u_{j,k-1}, u_{-j,k-1})$. Next, take the derivative of both sides with respect to $X_{j,k-1}$, one has

$$\alpha_j \frac{\partial V_j(x_k)}{\partial X_{j,k}} \frac{\partial X_{j,k}}{\partial X_{j,k-1}} - \left[\frac{\partial V_j(x_{k-1})}{\partial X_{j,k-1}} - \frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}}\right] = 0, \tag{7.12}$$

where $\partial X_{j,k}/\partial X_{j,k-1}$ is a Jacobian matrix. In Fig. 7.1, it is defined as the DER. According to (7.10), (7.12) can be rewritten as

$$\alpha_j \frac{\partial X_{j,k}}{\partial X_{j,k-1}} \lambda_j(x_k) - \left[\lambda_j(x_{k-1}) - \frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}}\right] = 0. \tag{7.13}$$

From the above analysis, it can be seen that only the system states at time steps $k-1$ and k are needed in equations (7.11) and (7.13), which avoid the system dynamics $f(x_k)$ and $g_{j,k}$, $j = 1, 2, \ldots, N$. In the following statement, we will give a detailed description about the updating rules of weights for critic networks and action networks.

7.3.2 *MF-CGDHP implementation based on neural networks*

In this subsection, we implement the ADP algorithm by neural networks based on the MF-CGDHP structure for MP-NZSG problem. The architecture has been shown in Fig. 7.1, which contains N critic networks and N action networks.

Remark 7.1. Before the statement about the neural networks, we firstly give a brief description:

1. Three-layer neural networks are utilized in this paper, which are input layer, hidden layer and output layer.
2. We randomly initialize the weights of input-to-hidden layer and keep them constant, and make the number of hidden layer neurons sufficiently large, then the neural network approximation errors can be small arbitrarily.
3. A sigmoid function defined as

$$S(x) = \frac{1 - e^{-x}}{1 + e^{-x}}, \tag{7.14}$$

is chose as the activation function.

7.3.2.1 *Critic network*

Based on the approximate capability of neural network, a critic network is used to approximate the value function and its derivatives. Then the output of critic network can be formulated as

$$\begin{aligned} \begin{bmatrix} \hat{V}_j(x_k) \\ \hat{\lambda}_j(x_k) \end{bmatrix} &= \begin{bmatrix} \hat{w}_{j,k,c2}^{a\mathsf{T}} \\ \hat{w}_{j,k,c2}^{b\mathsf{T}} \end{bmatrix} \phi(w_{j,c1}^{\mathsf{T}} X_{j,k}) \\ &= \hat{w}_{j,k,c2}^{\mathsf{T}} \phi(w_{j,c1}^{\mathsf{T}} X_{j,k}), \end{aligned} \tag{7.15}$$

in which $X_{j,k}$ is the input signal; $\hat{V}_j(x_k)$ and $\hat{\lambda}_j(x_k)$ denote the approximate values of $V_j(x_k)$ and $\lambda_j(x_k)$, respectively; $\hat{w}_{j,c1}$ is the weight vector between input layer and hidden layer, and the weight vector between hidden layer and output layer is denoted as $\hat{w}_{j,k,c2} = \left[\hat{w}_{j,k,c2}^{a}\ \hat{w}_{j,k,c2}^{b}\right]$. $\phi(\cdot)$ represents the activation function of the neural networks in the following statement.

According to (7.11), the approximate error $e_{j,k,c1}$ between $\hat{V}_j(x_k)$ and $\hat{V}_j(x_{k-1})$ can be defined as

$$e_{j,k,c1} = \alpha_j \hat{V}_j(x_k) - \hat{V}_j(x_{k-1}) + r_j(x_{k-1}). \tag{7.16}$$

Similarly, based on (7.13), the approximate error $e_{j,k,c2}$ between $\hat{\lambda}_j(x_k)$ and $\hat{\lambda}_j(x_{k-1})$ is described as

$$e_{j,k,c2} = \alpha_j \frac{\partial X_{j,k}}{\partial X_{j,k-1}} \hat{\lambda}_j(x_k) - \hat{\lambda}_j(x_{k-1}) + \frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}}. \tag{7.17}$$

In order to train the weights of critic network simultaneously, the objective is to minimize the following global error function:

$$E_{j,k,c} = \eta_1 E_{j,k,c1} + \eta_2 E_{j,k,c2}, \tag{7.18}$$

where η_1 and η_2 are the scaling factors with $0 < \eta_1 \leq 1$ and $0 < \eta_2 \leq 1$. $E_{j,k,c1}$ and $E_{j,k,c2}$ are as follows:

$$E_{j,k,c1} = \frac{1}{2} e_{j,k,c1}^2, \tag{7.19}$$

and

$$E_{j,k,c2} = \frac{1}{2} e_{j,k,c2}^2. \tag{7.20}$$

The gradient-based adaptation rule is used to update the weight as follows:

$$\hat{w}_{j,k+1,c2} = \hat{w}_{j,k,c2} + \Delta w_{j,k,c2}. \tag{7.21}$$

Let $\Delta w_{j,k,c2}$ be derived as

$$\begin{aligned}
\Delta w_{j,k,c2} &= -\beta_{j,c} \left[\frac{\partial E_{j,k,c}}{\partial \hat{w}_{j,k,c2}} \right] \\
&= -\beta_{j,c} \left[\eta_1 \frac{\partial E_{j,k,c1}}{\partial \hat{w}_{j,k,c2}} + \eta_2 \frac{\partial E_{j,k,c2}}{\partial \hat{w}_{j,k,c2}} \right] \\
&= -\beta_{j,c} \left[\eta_1 \alpha_j e_{j,k,c1} \phi_{j,k,c} \eta_2 \alpha_j \phi_{j,k,c} \left(\frac{\partial X_{j,k}}{\partial X_{j,k-1}} e_{j,k,c2} \right)^{\mathsf{T}} \right],
\end{aligned} \tag{7.22}$$

where $\beta_{j,c}$ is the learning rate of player j, $\phi_{j,k,c} = \phi(w_{j,c1}^{\mathsf{T}} X_{j,k})$ is the output of hidden layer.

Remark 7.2. Equation (7.22) is not easy to understand, following statements may be helpful. Suppose that there are H_c neurons in the hidden layer of the critic network. According to (7.1) and (7.15), the dimensions of $\hat{w}_{j,k,c2}$, $\phi_{j,k,c}$, $e_{j,k,c1}$, $e_{j,k,c2}$, and $\partial X_{j,k}/\partial X_{j,k-1}$ are shown below.

$$\begin{aligned}
\hat{w}_{j,k,c2} &\in \mathbb{R}^{H_c\times(1+n+m_1+m_2+\cdots+m_N)},\\
\phi_{j,k,c} &\in \mathbb{R}^{H_c\times 1},\\
e_{j,k,c1} &\in \mathbb{R}^{1\times 1},\\
\frac{\partial X_{j,k}}{\partial X_{j,k-1}} &\in \mathbb{R}^{(n+m_1+m_2+\cdots+m_N)\times(n+m_1+m_2+\cdots+m_N)},\\
e_{j,k,c2} &\in \mathbb{R}^{(n+m_1+m_2+\cdots+m_N)\times 1}.
\end{aligned}$$

It can be seen that the dimension of (7.22) is reasonable, which is equal to the design of $\hat{w}_{j,k,c2}$ in (7.15).

Define $w_{j,c2}$ to represent the optimal weight vector. We can have the approximate error $\tilde{w}_{j,k,c2}$ as

$$\tilde{w}_{j,k,c2} = \hat{w}_{j,k,c2} - w_{j,c2}. \tag{7.23}$$

Then according to (7.21) and (7.22), (7.23) can be rewritten as

$$\begin{aligned}
\tilde{w}_{j,k+1,c2} =& \ \hat{w}_{j,k+1,c2} - w_{j,c2}\\
=& \ \tilde{w}_{j,k,c2} + \Delta w_{j,k,c2}\\
=& \left[\tilde{w}^a_{j,k,c2}\ \tilde{w}^b_{j,k,c2}\right] - \beta_{j,c}\Bigg[\eta_1\alpha_j\phi_{j,k,c}\\
&\times(\alpha_j\hat{w}^{a\mathsf{T}}_{j,k,c2}\phi_{j,k,c} + r_j(x_{k-1}) - \hat{w}^{a\mathsf{T}}_{j,k-1,c2}\phi_{j,k-1,c})\\
&\eta_2\alpha_j\phi_{j,k,c}\left(\frac{\partial X_{j,k}}{\partial X_{j,k-1}}\left(\alpha_j\frac{\partial X_{j,k}}{\partial X_{j,k-1}}\hat{w}^{b\mathsf{T}}_{j,k,c2}\phi_{j,k,c}\right.\right.\\
&\left.\left.+\frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}} - \hat{w}^{b\mathsf{T}}_{j,k-1,c2}\phi_{j,k-1,c}\right)\right)^{\mathsf{T}}\Bigg].
\end{aligned} \tag{7.24}$$

7.3.2.2 *Action network*

Similar to the structure of critic network, define $w_{j,a1}$ as the constant weight vector between input layer and hidden layer, $\hat{w}_{j,k,a2}$ as the weight vector between hidden layer and output layer. Moreover, in the training process, we merely update the weight $\hat{w}_{j,k,a2}$. However, in the action network, only the system state x_k is used as input to obtain the optimal control as the output of the network. Then the output can be formulated as

$$\hat{u}_j(x_k) = \hat{w}_{j,k,a2}^{\mathsf{T}} \phi(w_{j,a1}^{\mathsf{T}} x_k), \tag{7.25}$$

in which $\phi(\cdot)$ is the activation function with the form of (7.14).

As shown in Fig. 7.1, the GDHP design for each player is a combination of HDP and DHP. According to (7.16)–(7.22), we can see that GDHP minimizes the error with respect to both $\hat{V}_j(x_k)$ and its derivatives. Since we already have high quality estimate of $\partial \hat{V}_j(x_k)/\partial \hat{u}_{j,k}$ in $\hat{\lambda}_j(x_k)$, the objective error function of the action network becomes

$$E_{j,k,a} = \frac{1}{2} e_{j,k,a}^{\mathsf{T}} e_{j,k,a}, \tag{7.26}$$

where $e_{j,k,a} = \hat{\lambda}_j(x_k)$.

Using the gradient-based adaptation rule, the weight $\hat{w}_{j,k,a2}$ can be updated by

$$\hat{w}_{j,k+1,a2} = \hat{w}_{j,k,a2} + \Delta w_{j,k,a2}, \tag{7.27}$$

in which $\Delta w_{j,k,a2}$ is

$$\begin{aligned} \Delta w_{j,k,a2} &= -\beta_{j,a} \frac{\partial E_{j,k,a}}{\partial e_{j,k,a}} \frac{\partial e_{j,k,a}}{\partial \hat{\lambda}_j(x_k)} \frac{\partial \hat{\lambda}_j(x_k)}{\partial \hat{u}_j(x_k)} \frac{\partial \hat{u}_j(x_k)}{\partial \hat{w}_{j,k,a2}} \\ &= -\beta_{j,a} \phi_{j,k,a} ([\hat{w}_{j,k,c2}^{b\mathsf{T}} \phi_{j,k,c}]^{\mathsf{T}} [\hat{w}_{j,k,c2}^{b\mathsf{T}} \Gamma_{j,k}]), \end{aligned} \tag{7.28}$$

where $\beta_{j,a}$ is learning rate of play j, $\phi_{j,k,a} = \phi(w_{j,a1}^{\mathsf{T}} x_k)$,

$$\Gamma_{j,k} = (1/2)[\underbrace{(1-\phi_{j,k,c}^2) \ldots (1-\phi_{j,k,c}^2)}_{m_j}] w_{j,c1,n+j}^{\mathsf{T}}.$$

Remark 7.3. The formula of $\Delta w_{j,k,a2}$ may be quite complicated, exact derivations of dimensions will be given. Suppose that there are H_a neurons in the hidden layer, the dimension of $w_{j,k,a2}$ is $\mathbb{R}^{H_a \times m_j}$. Other dimensions of (7.28) are presented based on Remark 7.2.

$$\begin{aligned}
\phi_{j,k,a} &\in \mathbb{R}^{H_a \times 1},\\
\hat{w}_{j,k,c2}^{b\mathsf{T}} \phi_{j,k,c} &\in \{\mathbb{R}^{H_c \times (n+m_1+m_2+\cdots+m_N)}\}^{\mathsf{T}} \times \mathbb{R}^{H_c \times 1}\\
&= \mathbb{R}^{(n+m_1+m_2+\cdots+m_N)\times 1},\\
1-\phi_{j,k,c}^2 &\in \mathbb{R}^{H_c \times 1},\\
w_{j,c1} &\in \mathbb{R}^{(n+m_1+\cdots+m_j+\cdots+m_N)\times H_c},\\
w_{j,c1,n+j} &\in \mathbb{R}^{m_j \times H_c},\\
\Gamma_{j,k} &\in \mathbb{R}^{H_c \times m_j} \times \{\mathbb{R}^{m_j \times H_c}\}^{\mathsf{T}} = \mathbb{R}^{H_c \times m_j},\\
\hat{w}_{j,k,c2}^{b\mathsf{T}} \Gamma_{j,k} &\in \{\mathbb{R}^{H_c \times (n+m_1+m_2+\cdots+m_N)}\}^{\mathsf{T}} \times \mathbb{R}^{H_c \times m_j}\\
&= \mathbb{R}^{(n+m_1+m_2+\cdots+m_N)\times m_j}.
\end{aligned}$$

From the above analysis, we can see that the dimension of (7.28) are also reasonable.

Define the ideal weight vector of $\hat{w}_{j,k,a2}$ as $w_{j,a2}$. Then the approximate error between $\hat{w}_{j,k,a2}$ and $w_{j,a2}$ is defined as

$$\tilde{w}_{j,k,a2} = \hat{w}_{j,k,a2} - w_{j,a2}. \tag{7.29}$$

According to (7.27) and (7.28), (7.29) can be rewritten as

$$\begin{aligned}
\tilde{w}_{j,k+1,a2} &= \hat{w}_{j,k+1,a2} - w_{j,a2}\\
&= \tilde{w}_{j,k,a2} + \Delta w_{j,k,a2}\\
&= \tilde{w}_{j,k,a2} - \beta_{j,a}\phi_{j,k,a}\left(\left[\hat{w}_{j,k,c2}^{b\mathsf{T}}\phi_{j,k,c}\right]^{\mathsf{T}}\left[\hat{w}_{j,k,c2}^{b\mathsf{T}}\Gamma_{j,k}\right]\right).
\end{aligned} \tag{7.30}$$

From the updating rules (7.21) of critic network and (7.27) of action network, it is unnecessary to use a neural network or identifier to identify the unknown nonlinear system for the dynamic information $f(x_k)$ and $g_{j,k}$. Subscripts j, $c1$, $c2$, $a1$, $a2$, a and b are used to

distinguish different parameters, only the system states x_{k-1} and x_k are needed in the learning process. $\beta_{j,c}$ and $\beta_{j,a}$ are the learning rates of the jth critic network and the jth action network. As we know, to address the MP-NZSG, we have to solve (7.9) which is a coupled equation. As shown in Fig. 7.1, the input signal of the jth critic network includes not only the output of the jth action network, but also the outputs of other action networks. This structure can couple all critic networks and action networks together. Furthermore, for each player, the weight of action network is directly trained by the output of the critic network. For the convenience of analysis later, the subscript "2", which denotes the hidden layer to output layer, is omitted. According to (7.7)–(7.8), the obtained policy $u_{j,k}$ could be optimal control when the value function $V_j(x_{k+1})$ is optimal. Therefore, the training process of all neural networks can be summarized as: First, the 1th, 2th, ..., Nth critic networks are trained one by one. After that, we update the weights $\hat{w}_{j,k,c}, j = 1, 2, \ldots, N$. Second, the 1th, 2th, ..., Nth action networks are trained also one by one. Thereafter, we fix the weights $\hat{w}_{j,k,a}, j = 1, 2, \ldots, N$. Third, since all the weights of neural networks have been trained in this round, according to the next system state, we start a new training round, and so on, until we get the optimal weights of all neural networks. Then the above training process of the neural network weights is summarized in Algorithm 7.1.

7.4 Stability Analysis of MF-CGDHP Algorithm

The MF-CGDHP structure shown in Fig. 7.1 is used to implement the MF-CGDHP algorithm, which is employed to solve the coupled HJ equation (7.9) and obtain the optimal control policies. In this section, based on Lyapunov stability construct, it is proved that the estimation errors of all critic and action networks weights are UUB. Before the stability analysis, we present the following assumption.

Assumption 7.3. Let $w_{j,c}$ and $w_{j,a}$ be the optimal weights for the jth critic network and action network, respectively and assume they are bounded by two positive constants, i.e.,

$$\|w_{j,c}\| < w_{j,c,m}, \quad \|w_{j,a}\| < w_{j,a,m}, \quad j = 1, 2, \ldots, N, \tag{7.31}$$

in which m means "maximum".

Algorithm 7.1 MF-CGDHP for MP-NZSG

Require:

Critic network: $V_j(x_c) \leftarrow p_{j,c}(x_c, \hat{w}_{j,c})$
j: $j = 1, 2, \ldots, N$;
$V_j(x_c)$: the jth value function;
$p_{j,c}$: the jth critic network;
x_c: the input signal of critic network, $x_c = [x_k^\mathsf{T}, \hat{u}_1^\mathsf{T}, \hat{u}_2^\mathsf{T}, \ldots, \hat{u}_N^\mathsf{T}]^\mathsf{T}$;
$\hat{w}_{j,c}$: the estimated weight of critic network;
Action network: $\hat{u}_j(x_a) \leftarrow p_{j,a}(x_a, \hat{w}_{j,a})$
j: $j = 1, 2, \ldots, N$;
$\hat{u}_j(x_a)$: the jth action function;
$p_{j,a}$: the jth action network;
x_a: the input signal of action network, $x_a = x_k$;
$\hat{w}_{j,a}$: the estimated weight of action network;
$N_{j,c}$ and $N_{j,a}$: the maximum cycles of the jth critic network and action network, respectively;
$\varepsilon_{j,c}$ and $\varepsilon_{j,a}$: internal training error thresholds for the jth critic network and action network, respectively;
N_{cyc}: the maximum cycle of learning;

Ensure:

for $1 : N_{cyc}$ **do**
  Initialize x_0, $\hat{w}_{j,0,a}$ and $\hat{w}_{j,0,c}$ with $j = 1, 2, \ldots, N$;
  Execute the jth action and obtain current state;
  $x_k = \text{system}(x_{k-1}, \hat{u}_{1,k-1}, \hat{u}_{2,k-1}, \ldots, \hat{u}_{N,k-1})$;

  Block of training critic network, $j = 1, 2, \ldots, N$.
  while $(E_{j,k,c} > \varepsilon_{j,c}$ & $cyc < N_{j,c})$ **do**
    $V_j(x_c) \leftarrow p_{j,k,c}(x_c, \hat{w}_{j,k,c})$;
    $e_{j,k,c1} =$ (7.16), $e_{j,k,c2} =$ (7.17), $E_{j,k,c} =$ (7.18);
    Update the weight $\hat{w}_{j,k,c}$ by (7.21) and (7.22);
    $\hat{w}_{j,k+1,c} = \hat{w}_{j,k,c} + \Delta w_{j,k,c}$.
  end while

  Block of training action network, $j = 1, 2, \ldots, N$.
  while $(E_{j,k,a} > \varepsilon_{j,a}$ & $cyc < N_{j,a})$ **do**
    $\hat{u}_j(x_a) \leftarrow p_{j,a}(x_a, \hat{w}_{j,a})$;
    $V_j(x_c) \leftarrow p_{j,k,c}(x_c, \hat{w}_{j,k,c})$;
    $e_{j,k,a} = \hat{\lambda}_j(x_k)$, $E_{j,k,a} =$ (7.26);
    Update the weight $\hat{w}_{j,k,a}$ by (7.27) and (7.28);
    $\hat{w}_{j,k,a} = \hat{w}_{j,k,a} + \Delta w_{j,k,a}$;
  end while

  $\hat{w}_{j,k+1,c} = \hat{w}_{j,k,c}$, $j = 1, 2, \ldots, N$;
  $\hat{w}_{j,k+1,a} = \hat{w}_{j,k,a}$, $j = 1, 2, \ldots, N$.
end for
Return $\hat{w}_{j,N_{cyc},c}$, $\hat{w}_{j,N_{cyc},a}$, $j = 1, 2, \ldots, N$.

In order to analyze approximate error (7.24) of critic network conveniently, two definitions (7.32) and (7.33) are needed. According to the definition of critic network (7.15), the output of critic network not only estimates the value function $V_j(x_k)$, but also includes its derivatives $\lambda_j(x_k)$. Then, in the approximate error (7.24), there are two terms in the vector $\tilde{w}_{j,k+1,c2}$. The first one is weight approximation error of value function, as shown in (7.32). The second one is weight approximation error of its derivatives, as shown in (7.33).

$$\begin{aligned} C^a_{j,k} &= \tilde{w}^a_{j,k,c} - \beta_{j,c}\eta_1\alpha_j\phi_{j,k,c}(\alpha_j\hat{w}^{a\mathsf{T}}_{j,k,c}\phi_{j,k,c} \\ &\quad + r_j(x_{k-1}) - \hat{w}^{a\mathsf{T}}_{j,k-1,c}\phi_{j,k-1,c}) \\ &= [I - \alpha_j^2\beta_{j,c}\eta_1\phi_{j,k,c}\phi^{\mathsf{T}}_{j,k,c}]\tilde{w}^a_{j,k,c} \\ &\quad - \beta_{j,c}\eta_1\alpha_j\phi_{j,k,c}[\alpha_j w^{a\mathsf{T}}_{j,c}\phi_{j,k,c} \\ &\quad + r_j(x_{k-1}) - \hat{w}^{a\mathsf{T}}_{j,k-1,c}\phi_{j,k-1,c}. \end{aligned} \tag{7.32}$$

$$\begin{aligned} C^b_{j,k} &= \tilde{w}^b_{j,k,c} - \beta_{j,c}\eta_2\alpha_j\phi_{j,k,c}\left[\frac{\partial X_{j,k}}{\partial X_{j,k-1}}\left(\alpha_j\frac{\partial X_{j,k}}{\partial X_{j,k-1}}\right.\right. \\ &\quad \left.\left.\times\hat{w}^{b\mathsf{T}}_{j,k,c}\phi_{j,k,c} + \frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}} - \hat{w}^{b\mathsf{T}}_{j,k-1,c}\phi_{j,k-1,c}\right)\right]^{\mathsf{T}} \\ &= (I - \alpha_j^2\|A\|^2\beta_{j,c}\eta_2\phi_{j,k,c}\phi^{\mathsf{T}}_{j,k,c})\tilde{w}^b_{j,k,c} \\ &\quad - \beta_{j,c}\eta_2\alpha_j\phi_{j,k,c}\left[A\left(\alpha_j A w^{b\mathsf{T}}_{j,c}\phi_{j,k,c}\right.\right. \\ &\quad \left.\left.+\frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}} - \hat{w}^{b\mathsf{T}}_{j,k-1,c}\phi_{j,k-1,c}\right)\right]^{\mathsf{T}}. \end{aligned} \tag{7.33}$$

$A = \partial X_{j,k}/\partial X_{j,k-1}$, $w^a_{j,c}$ and $w^b_{j,c}$ denote the optimal value of $\hat{w}^a_{j,k,c}$ and $\hat{w}^b_{j,k,c}$, respectively. Then (7.24) can be written as

$$\tilde{w}_{j,k+1,c} = \hat{w}_{j,k+1,c} - w_{j,c} = \left[C^a_{j,k}, C^b_{j,k}\right]. \tag{7.34}$$

Consider the two update rules for approximate errors in (7.24) and (7.30), and rewrite these learning processes into a more general

form as

$$\tilde{w}_{j,k+1} = \hat{w}_{j,k} - p_j\left(\hat{w}_{j,k}, \hat{w}_{j,k-1}, \phi_{j,k}, \phi_{j,k-1}\right). \tag{7.35}$$

Then the asymptotic behavior of the estimation error of the weights, $w_{j,k}$, is to be analyzed by studying the stability properties of system (7.35).

Definition 7.3 (Uniformly Ultimately Boundedness). The solution, $w_{j,k}$, of the discrete-time dynamic system (7.35) is said to be UUB within a bound $\varepsilon > 0$, if for any $\delta > 0$ and $t_0 > 0$, there exists a positive number $N = N(\delta, \varepsilon)$ independent of t_0, such that $\|\tilde{w}_{j,k}\| \leq \varepsilon$ for all $t \geq N + t_0$ when $\|\tilde{w}_{j,t_0}\| \leq \delta$.

Theorem 7.1. *Consider the discrete-time nonlinear system* (7.1)*, let the N critic and action networks output be given as* (7.15) *and* (7.25)*, the N critic and action networks weights update according to* (13) *and* (16)*, respectively. Then, the sum of all critic network estimation errors* $\sum \tilde{w}_{j,k,c}$*, and action network estimation errors* $\sum \tilde{w}_{j,k,a}$ *are UUB, respectively, provided the following conditions are met*:

$$\frac{1}{\sqrt{2}} < \alpha_j < 1, \quad j = 1, 2, \ldots, N, \tag{7.36}$$

$$\beta_{j,c} < \frac{1}{\left(\alpha_j^2 \eta_1 \|\phi_{j,k,c}\|^2\right)}$$

or

$$\beta_{j,c} < \frac{1}{\left(\alpha_j^2 \eta_2 \|A\|^2 \|\phi_{j,k,c}\|^2\right)}, \quad j = 1, 2, \ldots, N, \tag{7.37}$$

$$\beta_{j,a} < \frac{1}{\|\phi_{j,k,a}\|^2}, \quad j = 1, 2, \ldots, N. \tag{7.38}$$

Proof. Define the Lyapunov function candidate as

$$\begin{aligned} L_{S,k} = & \sum_{j=1}^{N} \left\{ \frac{1}{\beta_{j,c}} \mathrm{tr}\left\{\tilde{w}_{j,k,c}^{\mathsf{T}} \tilde{w}_{j,k,c}\right\} + \frac{1}{l_j \beta_{j,a}} \mathrm{tr}\left\{\tilde{w}_{j,k,a}^{\mathsf{T}} \tilde{w}_{j,k,a}\right\} \right. \\ & \left. + \frac{1}{2}\eta_1 \left\|\varsigma_{j,k-1}^{a}\right\|^2 + \frac{1}{2}\eta_2 \|\varsigma_{j,k-1}^{b}\|^2 \right\} \\ = & \sum_{j=1}^{N} L_{j,1} + L_{j,2} + L_{j,3} + L_{j,4} = \sum_{j=1}^{N} L_{j,k}, \end{aligned} \tag{7.39}$$

where $\varsigma^a_{j,k-1,c} = \tilde{w}^{a\mathsf{T}}_{j,k-1,c}\phi_{j,k-1,c}$, $\varsigma^b_{j,k-1,c} = \tilde{w}^{b\mathsf{T}}_{j,k-1,c}\phi_{j,k-1,c}$. Then the first difference of $\Delta L_{j,k}$ in (7.39) will be

$$\Delta L_{j,k} = \Delta L_{j,1} + \Delta L_{j,2} + \Delta L_{j,3} + \Delta L_{j,4}. \tag{7.40}$$

For the convergence of calculation, we analyze each term separately,

$$\begin{aligned}\Delta L_{j,1} &= \frac{1}{\beta_{j,c}}\mathrm{tr}\left\{\tilde{w}^{\mathsf{T}}_{j,k+1,c}\tilde{w}_{j,k+1,c} - \tilde{w}^{\mathsf{T}}_{j,k,c}\tilde{w}_{j,k,c}\right\} \\ &= \frac{1}{\beta_{j,c}}\mathrm{tr}\left\{C^{a\mathsf{T}}_{j,k}C^a_{j,k} - \tilde{w}^{a\mathsf{T}}_{j,k,c}\tilde{w}^a_{j,k,c}\right\} \\ &\quad + \frac{1}{\beta_{j,c}}\mathrm{tr}\left\{C^{b\mathsf{T}}_{j,k}C^b_{j,k} - \tilde{w}^{b\mathsf{T}}_{j,k,c}\tilde{w}^b_{j,k,c}\right\}.\end{aligned} \tag{7.41}$$

Define the two terms in last row of (7.41) as follows:

$$\Delta L_{j,11} = \frac{1}{\beta_{j,c}}\mathrm{tr}\left\{C^{a\mathsf{T}}_{j,k}C^a_{j,k} - \tilde{w}^{a\mathsf{T}}_{j,k,c}\tilde{w}^a_{j,k,c}\right\}, \tag{7.42}$$

$$\Delta L_{j,12} = \frac{1}{\beta_{j,c}}\mathrm{tr}\left\{C^{b\mathsf{T}}_{j,k}C^b_{j,k} - \tilde{w}^{b\mathsf{T}}_{j,k,c}\tilde{w}^b_{j,k,c}\right\}. \tag{7.43}$$

Considering the first term $\Delta L_{j,11}$, we have

$$\begin{aligned}\Delta L_{j,11} &= \frac{1}{\beta_{j,c}}\mathrm{tr}\left\{C^{a\mathsf{T}}_{j,k}C^a_{j,k} - \tilde{w}^{a\mathsf{T}}_{j,k,c}\tilde{w}^a_{j,k,c}\right\} \\ &= \frac{1}{\beta_{j,c}}\mathrm{tr}\left\{\tilde{w}^{a\mathsf{T}}_{j,k,c}B^{\mathsf{T}}B\tilde{w}^a_{j,k,c} - \tilde{w}^{a\mathsf{T}}_{j,k,c}\tilde{w}^a_{j,k,c}\right. \\ &\quad - \tilde{w}^{a\mathsf{T}}_{j,k,c}B^{\mathsf{T}}\beta_{j,c}\eta_1\alpha_j\phi_{j,k,c}C^{\mathsf{T}} \\ &\quad - \beta_{j,c}\eta_1\alpha_j C\phi^{\mathsf{T}}_{j,k,c}B\tilde{w}^a_{j,k,c} \\ &\quad \left.+\beta^2_{j,c}\eta^2_1\alpha^2_j C\phi^{\mathsf{T}}_{j,k,c}\phi_{j,k,c}C^{\mathsf{T}}\right\},\end{aligned} \tag{7.44}$$

in which $B = I - \alpha^2_j\beta_{j,c}\eta_1\phi_{j,k,c}\phi^{\mathsf{T}}_{j,k,c}$, $C = \alpha_j w^{a\mathsf{T}}_{j,c}\phi_{j,k,c} + r_j(x_{k-1}) - \hat{w}^{a\mathsf{T}}_{j,k-1,c}\phi_{j,k-1,c}$,

$$\begin{aligned}&\tilde{w}^{a\mathsf{T}}_{j,k,c}B^{\mathsf{T}}B\tilde{w}^a_{j,k,c} - \tilde{w}^{a\mathsf{T}}_{j,k,c}\tilde{w}^a_{j,k,c} \\ &= \tilde{w}^{a\mathsf{T}}_{j,k,c}\left[I - \alpha^2_j\beta_{j,c}\eta_1\phi_{j,k,c}\phi^{\mathsf{T}}_{j,k,c}\right]^{\mathsf{T}}\end{aligned}$$

$$\begin{aligned}
&\times \left[I - \alpha_j^2 \beta_{j,c} \eta_1 \phi_{j,k,c} \phi_{j,k,c}^{\mathsf{T}}\right] \tilde{w}_{j,k,c}^a - \tilde{w}_{j,k,c}^{a\mathsf{T}} \tilde{w}_{j,k,c}^a \\
&= -\alpha_j^2 \beta_{j,c} \eta_1 \left\|\varsigma_{j,k,c}^a\right\|^2 - \alpha_j^2 \beta_{j,c} \eta_1 \\
&\quad \times \left(1 - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right) \left\|\varsigma_{j,k,c}^a\right\|^2. \qquad (7.45)
\end{aligned}$$

Substituting (7.45) into (7.44), we get

$$\begin{aligned}
\Delta L_{j,11} = &- \alpha_j^2 \eta_1 \left\|\varsigma_{j,k,c}^a\right\|^2 + \beta_{j,c} \eta_1^2 \alpha_j^2 \|\phi_{j,k,c}\|^2 \|C\|^2 \\
&- \alpha_j^2 \eta_1 \left(1 - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right) \left\|\varsigma_{j,k,c}^a\right\|^2 \qquad (7.46) \\
&- 2\mathrm{tr}\left\{\eta_1 \alpha_j \left[I - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right] \varsigma_{j,k,c}^a C^T\right\}.
\end{aligned}$$

The last two terms in (7.46) can be rewritten as

$$\begin{aligned}
&- \alpha_j^2 \eta_1 \left(1 - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right) \left\|\varsigma_{j,k,c}^a\right\|^2 \\
&- 2\mathrm{tr}\left\{\eta_1 \alpha_j \left[I - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right] \varsigma_{j,k,c}^a C^{\mathsf{T}}\right\} \\
= &- \alpha_j^2 \eta_1 \left(1 - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right) \\
&\times \left\{\left\|\varsigma_{j,k,c}^a\right\|^2 + 2\mathrm{tr}\left\{\varsigma_{j,k}^a \alpha_j^{-1} C^{\mathsf{T}}\right\}\right\} \qquad (7.47) \\
= &- \alpha_j^2 \eta_1 \left(1 - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right) \\
&\times \left\{\left\|\varsigma_{j,k,c}^a + \alpha_j^{-1} C\right\|^2 - \left\|\alpha_j^{-1} C\right\|^2\right\}.
\end{aligned}$$

Substituting (7.47) into (7.46), we have

$$\begin{aligned}
\Delta L_{j,11} = &- \alpha_j^2 \eta_1 \left\|\varsigma_{j,k,c}^a\right\|^2 + \beta_{j,c} \eta_1^2 \alpha_j^2 \|\phi_{j,k,c}\|^2 \|C\|^2 \\
&- \alpha_j^2 \eta_1 \left(1 - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right) \left\|\varsigma_{j,k,c}^a + \alpha_j^{-1} C\right\|^2 \\
&+ \eta_1 \left(1 - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right) \|C\|^2 \qquad (7.48) \\
= &- \alpha_j^2 \eta_1 \left\|\varsigma_{j,k,c}^a\right\|^2 + \eta_1 \|C\|^2 \\
&- \alpha_j^2 \eta_1 \left(1 - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right) \left\|\varsigma_{j,k,c}^a + \alpha_j^{-1} C\right\|^2.
\end{aligned}$$

By substituting C into (7.48) and applying the Cauchy–Schwarz inequality, we have

$$\begin{aligned}
\Delta L_{j,11} \leq & -\alpha_j^2\eta_1\|\varsigma_{j,k,c}^a|^2 - \alpha_j^2\eta_1\left(1-\alpha_j^2\beta_{j,c}\eta_1\|\phi_{j,k,c}\|^2\right) \\
& \times \|\varsigma_{j,k,c}^a + w_{j,c}^{a\mathsf{T}}\phi_{j,k,c} \\
& +\alpha_j^{-1}r_j(x_{k-1}) - \alpha_j^{-1}\hat{w}_{j,k-1,c}^{a\mathsf{T}}\phi_{j,k-1,c}\|^2 \\
& + 2\eta_1\left\{\left\|\alpha_j w_{j,c}^{a\mathsf{T}}\phi_{j,k,c} + r_j(x_{k-1})\right.\right. \\
& \left. - \frac{1}{2}\hat{w}_{j,k-1,c}^{a\mathsf{T}}\phi_{j,k-1,c} - \frac{1}{2}w_{j,k-1,c}^{a\mathsf{T}}\phi_{j,k-1,c}\right\|^2 \\
& \left. + \left\|\frac{1}{2}\tilde{w}_{j,k-1,c}^{a\mathsf{T}}\phi_{j,k-1,c}\right\|^2\right\} \qquad (7.49) \\
= & -\alpha_j^2\eta_1\|\varsigma_{j,k,c}^a\|^2 - \alpha_j^2\eta_1\left(1-\alpha_j^2\beta_{j,c}\eta_1\|\phi_{j,k,c}\|^2\right) \\
& \times \|\varsigma_{j,k,c}^a + w_{j,c}^{a\mathsf{T}}\phi_{j,k,c} \\
& + \alpha_j^{-1}r_j(x_{k-1}) - \alpha_j^{-1}\hat{w}_{j,k-1,c}^{a\mathsf{T}}\phi_{j,k-1,c}\|^2 \\
& + \frac{\eta_1}{2}\|\varsigma_{j,k-1,c}^a\|^2 + 2\eta_1\left\|\alpha_j w_{j,c}^{a\mathsf{T}}\phi_{j,k,c} + r_j(x_{k-1})\right. \\
& \left. - \frac{1}{2}\hat{w}_{j,k-1,c}^{a\mathsf{T}}\phi_{j,k-1,c} - \frac{1}{2}w_{j,k-1,c}^{a\mathsf{T}}\phi_{j,k-1,c}\right\|^2.
\end{aligned}$$

$\Delta L_{j,12}$ can be analyzed similarly. According to (7.43),

$$\begin{aligned}
\Delta L_{j,12} = & \frac{1}{\beta_{j,c}}\mathrm{tr}\left\{C_{j,k}^{b\mathsf{T}}C_{j,k}^b - \tilde{w}_{j,k,c}^{b\mathsf{T}}\tilde{w}_{j,k,c}^b\right\} \\
= & \frac{1}{\beta_{j,c}}\mathrm{tr}\left\{\tilde{w}_{j,k,c}^{b\mathsf{T}}D^{\mathsf{T}}D\tilde{w}_{j,k,c}^b - \tilde{w}_{j,k,c}^{b\mathsf{T}}\tilde{w}_{j,k,c}^b\right. \\
& + \beta_{j,c}^2\eta_2^2\alpha_j^2((EA^{\mathsf{T}})\phi_{j,k,c}^{\mathsf{T}})(\phi_{j,k,c}(AE^{\mathsf{T}})) \\
& - \beta_{j,c}\eta_2\alpha_j((E^{\mathsf{T}}A)\phi_{j,k,c}^{\mathsf{T}})(D\tilde{w}_{j,k,c}^b) \\
& \left. -\beta_{j,c}\eta_2\alpha_j(\tilde{w}_{j,k,c}^{b\mathsf{T}}D^{\mathsf{T}})(\phi_{j,k,c}(AE^{\mathsf{T}}))\right\},
\end{aligned} \qquad (7.50)$$

where $D = I - \alpha_j^2 \|A\|^2 \beta_{j,c} \eta_2 \phi_{j,k,c} \phi_{j,k,c}^{\mathsf{T}}$, $E = \alpha_j A w_{j,c}^{b\mathsf{T}} \phi_{j,k,c} + \partial r_j(x_{k-1})/\partial X_{j,k-1} - \hat{w}_{j,k-1,c}^{b\mathsf{T}} \phi_{j,k-1,c}$. The first two terms of (7.50) can be rewritten as

$$\begin{aligned}
&\tilde{w}_{j,k,c}^{b\mathsf{T}} D^{\mathsf{T}} D \tilde{w}_{j,k,c}^{b} - \tilde{w}_{j,k,c}^{b\mathsf{T}} \tilde{w}_{j,k,c}^{b} \\
&\quad = \tilde{w}_{j,k,c}^{b\mathsf{T}} \left(I - \alpha_j^2 \|A\|^2 \beta_{j,c} \eta_2 \phi_{j,k,c} \phi_{j,k,c}^{\mathsf{T}} \right) \\
&\qquad \times \left(I - \alpha_j^2 \|A\|^2 \beta_{j,c} \eta_2 \phi_{j,k,c} \phi_{j,k,c}^{\mathsf{T}} \right) \tilde{w}_{j,k,c}^{b} - \tilde{w}_{j,k,c}^{b\mathsf{T}} \tilde{w}_{j,k,c}^{b} \\
&\quad = -\alpha_j^2 \beta_{j,c} \eta_2 \|A\|^2 \left\| \varsigma_{j,k,c}^{b} \right\|^2 - \alpha_j^2 \beta_{j,c} \eta_2 \\
&\qquad \times \left(1 - \alpha_j^2 \beta_{j,c} \eta_2 \|A\|^2 \|\phi_{j,k,c}\|^2 \right) \|A\|^2 \left\| \varsigma_{j,k,c}^{b} \right\|^2 .
\end{aligned} \tag{7.51}$$

Substituting (7.51) into (7.50), we have

$$\begin{aligned}
\Delta L_{j,12} = &-\alpha_j^2 \eta_2 \|A\|^2 \left\| \varsigma_{j,k,c}^{b} \right\|^2 \\
&+ \beta_{j,c} \eta_2^2 \alpha_j^2 \|A\|^2 \|\phi_{j,k,c}\|^2 \|E\|^2 \\
&- \alpha_j^2 \eta_2 \left(1 - \alpha_j^2 \|A\|^2 \beta_{j,c} \eta_2 \|A\|^2 \|\phi_{j,k,c}\|^2 \right) \\
&\times \|A\|^2 \left\| \varsigma_{j,k,c}^{b} \right\|^2 \\
&- 2\mathrm{tr} \left\{ \eta_2 \alpha_j \left[1 - \alpha_j^2 \beta_{j,c} \eta_2 \|A\|^2 \|\phi_{j,k,c}\|^2 \right] \right. \\
&\left. \times \varsigma_{j,k,c}^{b} (A E^{\mathsf{T}}) \right\},
\end{aligned} \tag{7.52}$$

in which

$$\begin{aligned}
&-\alpha_j^2 \eta_2 \left(1 - \alpha_j^2 \beta_{j,c} \eta_2 \|A\|^2 \|\phi_{j,k,c}\|^2 \right) \|A\|^2 \left\| \varsigma_{j,k,c}^{b} \right\|^2 \\
&\quad - 2\mathrm{tr} \left\{ \eta_2 \alpha_j \left[1 - \alpha_j^2 \beta_{j,c} \eta_2 \|A\|^2 \|\phi_{j,k,c}\|^2 \right] \varsigma_{j,k,c}^{b} (A E^{\mathsf{T}}) \} \right\} \\
&\quad = -\alpha_j^2 \eta_2 \left(1 - \alpha_j^2 \beta_{j,c} \eta_2 \|A\|^2 \|\phi_{j,k,c}\|^2 \right) \\
&\qquad \times \left\{ \|A\|^2 \left\| \varsigma_{j,k,c}^{b} \right\|^2 + 2\mathrm{tr} \left\{ \alpha_j^{-1} \varsigma_{j,k,c}^{b} (A E^{\mathsf{T}}) \right\} \right\}
\end{aligned}$$

$$
\begin{aligned}
&= -\alpha_j^2\eta_2\left(1-\alpha_j^2\beta_{j,c}\eta_2\|A\|^2\|\phi_{j,k,c}\|^2\right)\\
&\quad\times\left\{\|A\|^2\left\|\varsigma_{j,k,c}^b\right\|^2+2\mathrm{tr}\left\{\varsigma_{j,k,c}^b(AE^{\mathsf{T}})\right\}\right.\\
&\quad\left.+\left\|\alpha_j^{-1}E\right\|^2-\left\|\alpha_j^{-1}E\right\|^2\right\}\\
&= -\alpha_j^2\eta_2\left(1-\alpha_j^2\beta_{j,c}\eta_2\|A\|^2\|\phi_{j,k,c}\|^2\right)\\
&\quad\times\left\|A^{\mathsf{T}}\varsigma_{j,k,c}^b+\alpha_j^{-1}E\right\|^2\\
&\quad+\eta_2\left(1-\alpha_j^2\beta_{j,c}\eta_2\|A\|^2\|\phi_{j,k,c}\|^2\right)\|E\|^2. \qquad (7.53)
\end{aligned}
$$

Substituting (7.53) into (7.52), we get

$$
\begin{aligned}
\Delta L_{j,12} =& -\alpha_j^2\eta_2\|A\|^2\left\|\varsigma_{j,k,c}^b\right\|^2+\beta_{j,c}\eta_2^2\alpha_j^2\|A\|^2\|\phi_{j,k,c}\|^2\|E\|^2\\
&-\alpha_j^2\eta_2\left(1-\alpha_j^2\beta_{j,c}\eta_2\|A\|^2\|\phi_{j,k,c}\|^2\right)\times\left\|A^{\mathsf{T}}\varsigma_{j,k,c}^b+\alpha_j^{-1}E\right\|^2\\
&+\eta_2\left(1-\alpha_j^2\beta_{j,c}\eta_2\|A\|^2\|\phi_{j,k,c}\|^2\right)\|E\|^2\\
=& -\alpha_j^2\eta_2\|A\|^2\left\|\varsigma_{j,k,c}^b\right\|^2+\eta_2\|E\|^2\\
&-\alpha_j^2\eta_2\left(1-\alpha_j^2\beta_{j,c}\eta_2\|A\|^2\|\phi_{j,k,c}\|^2\right)\times\left\|A^{\mathsf{T}}\varsigma_{j,k,c}^b+\alpha_j^{-1}E\right\|^2. \qquad (7.54)
\end{aligned}
$$

Based on the Cauchy–Schwarz inequality, we have

$$
\begin{aligned}
\Delta L_{j,12} =& -\alpha_j^2\eta_2\|A\|^2\left\|\varsigma_{j,k,c}^b\right\|^2+\eta_2\|E\|^2\\
&-\alpha_j^2\eta_2\left(1-\alpha_j^2\beta_{j,c}\eta_2\|A\|^2\|\phi_{j,k,c}\|^2\right)\times\left\|A^{\mathsf{T}}\varsigma_{j,k,c}^b+\alpha_j^{-1}E\right\|^2\\
=& -\alpha_j^2\eta_2\|A\|^2\left\|\varsigma_{j,k,c}^b\right\|^2\\
&-\alpha_j^2\eta_2\left(1-\alpha_j^2\beta_{j,c}\eta_2\|A\|^2\|\phi_{j,k,c}\|^2\right)\times\left\|A^{\mathsf{T}}\varsigma_{j,k,c}^b+\alpha_j^{-1}E\right\|^2
\end{aligned}
$$

$$
\begin{aligned}
&+\eta_2\left\|\alpha_j A w_{j,c}^{b\mathsf{T}}\phi_{j,k,c}+\frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}}\right.\\
&-\frac{1}{2}\hat{w}_{j,k-1,c}^{b\mathsf{T}}\phi_{j,k-1,c}-\frac{1}{2}w_{j,k-1,c}^{b\mathsf{T}}\phi_{j,k-1,c}\\
&\left.-\frac{1}{2}\tilde{w}_{j,k-1,c}^{b\mathsf{T}}\phi_{j,k-1,c}\right\|^2\\
\le&-\alpha_j^2\eta_2\|A\|^2\left\|\varsigma_{j,k,c}^b\right\|^2\\
&-\alpha_j^2\eta_2\left(1-\alpha_j^2\beta_{j,c}\eta_2\|A\|^2\|\phi_{j,k,c}\|^2\right)\times\left\|A^{\mathsf{T}}\varsigma_{j,k,c}^b+\alpha_j^{-1}E\right\|^2\\
&+2\eta_2\left\{\left\|\alpha_j A w_{j,c}^{b\mathsf{T}}\phi_{j,k,c}+\frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}}\right.\right.\\
&-\frac{1}{2}\hat{w}_{j,k-1,c}^{b\mathsf{T}}\phi_{j,k-1,c}-\frac{1}{2}w_{j,k-1,c2}^{b\mathsf{T}}\phi_{j,k-1,c}\Big\|^2\\
&\left.+\left\|\frac{1}{2}\varsigma_{j,k-1,c}^b\right\|^2\right\}.
\end{aligned}
\tag{7.55}
$$

Equation (7.55) can be further simplified to

$$
\begin{aligned}
\Delta L_{j,12}\le&-\alpha_j^2\eta_2\|A\|^2\left\|\varsigma_{j,k,c}^b\right\|^2\\
&-\alpha_j^2\eta_2\left(1-\alpha_j^2\beta_{j,c}\eta_2\|A\|^2\|\phi_{j,k,c}\|^2\right)\\
&\times\left\|A^{\mathsf{T}}\varsigma_{j,k,c}^b+\alpha_j^{-1}E\right\|^2\\
&+2\eta_2\left\|\alpha_j A w_{j,c}^{b\mathsf{T}}\phi_{j,k,c}+\frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}}\right.\\
&\left.-\frac{1}{2}\hat{w}_{j,k-1,c2}^{b\mathsf{T}}\phi_{j,k-1,c}-\frac{1}{2}w_{j,k-1,c2}^{b\mathsf{T}}\phi_{j,k-1,c}\right\|^2\\
&+\frac{\eta_2}{2}\left\|\varsigma_{j,k-1,c}^b\right\|^2.
\end{aligned}
\tag{7.56}
$$

$\Delta L_{j,2}$ can be derived as

$$\begin{aligned}\Delta L_{j,2} &= \frac{1}{l_j\beta_{j,a}}\mathrm{tr}\left\{\tilde{w}_{j,k+1,a}^{\mathsf{T}}\tilde{w}_{j,k+1,a} - \tilde{w}_{j,k,a}^{\mathsf{T}}\tilde{w}_{j,k,a}\right\}\\ &= \frac{1}{l_j}\mathrm{tr}\left\{-2\varsigma_{j,k,a}\left[\hat{w}_{j,k,c}^{b\mathsf{T}}\Gamma_{j,k}\right]^{\mathsf{T}}\left[\hat{w}_{j,k,c}^{b\mathsf{T}}\phi_{j,k,c}\right]\right.\\ &\quad \left.+\beta_{j,a}\left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\phi_{j,k,c}\right\|^2\|\phi_{j,k,a}\|^2\left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\Gamma_{j,k}\right\|^2\right\}\\ &= \frac{1}{l_j}\left\{-\left(\left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\Gamma_{j,k}\right\|^2 - \beta_{j,a}\|\phi_{j,k,a}\|^2\left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\Gamma_{j,k}\right\|^2\right)\right.\\ &\quad \times\left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\phi_{j,k,c}\right\|^2 - \left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\Gamma_{j,k}\varsigma_{j,k,a}\right\|^2\\ &\quad \left.+\left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\phi_{j,k,c} - \hat{w}_{j,k,c}^{b\mathsf{T}}\Gamma_{j,k}\varsigma_{j,k,a}\right\|^2\right\}, \end{aligned} \tag{7.57}$$

in which $\varsigma_{j,k,a} = \tilde{w}_{j,k,a}^{\mathsf{T}}\phi_{j,k,a}$. As we know that $\Gamma_{j,k} = (1/2)(1-\phi_{j,k,c}^2)w_{j,c1,n+j}$ and $-1 \le \phi(x) \le 1$, based on the Cauchy–Schwarz inequality, (7.57) can be further derived as

$$\begin{aligned}\Delta L_{j,2} &\le \frac{1}{l_j}\left\{-\left(\left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\Gamma_{j,k}\right\|^2 - \beta_{j,a}\|\phi_{j,k,a}\|^2\left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\Gamma_{j,k}\right\|^2\right)\right.\\ &\quad \times\left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\phi_{j,k,c}\right\|^2 - \left\|\hat{w}_{j,k,c}^{b\mathsf{T}}\Gamma_{j,k}\varsigma_{j,k,a}\right\|^2\\ &\quad \left.+4\left(\left\|w_{j,k,c}^{b\mathsf{T}}\phi_{j,k,c}\right\|^2 + \left\|\varsigma_{j,k,c}^{b}\right\|^2\right)\right\}. \end{aligned} \tag{7.58}$$

According to (7.39), $\Delta L_{j,3}$ and $\Delta L_{j,4}$ can be calculated as

$$\Delta L_{j,3} = \frac{1}{2}\eta_1\left(\left\|\varsigma_{j,k,c}^{a}\right\|^2 - \left\|\varsigma_{j,k-1,c}^{a}\right\|^2\right), \tag{7.59}$$

$$\Delta L_{j,4} = \frac{1}{2}\eta_2\left(\left\|\varsigma_{j,k,c}^{b}\right\|^2 - \left\|\varsigma_{j,k-1,c}^{b}\right\|^2\right). \tag{7.60}$$

Note that the four terms of $\Delta L_{j,k}$ have already known, let l_j satisfy

$$\frac{4}{\left(\alpha_j^2\|A\|^2 - 1/2\right)\eta_2} < l_j. \tag{7.61}$$

If α_j, $\beta_{j,c}$ and $\beta_{j,a}$ satisfy (7.36), (7.37) and (7.38) respectively, substituting (7.49), (7.56), (7.59), (7.60) and (7.61) into (7.40), we get

$$\begin{aligned}
\Delta L_{j,k} =& \Delta L_{j,1} + \Delta L_{j,2} + \Delta L_{j,3} + \Delta L_{j,4} \\
\leq & -\eta_1 \left(\alpha_j^2 - \frac{1}{2}\right) \left\|\varsigma_{j,k,c}^a\right\|^2 \\
& - \left(\alpha_j^2 \eta_2 \|A\|^2 - \frac{\eta_2}{2} - \frac{4}{l_j}\right) \left\|\varsigma_{j,k,c}^b\right\|^2 \\
& - \alpha_j^2 \eta_1 \left(1 - \alpha_j^2 \beta_{j,c} \eta_1 \|\phi_{j,k,c}\|^2\right) \\
& \times \left\|\varsigma_{j,k,c}^a + w_{j,c}^{a\mathsf{T}} \phi_{j,k,c} + \alpha_j^{-1} r_j(x_{k-1})\right. \\
& \left. - \alpha_j^{-1} \hat{w}_{j,k-1,c}^{a\mathsf{T}} \phi_{j,k-1,c}\right\|^2 - \alpha_j^2 \eta_2 \\
& \left(1 - \alpha_j^2 \beta_{j,c} \eta_2 \|A\|^2 \|\phi_{j,k,c}\|^2\right) \left\|A^{\mathsf{T}} \varsigma_{j,k,c}^b + \alpha_j^{-1} E\right\|^2 \\
& - \frac{1}{l_j} \left(\left\|\hat{w}_{j,k,c}^{b\mathsf{T}} \Gamma_{j,k}\right\|^2 - \beta_{j,a} \|\phi_{j,k,a}\|^2 \left\|\hat{w}_{j,k,c}^{b\mathsf{T}} \Gamma_{j,k}\right\|^2\right) \\
& \times \left\|\hat{w}_{j,k,c}^{b\mathsf{T}} \phi_{j,k,c}\right\|^2 - \frac{1}{l_j} \left\|\hat{w}_{j,k,c}^{b\mathsf{T}} \Gamma_{j,k} \varsigma_{j,k,a}\right\|^2 + M^2,
\end{aligned} \tag{7.62}$$

in which M^2 is defined as

$$\begin{aligned}
M^2 = & \frac{4}{l_j} \left\|w_{j,k,c}^{b\mathsf{T}} \phi_{j,k,c}\right\|^2 \\
& + 2\eta_2 \left\|\alpha_j A w_{j,c}^{b\mathsf{T}} \phi_{j,k,c} + \frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}}\right. \\
& \left. - \frac{1}{2} \hat{w}_{j,k-1,c2}^{b\mathsf{T}} \phi_{j,k-1,c} - \frac{1}{2} w_{j,k-1,c}^{b\mathsf{T}} \phi_{j,k-1,c}\right\|^2 \\
& + 2\eta_1 \left\|\alpha_j w_{j,c}^{a\mathsf{T}} \phi_{j,k,c} + r_j(x_{k-1})\right. \\
& \left. - \frac{1}{2} \hat{w}_{j,k-1,c}^{a\mathsf{T}} \phi_{j,k-1,c} - \frac{1}{2} w_{j,k-1,c}^{a\mathsf{T}} \phi_{j,k-1,c}\right\|^2.
\end{aligned} \tag{7.63}$$

Applying the Cauchy–Schwarz inequality to (7.63), we have

$$
\begin{aligned}
M^2 \leq & \frac{4}{l_j}\left\|w_{j,k,c}^{b\mathsf{T}}\phi_{j,k,c}\right\|^2 \\
& + 8\eta_2\left(\left\|\alpha_j A w_{j,c}^{b\mathsf{T}}\phi_{j,k,c}\right\|^2 + \left\|\frac{\partial r_j(x_{k-1})}{\partial X_{j,k-1}}\right\|^2\right. \\
& \left. + \left\|\frac{1}{2}\hat{w}_{j,k-1,c2}^{b\mathsf{T}}\phi_{j,k-1,c}\right\|^2 + \left\|\frac{1}{2}w_{j,k-1,c2}^{b\mathsf{T}}\phi_{j,k-1,c}\right\|^2\right) \\
& 8\eta_1\left(\left\|\alpha_j w_{j,c}^{a\mathsf{T}}\phi_{j,k,c}\right\|^2 + \|r_j(x_{k-1})\|^2\right. \\
& \left. + \left\|\frac{1}{2}\hat{w}_{j,k-1,c}^{a\mathsf{T}}\phi_{j,k-1,c}\right\|^2 + \left\|\frac{1}{2}w_{j,k-1,c}^{a\mathsf{T}}\phi_{j,k-1,c}\right\|^2\right) \\
\leq & \frac{4}{l_j}\left(w_{j,c,m}^{b}\phi_{j,c,m}\right)^2 \\
& + 8\eta_2\left[\left(\alpha_j A_m w_{j,c,m}^{b}\phi_{j,c,m}\right)^2 + (r_{j,X,m})^2\right. \\
& \left. + \left(\frac{1}{2}w_{j,c,m}^{b}\phi_{j,c,m}\right)^2 + \left(\frac{1}{2}w_{j,c,m}^{b}\phi_{j,c,m}\right)^2\right] \\
& + 8\eta_1\left[\left(\alpha_j w_{j,c,m}^{a}\phi_{j,c,m}\right)^2 + (r_{j,m})^2\right. \\
& \left. + \left(\frac{1}{2}w_{j,c,m}^{a}\phi_{j,c,m}\right)^2 + \left(\frac{1}{2}w_{j,c,m}^{a}\phi_{j,c,m}\right)^2\right] \\
= & \left(\frac{4}{l_j} + 4\eta_2\right)\left(w_{j,c,m}^{b}\phi_{j,c,m}\right)^2 + 4\eta_1\left(w_{j,c,m}^{a}\phi_{j,c,m}\right)^2 \\
& + 8\eta_2[\left(\alpha_j A_m w_{j,c,m}^{b}\phi_{j,c,m}\right)^2 + (r_{j,X,m})^2] \\
& + 8\eta_1\left[\left(\alpha_j w_{j,c,m}^{a}\phi_{j,c,m}\right)^2 + (r_{j,m})^2\right] = M_m^2,
\end{aligned}
\tag{7.64}
$$

in which $w_{j,c,m}^{a}$, $w_{j,c,m}^{b}$, $\phi_{j,c,m}$, $r_{j,m}$, $r_{j,X,m}$ and A_m are the upper bounds of $w_{j,k,c}^{a}$, $w_{j,k,c}^{b}$, $\phi_{j,k,c}$, $r_j(x_k)$, $\partial r_j(x_k)/\partial X_{j,k}$ and A, respectively.

If conditions (7.36), (7.37) and (7.38) hold, then for any

$$\left\|\varsigma_{j,k,c}^{a}\right\| > \sqrt{M_m^2 \Big/ \left(\alpha_j^2 - \frac{1}{2}\right)}, \tag{7.65}$$

the first difference $\Delta L_{j,k} \leq 0$ and then $\Delta L_{S,k} = \sum_{j=1}^{N} \Delta L_{j,k} \leq 0$. Therefore, according to the Lyapunov extension theorem, this demonstrates that the errors between the optimal network weights $w_{j,c}$, $w_{j,a}$ and their respective estimations $\hat{w}_{j,k,c}$, $\hat{w}_{j,k,a}$ are UUB. □

Remark 7.4. The activation function $\phi(x)$ with the form of (7.14) employed by critic and action networks is always bounded within $[-1, 1]$. According to (7.15) and (7.25), the optimal outputs of critic and action networks can be derived as

$$\begin{bmatrix} V_j^*(x_k) \\ \lambda_j(x_k) \end{bmatrix} = w_{j,k,c}^{\mathsf{T}} \phi_{j,k,c}, \tag{7.66}$$

$$u_j{}^*(x_k) = w_{j,k,a}^{\mathsf{T}} \phi_{j,k,a}. \tag{7.67}$$

If Assumption 7.3 holds, then the estimation errors of the optimal value function $V_j^*(x_k)$ and control policy $u_j{}^*(x_k)$ are also UUB.

Remark 7.5. In three-layer neural network, if the activation function is chose as (7.14) with $-1 \leq \phi(x) \leq -1$, then we can get

$$\|\phi(x)\|^2 \leq N_h, \tag{7.68}$$

where N_h is the numbers of hidden nodes in the neural network. Based on the above theory, conditions (7.37) and (7.38) for the critic and action networks can be further evolved as

$$\beta_{j,c} < 1/\left(\alpha_j^2 N_{j,h,c}\right), \quad j = 1, 2, \ldots N \tag{7.69}$$

and

$$\beta_{j,a} < 1/N_{j,h,a}, \quad j = 1, 2, \ldots N, \tag{7.70}$$

in which $N_{j,h,c}$ and $N_{j,h,a}$ are the the numbers of hidden nodes in the critic and action networks. We can see that (7.69) and (7.70) are simple and intuitive sufficient conditions, which can be used to guide the selection of learning rates.

7.5 Simulation Example

Consider the following nonlinear system:

$$x_{k+1} = f(x_k) + g_{1,k}u_{1,k} + g_{2,k}u_{2,k}, \tag{7.71}$$

where the system state $x_k \in \mathbb{R}^2$, control policies $u_{j,k} \in \mathbb{R}$, $j = 1, 2$. Define $x_k = [x_k(1), x_k(2)]^{\mathsf{T}}$. The exact form of $f(x_k)$ and $g_{j,k}$, $j = 1, 2$, are defined as

$$f(x_k) = \begin{bmatrix} x_k(2) \\ \{-x_k(2) - 0.5x_k(1) \\ +0.25x_k(2)(\cos(2x_k(1)) + 2)^2 \\ +0.25x_k(2)(\sin(2x_k(1)) + 2)^2\} \end{bmatrix}, \tag{7.72}$$

$$g_{1,k} = \begin{bmatrix} 0 \\ \cos(2x_k(1)) + 2 \end{bmatrix}, \tag{7.73}$$

and

$$g_{2,k} = \begin{bmatrix} 0 \\ \sin\left(4x_k^2(1)\right) + 2 \end{bmatrix}, \tag{7.74}$$

respectively. In this simulation, the performance index function of each player is defined as (7.2) with $Q_1 = 2$, $R_{11} = 2$, $R_{12} = 2$, $Q_2 = 1$, $R_{21} = 1$ and $R_{22} = 1$. Above matrices are set according to the practical design purpose of how to balance the state-related utility and the control-related utility.

In the implementation of MF-CGDHP algorithm, three-layer feedforward neural networks are employed. The structures of action and critic networks are as 2–5–1 (i.e. two neuron in the input layer, five neurons in the hidden layer and one neuron in the output layer) and 4–8–5 for each player. The sigmoid function (7.14) is selected as the activation function for both critic and action networks. In the learning of the MF-CGDHP algorithm, the initial weights for all neural networks are chosen randomly within $[-0.5, 0.5]$. According to Remark 7.1, only the weights between hidden layer and output layer are trained. The weights updating rules of all critic and

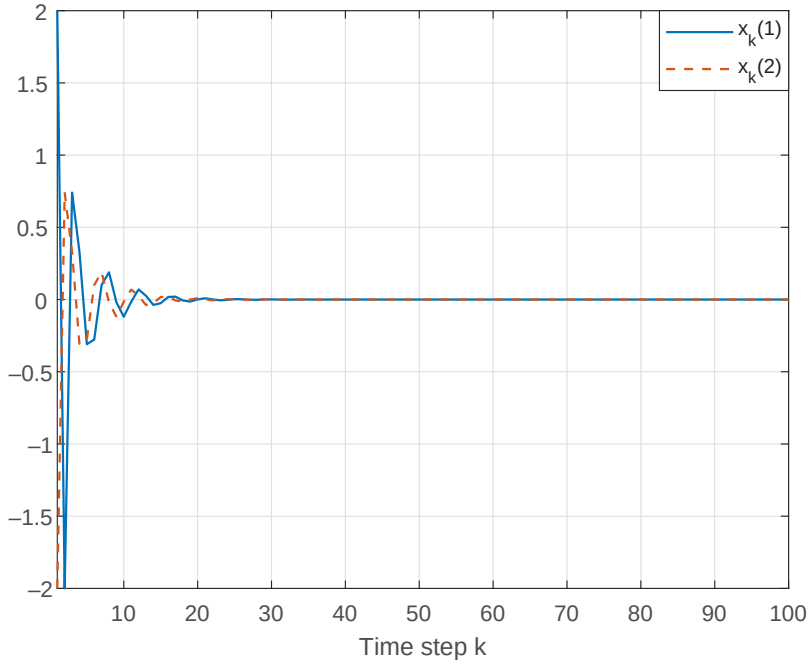

Fig. 7.2. Trajectories of the system state x_k.

action networks can be seen in (7.21) and (7.27) with $\eta_1 = \eta_2 = 0.5$. Considering the conditions in Theorem 7.1, we choose the value of discount factor as $\alpha_1 = \alpha_2 = 0.9 \in \left(1/\sqrt{2}, 1\right)$. According to Remark 7.5 and the structures of neural networks, the learning rates are set as $\beta_{1,c} = \beta_{2,c} = 0.01 < 1/\left(8 \times 0.9^2\right) \approx 0.154$, $\beta_{1,a} = \beta_{2,a} = 0.01 < 1/5 = 0.2$. The numbers of internal cycles are defined as $N_{j,c} = N_{j,a} = 100, j = 1, 2$, the error threshold is selected as $\varepsilon_{j,c} = \varepsilon_{j,a} = 10^{-6}$, $j = 1, 2$. The number of the maximal training cycle number is set as $N_{cyc} = 200$ in the simulation.

The MF-CGDHP method is used in the learning process at the time step $k = 0$ with the initial state $x_0 = [-0.5, -0.5]$. In the learning process of the weights for all critic and action networks, the MF-CGDHP method takes 0.0753s. After that, we apply the obtained weights of critic and action networks to the system (7.71). The MF-CGDHP method takes 0.0031s for the rest simulation. The trajectories of system state x_k are presented in Fig. 7.2. The trajectories of the control laws for two players are shown in Fig. 7.3.

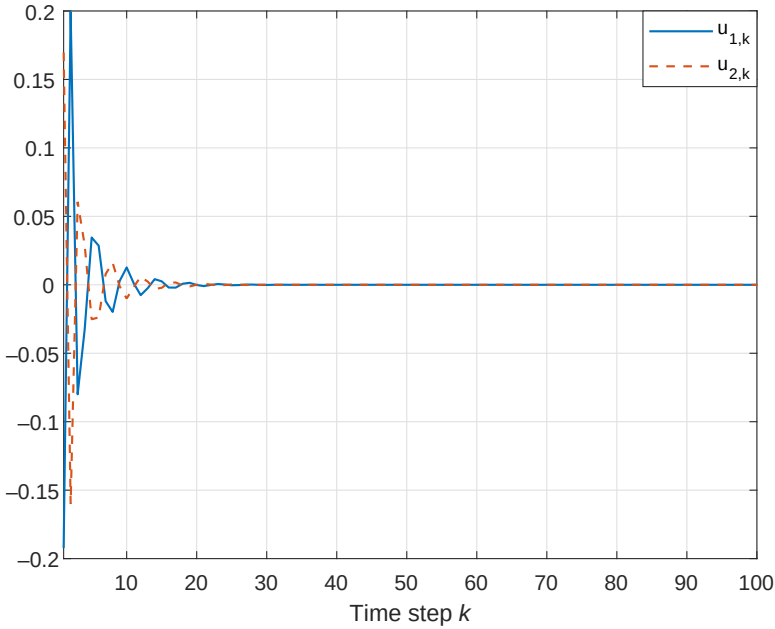

Fig. 7.3. Trajectories of the control input $u_{j,k}$ for player $j, j = 1, 2$.

Observing Figs. 7.2 and 7.3, one can find that the state trajectories run to zero under the obtained control policies.

7.6 Conclusion

In this chapter, an online adaptive optimal learning algorithm is developed to solve the MP-NZSG problem of unknown nonlinear systems. The MF-CGDHP structure is used to implement the approximate method which does not require any model network. For each player, a critic network is used to approximate its value function and partial derivatives. An action network is employed to approximate its optimal control policy. Based on the updating rules of weights in the critic and action networks, we have proved that the approximation errors are UUB for all neural networks when some conditions are satisfied. To demonstrate the effectiveness of proposed results, simulation results have been shown.

References

[1] Vamvoudakis, K.G. and Lewis, F.L. (2011). Multi-player non-zero-sum games: Online adaptive learning solution of coupled HamiltonCJacobi equations. *Automatica* 47(8), 1556C-1569.

[2] Ni, Z., He, H., Zhong, X., and Danil, V.P. (2015). Model-free dual heuristic dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems* 26(8), 1834–1839.

[3] Zhong, X., He, H., Wang, D. and Ni, Z. (2018). Model-free adaptive control for unknown nonlinear zero-sum differential game. *IEEE Transactions on Cybernetics* 48(5), 1633–1646.

Chapter 8

Continuous-Time Distributed Policy Iteration for Multi-Controller Nonlinear Systems

8.1 Introduction

In this chapter, a novel distributed policy iteration algorithm is established for the infinite horizon optimal control problems of continuous-time nonlinear systems. In each iteration of the developed distributed policy iteration algorithm, only one controller's control law is updated and other controllers remain unchanged. The main contribution of the present algorithm is to improve the iterative control law one by one, instead of updating all the control laws in each iteration of the traditional policy iteration algorithms, which effectively releases the computation burden in each iteration. The properties of distributed policy iteration algorithm for continuous-time nonlinear systems are analyzed. Admissibility of the present methods has also been analyzed. Besides, monotonicity, convergence, and optimality have been discussed, which show that the iterative value function is non-increasingly convergent to the solution of the Hamilton–Jacobi–Bellman equation. Finally, numerical simulations are conducted to illustrate the effectiveness of the proposed method.

8.2 Problem Formulations

Consider the following continuous-time multi-controller nonlinear system

$$\dot{x} = F(x, u_1, \ldots, u_N), \tag{8.1}$$

where the system state is denoted by $x = x(t) \in \mathbb{R}^n$, and $u_i = u_i(t) \in \mathbb{R}^{m_i}$, $i = 1, 2, \ldots, N$, represents the control inputs. $F(\cdot)$ is regarded as the system function. N stands for the number of the controllers, which is usually positive integer. Let x_0 be the initial condition of the nonlinear system. Some assumptions are given in the following for further analysis.

Assumption 8.1. The system (8.1) is controllable, and the system states all belong to a compact set where the origin is contained; the system function $F(x, u_1, \ldots, u_n)$ is Lipschitz continuous for x and u_i, $i = 1, 2, \ldots, N$; the equilibrium point of the system (8.1) is $x = 0$, when the control inputs satisfy $u = 0$, i.e., $F(0, 0, \ldots, 0) = 0$; the multi-control law $u_i(x)$, $i = 1, 2, \ldots, N$, is continuous on Ω and $u_i = u_i(x) = 0$ always holds for $x = 0$.

To analyse the optimal control problem of system (8.1), the performance index function is given with the following definition:

$$J(x) = \int_t^\infty U(x(s), u_1(s), \ldots, u_N(s))\mathrm{d}s, \tag{8.2}$$

where $U(x, u_1, \ldots, u_N)$ represents the utility function and positive definite for x and u_i, $i = 1, 2, \ldots, N$.

The admissible control laws of multi-controllers can be defined as $\mu_i \in \Psi(\Omega)$, $i = 1, 2, \ldots, N$, and $\Psi(\Omega)$ can be considered as the set of admissible controls on Ω. Under the admissible controls, the value function is given as

$$V(x) = \int_t^\infty U(x(s), \mu_1(x(s)), \mu_2(x(s)), \ldots, \mu_N(x(s)))\mathrm{d}s. \tag{8.3}$$

If the value function is continuously differentiable respect to t, it can be transformed into the following form which is called nonlinear

Lyapunov equation

$$U(x,\mu_1,\ldots,\mu_N)+\left(\frac{\partial V(x)}{\partial x}\right)^{\mathsf{T}}F(x,\mu_1,\ldots,\mu_N)=0. \tag{8.4}$$

Based on the definition in (8.2), the optimal performance index function is defined as

$$J^*(x)=\min_{\mu_1,\ldots,\mu_N\in\Psi(\Omega)}\left\{\int_t^\infty U(x(s),\mu_1(s),\ldots,\mu_N(s))\mathrm{d}s\right\}. \tag{8.5}$$

which can satisfy the HJB equation with $J^*(0)=0$, and the following equation can be derived as

$$\begin{aligned}&\min_{\mu_1,\ldots,\mu_N}\left\{U(x,\mu_1,\ldots,\mu_N)+\left(\frac{\partial J^*(x)}{\partial x}\right)^{\mathsf{T}}F(x,\mu_1,\ldots,\mu_N)\right\}\\&=U(x,\mu_1^*(x),\ldots,\mu_N^*(x))+\left(\frac{\partial J^*(x)}{\partial x}\right)^{\mathsf{T}}\times F(x,\mu_1^*(x),\ldots,\mu_N^*(x))\\&=0,\end{aligned} \tag{8.6}$$

where $\mu_1^*(x),\ldots\mu_N^*(x)$ are the optimal control laws. Generally, it is almost impossible to get $J^*(x)$ by directly solving the HJB equations, especially for multi-controller nonlinear systems. Hence, developing a novel distributed policy iteration algorithm to overcome this difficulty is very necessary.

8.3 Continuous-Time Distributed Policy Iteration: Derivations and Properties

In this section, the derivations of the continuous-time distributed policy iteration algorithm for multi-controller nonlinear systems will be discussed. Furthermore, some new methods to analyse the convergence and monotonicity will be introduced and the admissibility of the multi-control laws will also be proven.

8.3.1 *Derivations of the continuous-time distributed policy iteration algorithm*

Let $v_1^0(x), \ldots, v_N^0(x)$, $\forall x \in \mathbb{R}^n$ be arbitrary admissible control laws. Let $V^0(x)$ denote the initial iterative value function, such that

$$U(x, v_1^0(x), \ldots, v_N^0(x)) + \left(\frac{\partial V^0(x)}{\partial x}\right)^{\mathsf{T}} \times F(x, v_1^0(x), \ldots, v_N^0(x)) = 0. \tag{8.7}$$

Let $k = 1$ and $\tau_1 \in \mathcal{N}$, $\mathcal{N} = \{1, 2, \ldots, N\}$. Let $u_{(i)} = \{u_j : j \in \mathcal{N}, j \neq i\}$. Then, we have $U(x, u_{\tau_1}, u_{(\tau_1)}) = U(x, u_1, \ldots, u_N)$. For $k = 1$, the control law $v_{\tau_1}^1(x)$ can be calculated by

$$v_{\tau_1}^1(x) = \arg\min_{u_{\tau_1}} \left\{ U(x, u_{\tau_1}, v_{(\tau_1)}^0(x)) + \left(\frac{\partial V^0(x)}{\partial x}\right)^{\mathsf{T}} F(x, u_{\tau_1}, v_{(\tau_1)}^0(x)) \right\}. \tag{8.8}$$

Let $v_j^1(x) = v_j^0(x)$, for all $j \in \mathcal{N}$ and $j \neq \tau_1$. According to $v_1^1(x), v_2^1(x), \ldots, v_N^1(x)$, the corresponding value function $V^1(x)$ is calculated by

$$U(x, v_1^1(x), \ldots, v_N^1(x)) + \left(\frac{\partial V^1(x)}{\partial x}\right)^{\mathsf{T}} \times F(x, v_1^1(x), \ldots, v_N^1(x)) = 0. \tag{8.9}$$

For $k = 1, 2, \ldots$ let $\tau_k \in \mathcal{N}$ and $U(x, u_{\tau_k}, u_{(\tau_k)}) = U(x, u_1, \ldots, u_N)$, the iterative control law $v_{\tau_k}^k(x)$ can be derived by

$$v_{\tau_k}^k(x) = \arg\min_{u_{\tau_k}} \left\{ U(x, u_{\tau_k}, v_{(\tau_k)}^{k-1}(x)) + \left(\frac{\partial V^{k-1}(x)}{\partial x}\right)^{\mathsf{T}} F(x, u_{\tau_k}, v_{(\tau_k)}^{k-1}(x)) \right\}. \tag{8.10}$$

Let $v_j^k(x) = v_j^{k-1}(x)$, for all $j \in \mathcal{N}$ and $j \neq \tau_k$. According to $v_1^k(x), v_2^k(x), \ldots, v_N^k(x)$, the iterative value function $V^k(x)$ is

updated by

$$U(x, v_1^k(x), \ldots, v_N^k(x)) + \left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} \times F(x, v_1^k(x), \ldots, v_N^k(x)) = 0. \tag{8.11}$$

Then we can obtain distributed policy iteration algorithm as Algorithm 8.1.

Algorithm 8.1 Distributed policy iteration algorithm for multi-controller nonlinear systems

Require:

Choose randomly an admissible control law $v_i^0(x), i = 1, \ldots, N$;
Choose a computation precision ε.

Ensure:

1: Let the iteration index $k = 0$. Construct an iterative value function $V^0(x)$ to satisfy (8.7);
2: Let $k = k + 1$, choose $\tau_k \in \mathcal{N}$ randomly. Do **Policy Improvement**

$$v_{\tau_k}^k(x) = \arg\min_{u_{\tau_k}} \left\{ U\big(x, u_{\tau_k}, v_{(\tau_k)}^{k-1}(x)\big) + \left(\frac{\partial V^{k-1}(x)}{\partial x}\right)^{\mathsf{T}} F\big(x, u_{\tau_k}, v_{(\tau_k)}^{k-1}(x)\big) \right\};$$

3: Do **Policy Evaluation**

$$U(x, v_1^k(x), \ldots, v_N^k(x)) + \left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} \times F(x, v_1^k(x), \ldots, v_N^k(x)) = 0;$$

4: If $V^{k-1}(x) - V^k(x) > \varepsilon$, goto Step 2.
5: **return** $v_1^k(x), \ldots, v_N^k(x)$, $V^k(x)$.

In this chapter, the function $J^*(x)$ denotes the optimal performance index function under the optimal control laws $u_1^*(x)$, $u_2^*(x), \ldots, u_N^*(x)$. For $k = 0, 1, \ldots$, the function $V^k(x)$ is used in the iteration process, which denotes the iterative value function under

the iterative control laws $\nu_1^k(x), \nu_2^k(x), \ldots, \nu_N^k(x)$. In the following, the properties of $V^k(x)$ will be analyzed and the relationship between $V^k(x)$ and $J^*(x)$ will be proven.

8.3.2 *Property analysis*

In this subsection, the corresponding properties such as convergence and admissibility of the distributed policy iteration algorithm are analysed. For traditional policy iteration algorithms [1–6], all the control laws of the system must be updated in each iteration simultaneously to guarantee the convergence of $V^k(x)$ and the admissibility of the control laws. However, for distributed policy iteration algorithm (8.7)–(8.11), only one control input is updated such that the traditional analysis methods are unavailable for distributed policy iteration algorithm. Thus, some novel analysis methods will be established in this subsection. First, the admissibility of the distributed iterative control laws will be analyzed and Some lemmas are given in the following.

Lemma 8.1. *If $v_1^k(x), \ldots, v_N^k(x)$, $k = 0, 1, \ldots$, are admissible control laws for system* (8.1)*, there exists a value function $V^k(x)$ to satisfy*

$$U(x, v_1^k(x), \ldots, v_N^k(x)) + \left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} \times F(x, v_1^k(x), \ldots, v_N^k(x)) = 0. \tag{8.12}$$

Theorem 8.1. *For $k = 0, 1, \ldots$, the iterative value function $V^k(x)$ and the distributed iterative control laws $v_1^k(x), \ldots, v_N^k(x)$ can be obtained by* (8.7)–(8.11)*. If control laws $v_1^k(x), \ldots, v_N^k$ are admissible for nonlinear system* (8.1)*, then $v_1^{k+1}(x), \ldots, v_N^{k+1}$ are admissible control laws.*

Proof. For $k = 0, 1, \ldots$, as $v_1^k(x), \ldots, v_N^k$ are admissible control laws, (8.12) is always satisfied based on Lemma 8.1. Letting

$\tau_{k+1} \in \mathcal{N}$, $v_{\tau_{k+1}}^{k+1}(x)$ can be obtained by (8.10), which is expressed as

$$v_{\tau_{k+1}}^{k+1}(x) = \arg\min_{u_{\tau_{k+1}}} \left\{ U\big(x, u_{\tau_{k+1}}, v_{(\tau_{k+1})}^{k}(x)\big) + \left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} F\big(x, u_{\tau_{k+1}}, v_{(\tau_{k+1})}^{k}(x)\big) \right\}. \tag{8.13}$$

According to (8.12), it can be derived that

$$\left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} F(x, v_1^{k+1}(x), \ldots, v_N^{k+1}(x)) + U(x, v_1^{k+1}(x), \ldots, v_N^{k+1}(x)) \leq 0, \tag{8.14}$$

where $v_{\tau_{k+1}}^{k+1}(x)$ is obtained by (8.13) and $v_j^{k+1}(x) = v_j^k(x)$, for all $j \in \mathcal{N}$ and $j \neq \tau_{k+1}$.

According to Assumption 8.1, it can be derived that $V^k(x)$ is always positive due to the characteristics of the utility function. Then $V^k(x)$ is said to be a positive definite function. Choose $V^k(x)$ as the Lyapunov function candidate. Based on (8.14), we have the following inequality by calculating the derivative of $V^k(x)$ along $(v_1^{k+1}(x), \ldots, v_N^{k+1}(x))$, which is expressed as

$$\begin{aligned} \dot{V}^k(x) &= \left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} F(x, v_1^{k+1}(x), \ldots, v_N^{k+1}(x)) \\ &\leq -U(x, v_1^{k+1}(x), \ldots, v_N^{k+1}(x)). \end{aligned} \tag{8.15}$$

Thus, $v_1^{k+1}(x), \ldots, v_N^{k+1}(x)$ are stable control laws for system (8.1). Then, it can be derived that $\lim_{t\to\infty} U(x, v_1^{k+1}(x), \ldots, v_N^{k+1}(x)) = 0$.

Let $\Upsilon^{k+1}(x)$, $k = 0, 1, \ldots$, is a value function, such that

$$\Upsilon^{k+1}(x) = \int_t^{\infty} U(x(s), v_1^{k+1}(x(s)), \ldots, v_N^{k+1}(x(s)))\mathrm{d}s. \tag{8.16}$$

Next, we will prove that $\Upsilon^{k+1}(x)$, $\forall x \in \mathbb{R}^n$, is finite under the control laws $v_1^{k+1}(x), \ldots, v_N^{k+1}(x)$. Taking the derivative of $\Upsilon^{k+1}(x)$ along

time t, we have

$$\dot{\Upsilon}^{k+1}(x) = -U(x, v_1^{k+1}(x), \ldots, v_N^{k+1}(x)). \tag{8.17}$$

Considering (8.15) and (8.17), we can obtain

$$\dot{V}^k(x) \leq \dot{\Upsilon}^{k+1}(x), \quad \forall x \in \mathbb{R}^n. \tag{8.18}$$

As $v_1^k(x), \ldots, v_N^k(x)$ are admissible control laws, we define $V^k(x(\infty)) = \lim_{t\to\infty} V^k(x(t)) = 0$. From (8.16), we know that $\Upsilon^{k+1}(x(\infty)) = \lim_{t\to\infty} \Upsilon^{k+1}(x(t)) = 0$. According to (8.12) and (8.16), we can derive

$$\begin{aligned}\int_t^\infty \frac{\mathrm{d}V^k(x(s))}{\mathrm{d}s}\mathrm{d}s &= V^k(x(\infty)) - V^k(x(t)) \\ &\leq \int_t^\infty \frac{d\Upsilon^{k+1}(x(s))}{\mathrm{d}s}\mathrm{d}s \\ &= \Upsilon^{k+1}(x(\infty)) - \Upsilon^{k+1}(x(t)) \\ &= -\int_t^\infty U(x(s), v_1^{k+1}(x(s)), \ldots, v_N^{k+1}(x(s)))\mathrm{d}s.\end{aligned} \tag{8.19}$$

Then, we can get

$$\int_t^\infty U(x(s), v_1^{k+1}(x(s)), \ldots, v_N^{k+1}(x(s)))\mathrm{d}s \leq V^k(x(t)), \tag{8.20}$$

which shows that the distributed control laws $v_1^{k+1}(x), \ldots, v_N^{k+1}(x)$ are admissible for the system (8.1). The proof is complete. □

Furthermore, the properties for the iterative value function $V^k(x)$ will be discussed in the following.

Theorem 8.2. *For $k = 0, 1, \ldots$, let the iterative value function $V^k(x)$ and the distributed iterative control laws $v_1^k(x), \ldots, v_N^k(x)$ be obtained by (8.7)–(8.11). If $v_1^0(x), \ldots, v_N^0(x)$ are admissible control laws, the iterative value function $V^k(x)$, $k = 0, 1, \ldots$ is monotonically non-increasing as k increases, i.e.,*

$$V^{k+1}(x) \leq V^k(x), \quad \forall x \in \mathbb{R}^n. \tag{8.21}$$

Proof. Consider $k = 0$. For the admissible control laws $v_1^0(x), \ldots, v_N^0(x)$, according to Theorem 8.1, the iterative control laws $v_1^1(x), \ldots, v_N^1(x)$ are admissible control laws. According to (8.16), it can be easily derived that $V^1(x) = \Upsilon^1(x)$. Considering the derivative of $V^0(x)$ along $v_1^1(x), \ldots, v_N^1(x)$, according to (8.14), we can obtain

$$\begin{aligned}\dot{V}^0(x) &= \left(\frac{\partial V^0(x)}{\partial x}\right)^{\mathsf{T}} F(x, v_1^1(x), \ldots, v_N^1(x)) \\ &\leq -U(x, v_1^1(x), \ldots, v_N^1(x)),\end{aligned} \tag{8.22}$$

such that

$$\dot{V}^0(x) \leq \dot{V}^1(x), \quad \forall x \in \mathbb{R}^n. \tag{8.23}$$

According to Theorem 8.1, as $\nu_1^0(x), \nu_2^0(x), \ldots, \nu_N^0(x)$ are admissible control laws, then $\nu_1^1(x), \nu_2^1(x), \ldots, \nu_N^1(x)$ are admissible. These indicate that $V^0(x(\infty)) = 0$ and $V^1(x(\infty)) = 0$, which imply that $V^0(x(t)) \geq V^1(x(t))$, $\forall x \in \mathbb{R}^n$. By the implementation of mathematical induction, (8.21) can be guaranteed to hold for any $k = 0, 1, \ldots$. The proof is complete. □

According to Theorem 8.2, increasing iterative index k, the iterative value function is monotonically non-increasing. Next, the optimality of the iterative value function will be discussed.

Theorem 8.3. *For $k = 0, 1, \ldots$, the iterative value function $V^k(x)$ and the distributed iterative control laws $v_1^k(x), \ldots, v_N^k(x)$ are derived by* (8.7)–(8.11)*. Then, $V^k(x)$ converges to a suboptimal performance index function as $k \to \infty$.*

Proof. According to Lemma 8.1, we can derive

$$V^k(x) = \int_t^\infty U\big(x(s), v_1^k(x(s)), \ldots, v_N^k(x(s))\big)\mathrm{d}s. \tag{8.24}$$

As the utility function $U(x, u_1, \ldots, u_N)$ is a positive definite function for x and u_i, $i = 1, 2, \ldots, N$, according to Assumption 1, we know that $V^k(x) = 0$ for $x = 0$ and $V^k(x) > 0$ for all $x \neq 0$. Hence, for $k = 0, 1, \ldots$, $V^k(x)$ is a positive definite function for x.

For $k \to \infty$, there must exist a controller which is improved for infinite times. Without loss of generality, controller τ^o, $\tau^o \in \mathcal{N}$, is

assumed to improve for infinite times. Let $\mathcal{K}$ denote a set of iteration indices, which is defined as

$$\mathcal{K} = \{k \,|\, k = 0, 1, \ldots, \tau_k = \tau^o, \tau^o \in \mathcal{N}\}. \tag{8.25}$$

Let $\kappa_j \in \mathcal{K}$, $j = 0, 1, \ldots$. Without loss of generality, let $\kappa_0 < \kappa_1 < \cdots$. According to Theorem 8.2, $V^k(x)$ has been proved to be non-increasing as $k \to \infty$ and have the lower limit, which is defined as $V^\infty(x)$ i.e.,

$$V^\infty(x) = \lim_{k\to\infty} V^k(x). \tag{8.26}$$

By considering (8.9) and (8.11), for $k = 0, 1, \ldots$ and $\Delta T \geq 0$, it can be derived that

$$\begin{aligned} V^k(x) = &\int_t^{t+\Delta T} U(x(s), v_{\tau_k}^k(x(s)), v_{(\tau_k)}^k(x(s)))\mathrm{d}s \\ &+ V^k(x(t+\Delta T)). \end{aligned} \tag{8.27}$$

According to (8.27), for $\kappa_j \in \mathcal{K}$, $j = 0, 1, \ldots$, consider a new iterative value function Γ^{κ_j+1} as

$$\begin{aligned} \Gamma^{\kappa_j+1}(x) = &\int_t^{t+\Delta T} U(x(s), v_{\tau^o}^{\kappa_j+1}(x(s)), v_{(\tau^o)}^{\kappa_j}(x(s)))\mathrm{d}s \\ &+ V^{\kappa_j}(x(t+\Delta T)) \\ = &\int_t^{t+\Delta T} U(x(s), v_{\tau^o}^{\kappa_j+1}(x(s)), v_{(\tau^o)}^{\kappa_j+1}(x(s)))\mathrm{d}s \\ &+ V^{\kappa_j}(x(t+\Delta T)), \end{aligned} \tag{8.28}$$

where $v_{\tau^o}^{\kappa_j+1}(x)$ is defined by (8.10) for $k = \kappa_j$ and $v_{(\tau^o)}^{\kappa_j+1}(x) = v_{(\tau^o)}^{\kappa_j}(x)$. According to Theorem 8.2, the following conclusion can be drawn

$$V^{\kappa_j+1}(x) \leq \Gamma^{\kappa_j+1}(x). \tag{8.29}$$

If $k \to \infty$, it is obvious that $j \to \infty$ and $\kappa_j \to \infty$. Then, for $k \to \infty$, we have

$$V^{\infty}(x) \leq \int_t^{t+\Delta T} U(x(s), v_{\tau^o}^{\infty}(x(s)), v_{(\tau^o)}^{\infty}(x(s)))\mathrm{d}s + V^{\infty}(x(t+\Delta T)). \tag{8.30}$$

Define ε as a positive constant, i.e., $\varepsilon > 0$. Due to

$$\lim_{j\to\infty} V^{\kappa_j}(x) = \lim_{k\to\infty} V^{k}(x) = V^{\infty}(x), \tag{8.31}$$

there must be a positive integer $\kappa_\rho < \infty$, such that

$$V^{\kappa_\rho}(x) - \varepsilon \leq V^{\infty}(x) \leq V^{\kappa_\rho}(x). \tag{8.32}$$

Based on (8.27) and (8.32), we have

$$\begin{aligned} V^{\infty}(x) &\geq \int_t^{t+\Delta T} U(x(s), v_{\tau^o}^{\kappa_\rho}(x(s)), v_{(\tau^o)}^{\kappa_\rho}(x(s)))\mathrm{d}s \\ &\quad + V^{\kappa_\rho}(x(t+\Delta T)) - \varepsilon \\ &\geq \int_t^{t+\Delta T} U(x(s), v_{\tau^o}^{\kappa_\rho}(x(s)), v_{(\tau^o)}^{\kappa_\rho}(x(s)))\mathrm{d}s \\ &\quad + V^{\infty}(x(t+\Delta T)) - \varepsilon \\ &= \int_t^{t+\Delta T} U(x(s), v_{\tau^o}^{\kappa_\rho}(x(s)), v_{(\tau^o)}^{\infty}(x(s)))\mathrm{d}s \\ &\quad + V^{\infty}(x(t+\Delta T)) - \varepsilon. \end{aligned} \tag{8.33}$$

As ε is arbitrary, we have

$$V^{\infty}(x) \geq \int_t^{t+\Delta T} U(x(s), v_{\tau^o}^{\kappa_\rho}(x(s)), v_{(\tau^o)}^{\infty}(x(s)))\mathrm{d}s + V^{\infty}(x(t+\Delta T)). \tag{8.34}$$

According to (8.10), we define $v_{\tau^o}^{\infty}$ as

$$v_{\tau^o}^{\infty}(x) = \arg\min_{u_{\tau^o}} \left\{ U(x, u_{\tau^o}, v_{(\tau^o)}^{\infty}(x)) + \left(\frac{\partial V^{\infty}(x)}{\partial x}\right)^{\mathsf{T}} F(x, u_{\tau^o}, v_{(\tau^o)}^{\infty}(x)) \right\}. \tag{8.35}$$

According to (8.34) and (8.35), we have

$$V^{\infty}(x) \geq \int_{t}^{t+\Delta T} U(x(s), v_{\tau^o}^{\infty}(x(s)), v_{(\tau^o)}^{\infty}(x(s)))\mathrm{d}s + V^{\infty}(x(t+\Delta T)). \tag{8.36}$$

Combining (8.30) and (8.36), we can obtain

$$V^{\infty}(x) = \int_{t}^{t+\Delta T} U(x(s), v_{\tau^o}^{\infty}(x(s)), v_{(\tau^o)}^{\infty}(x(s)))\mathrm{d}s + V^{\infty}(x(t+\Delta T)). \tag{8.37}$$

As $\nu_1^k(x), \nu_2^k(x), \ldots, \nu_N^k(x),\ k = 0, 1, \ldots,$ are admissible control laws, which indicates $V^{\infty}(x(k+\Delta T)) = 0$ as $\Delta T \to \infty$. According to (8.35), the following equation of optimality can be derived with $\Delta T \to \infty$,

$$\begin{aligned} 0 =& U(x, v_{\tau^o}^{\infty}(x), v_{(\tau^o)}^{\infty}(x)) + \left(\frac{\partial V^{\infty}(x)}{\partial x}\right)^{\mathsf{T}} \\ & \times F(x, v_{\tau^o}^{\infty}(x), v_{(\tau^o)}^{\infty}(x)) \\ =& \min_{u_{\tau^o}} \{U(x, u_{\tau^o}, v_{(\tau^o)}^{\infty}(x)) + \left(\frac{\partial V^{\infty}(x)}{\partial x}\right)^{\mathsf{T}} \\ & \times F(x, u_{\tau^o}, v_{(\tau^o)}^{\infty}(x))\}. \end{aligned} \tag{8.38}$$

Thus, as $k \to \infty$, $V^k(x)$ converges to a suboptimal performance index function. The proof is complete. □

Remark 8.1. It shows in Theorem 8.3 that as $k \to \infty$, $V^{\infty}(x)$, which is the limit of the iterative value function and defined as $V^{\infty}(x) = \lim_{k\to\infty} V^k(x)$, converges to the solution of the HJB equation in (8.38), not in (8.5). Thus, the suboptimal performance index is actually achieved as $k \to \infty$.

In order to obtain the global convergence analysis, a new criterion is necessary. Before analyzing the global convergence property, some denotations should be defined such as $\mathcal{T}_i = \{k \,|\, \tau_k = i, i \in \mathcal{N}\}$ and π_i, which describes how many elements in $\mathcal{T}_i$.

Theorem 8.4. (Global convergence property). *For $k = 0, 1, \ldots,$ the iterative value function $V^k(x)$ and the distributed iterative control laws $v_1^k(x), \ldots, v_N^k(x)$ are obtained by* (8.7)–(8.11).

If $\pi_i \to \infty$, $\forall\, i \in \mathcal{N}$, *as* $k \to \infty$, $V^k(x)$ *is convergent to the optimal performance index function, i.e.,*

$$\lim_{k\to\infty} V^k(x) = J^*(x). \tag{8.39}$$

Proof. The proof is given by the following two steps.

(1) *Show that the iterative value function* $V^k(x)$ *will satisfies*

$$\lim_{k\to\infty} V^k(x) \geq J^*(x). \tag{8.40}$$

Letting

$$(\nu_1^{k+1}(x), \ldots, \nu_N^{k+1}(x)) = \arg \min_{u_1,u_2,\ldots,u_N} \left\{ U(x, u_1, \ldots, u_N) + \left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} F(x, u_1, \ldots, u_N) \right\}, \tag{8.41}$$

according to (8.7)–(8.11), for any $\tau_k \in \mathcal{N}$, $k = 0, 1, \ldots$, we can derive

$$\begin{aligned}
&\min_{u_1,u_2,\ldots,u_N} \left\{ U(x, u_1, \ldots, u_N) + \left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} F(x, u_1, \ldots, u_N) \right\} \\
&= U(x, \nu_1^{k+1}(x), \ldots, \nu_N^{k+1}(x)) + \left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} F(x, \nu_1^{k+1}(x), \ldots, \nu_N^{k+1}(x)) \\
&\leq \min_{u_{\tau_{k+1}}} \left\{ U(x, u_{\tau_{k+1}}, u^k_{(\tau_{k+1})}(x)) + \left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} F(x, u_{\tau_{k+1}}, u^k_{(\tau_{k+1})}(x)) \right\} \\
&= U(x, v^{k+1}_{\tau_{k+1}}(x), v^k_{(\tau_{k+1})}(x))
\end{aligned}$$

$$
\begin{aligned}
&+\left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} F(x, v^{k+1}_{\tau_{k+1}}(x), v^k_{(\tau_{k+1})}(x)) \\
&\leq 0 \\
&= U(x, u_1^*(x), \ldots, u_N^*(x)) \\
&\quad + \left(\frac{\partial J^*(x)}{\partial x}\right)^{\mathsf{T}} F(x, u_1^*(x), \ldots, u_N^*(x)) \\
&= \min_{u_1, u_2, \ldots, u_N} \Big\{ U(x, u_1, \ldots, u_N) \\
&\quad + \left(\frac{\partial J^*(x)}{\partial x}\right)^{\mathsf{T}} F(x, u_1, \ldots, u_N) \Big\}. \qquad (8.42)
\end{aligned}
$$

and

$$
\begin{aligned}
\dot{V}^k(x) - \dot{J}^*(x) = &\left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} F(\nu_1^{k+1}(x), \ldots, \nu_N^{k+1}(x)) \\
&+ U(\nu_1^{k+1}(x), \ldots, \nu_N^{k+1}(x)) \\
&- \left(\frac{\partial J^*(x)}{\partial x}\right)^{\mathsf{T}} F(\nu_1^{k+1}(x), \ldots, \nu_N^{k+1}(x)) \\
&- U(\nu_1^{k+1}(x), \ldots, \nu_N^{k+1}(x)) \\
= &\min_{u_1, u_2, \ldots, u_N} \Big\{ U(x, u_1, \ldots, u_N) \\
&+ \left(\frac{\partial V^k(x)}{\partial x}\right)^{\mathsf{T}} F(x, u_1, \ldots, u_N) \Big\} \\
&- \min_{u_1, u_2, \ldots, u_N} \Big\{ U(x, u_1, \ldots, u_N) \\
&+ \left(\frac{\partial J^*(x)}{\partial x}\right)^{\mathsf{T}} F(x, u_1, \ldots, u_N) \Big\} \\
&+ \min_{u_1, u_2, \ldots, u_N} \Big\{ U(x, u_1, \ldots, u_N)
\end{aligned}
$$

$$+\left(\frac{\partial J^*(x)}{\partial x}\right)^{\mathsf{T}} F(x,u_1,\ldots,u_N)\Bigg\}$$

$$-\Bigg(\left(\frac{\partial J^*(x)}{\partial x}\right)^{\mathsf{T}} F(\nu_1^{k+1}(x),\ldots,\nu_N^{k+1}(x))$$

$$+U(\nu_1^{k+1}(x),\ldots,\nu_N^{k+1}(x))\Bigg)$$

$$\leq 0. \tag{8.43}$$

Taking derivatives of $V^k(x)$ and $J^*(x)$ along $(\nu_1^{k+1}(x),\ldots,\nu_N^{k+1}(x))$, according to (8.41), we can obtain formula (8.43). Based on (8.43), we know that $\int_t^\infty \dot{V}^k(x(s))\mathrm{d}s \leq \int_t^\infty \dot{J}^*(x(s))\mathrm{d}s$, which is derived as $V^k(x(\infty)) - V^k(x(t)) \leq J^*(x(\infty)) - J^*(x(t))$. According to Theorem 8.1, the iterative control laws $\nu_1^k(x),\ldots,\nu_N^k(x)$ are admissible for system (8.1), which indicates that $V^k(x(\infty)) = 0$. The optimal control laws $u_1^*(x), u_2^*(x),\ldots,u_N^*(x)$ are admissible, which indicates that $J^*(x(\infty)) = 0$. Thus, we can derive $V^k(x) \geq J^*(x)$, $\forall k = 0,1,\ldots$. As $k \to \infty$, (8.40) is always satisfied.

(2) *Show that the iterative value function $V^k(x)$ can satisfy*

$$\lim_{k\to\infty} V^k(x) \leq J^*(x). \tag{8.44}$$

For $\pi_i \to \infty$, it shows that $k \to \infty$. For $k \to \infty$ and $\pi_i \to \infty$, according to Theorem 8.2, define

$$\mathcal{V}^\infty(x) := \lim_{k\to\infty} V^k(x). \tag{8.45}$$

As $\pi_i \to \infty$, $\forall\, i \in \mathcal{N}$, we can derive

$$v_i^\infty(x) = \arg\min_{u_i}\Bigg\{U(x,u_i,v_{(i)}^\infty(x))$$

$$+\ \frac{\partial \mathcal{V}^\infty(x)}{\partial x}F(x,u_i,v_{(i)}^\infty(x))\Bigg\}. \tag{8.46}$$

According to (8.11) and (8.46), we can derive

$$U(x,v_1^\infty(x),\ldots,v_N^\infty(x)) + \frac{\partial \mathcal{V}^\infty(x)}{\partial x}F(x,v_1^\infty(x),\ldots,v_N^\infty(x))$$

$$
\begin{aligned}
&= \min_{u_1}\Big\{U(x,u_1,v_2^\infty(x),\ldots,v_N^\infty(x))\\
&\quad+\frac{\partial\mathcal{V}^\infty(x)}{\partial x}F(x,u_1,v_2^\infty(x),\ldots,v_N^\infty(x))\Big\}\\
&= \min_{u_2}\Big\{\min_{u_1}\Big\{U(x,u_1,u_2,v_3^\infty(x),\ldots,v_N^\infty(x))\\
&\quad+\frac{\partial\mathcal{V}^\infty(x)}{\partial x}F(x,u_1,u_2,v_3^\infty(x),\ldots,v_N^\infty(x))\Big\}\Big\}\\
&= \min_{u_N}\Big\{\cdots\Big\{\min_{u_2}\Big\{\min_{u_1}\{U(x,u_1,\ldots,u_N)\\
&\quad+\frac{\partial\mathcal{V}^\infty(x)}{\partial x}F(x,u_1,\ldots,u_N)\}\Big\}\Big\}\cdots\Big\}\\
&= \min_{u_1,u_2,\ldots,u_N}\Big\{U(x,u_1,\ldots,u_N)\\
&\quad+\frac{\partial\mathcal{V}^\infty(x)}{\partial x}F(x,u_1,\ldots,u_N)\Big\}\\
&= 0. \qquad (8.47)
\end{aligned}
$$

According to (8.47), we have

$$
\begin{aligned}
\mathcal{V}^\infty(x) = &\int_t^{t+\Delta T} U(x(s),v_1^\infty(x(s)),\ldots,v_N^\infty(x(s)))\mathrm{d}s\\
&+\mathcal{V}^\infty(x(t+\Delta T)). \qquad (8.48)
\end{aligned}
$$

Let $\mu_1(x),\ldots,\mu_N(x)$ be admissible control laws for system (8.1). For $k=0,1,\ldots,$ we define

$$
\begin{aligned}
\Phi^{k+1}(x) = &\int_t^{t+\Delta T} U(x(s),\mu_1(x),\ldots,\mu_N(x))\mathrm{d}s\\
&+\Phi^k(x(t+\Delta T)), \qquad (8.49)
\end{aligned}
$$

where $\Phi^0(x)=\mathcal{V}^\infty(x)$. Next, we will prove that

$$
\Phi^{k+1}(x)\geq\mathcal{V}^\infty(x),\quad \forall k=0,1,\ldots. \qquad (8.50)
$$

For $k = 0$, it is easy to get

$$\begin{aligned}\Phi^1(x) &= \int_t^{t+\Delta T} U(x(s), \mu_1(x(s)), \ldots, \mu_N(x(s)))\mathrm{d}s \\ &\quad + \mathcal{V}^\infty(x(t+\Delta T)) \\ &\geq \min_{u_1,u_2,\ldots,u_N} \left\{ \int_t^{t+\Delta T} U(x(s), u_1, \ldots, u_N)\mathrm{d}s \right. \\ &\quad \left. + \ \mathcal{V}^\infty(x(t+\Delta T)) \right\} \\ &= \mathcal{V}^\infty(x). \end{aligned} \tag{8.51}$$

If (8.50) is satisfied for $k = l-1$, $l = 1, 2, \ldots$. When $k = l$, we can obtain

$$\begin{aligned}\Phi^{l+1}(x) &\geq \int_t^{t+\Delta T} U(x(s), \mu_1(x(s)), \ldots, \mu_N(x(s)))\mathrm{d}s \\ &\quad + \mathcal{V}^\infty(x(t+\Delta T)) \\ &\geq \mathcal{V}^\infty(x). \end{aligned} \tag{8.52}$$

By using mathematical induction, the above result (8.50) can be proven.

According to (8.49), we have

$$\begin{aligned}\Phi^{k+1}(x) &= \int_t^{t+(k+1)\Delta T} U(x(s), \mu_1(x(s)), \ldots, \mu_N(x(s)))\mathrm{d}s \\ &\quad + \Phi^0(x(t+(k+1)\Delta T)). \end{aligned} \tag{8.53}$$

As $\mu_1(x), \ldots, \mu_N(x)$ be admissible control laws, based on the result in (8.50), we can obtain

$$\begin{aligned}\lim_{k\to\infty} \Phi^{k+1}(x) &= \int_t^{\infty} U(x(s), \mu_1(x(s)), \ldots, \mu_N(x(s)))\mathrm{d}s \\ &\geq \mathcal{V}^\infty(x). \end{aligned} \tag{8.54}$$

As $\mu_1(x), \ldots, \mu_N(x)$ are arbitrary admissible control laws, the following inequality can be derive

$$\begin{aligned}\mathcal{V}^{\infty}(x) &\leq \min_{\mu_1,\ldots,\mu_N \in \Psi(\Omega)} \left\{ \int_t^{\infty} U(x(s), \mu_1(s), \ldots, \mu_N(s)) \mathrm{d}s \right\} \\ &= J^*(x_k), \end{aligned} \tag{8.55}$$

which proves (8.44). By combining the results in (8.40) and (8.44), (8.39) can be derived. The proof is complete. □

Remark 8.2. Theorem 8.4 indicates that the optimal control law can be obtained obtained by the distributed policy iteration algorithm (8.7)–(8.11). The distributed policy iteration algorithm (8.7)–(8.11) possesses inherent differences from the traditional policy iteration algorithms [1–6]. In each iteration of traditional policy iteration algorithms, all the control laws in the multi-control nonlinear system have to be updated simultaneously. If the dimension of the control is large, the computation burden for traditional policy iteration increases. According to the present distributed policy iteration algorithm (8.7)–(8.11), there is only one control law to update in each iteration, which effectively reduces the computation burden of the policy iteration algorithms. This is an important advantage of the distributed policy iteration. On the other hand, for traditional policy iteration algorithms [1–6], it has been proven that the iterative value function is convergent to the optimal performance index function as the iteration index increases to infinity. However, it is pointed out that if there exist controllers in the distributed policy iteration algorithm, the iterative value function is convergence to a suboptimal performance index function instead of the global optimal one. It is required all the distributed controllers have been improved for infinite times to guarantee the global optimal performance index function. In traditional policy iteration algorithms, the iterative value function is sure to converge the global optimal performance index function, where the sub-optimality will not happen. This is the disadvantage of the distributed policy iteration algorithm.

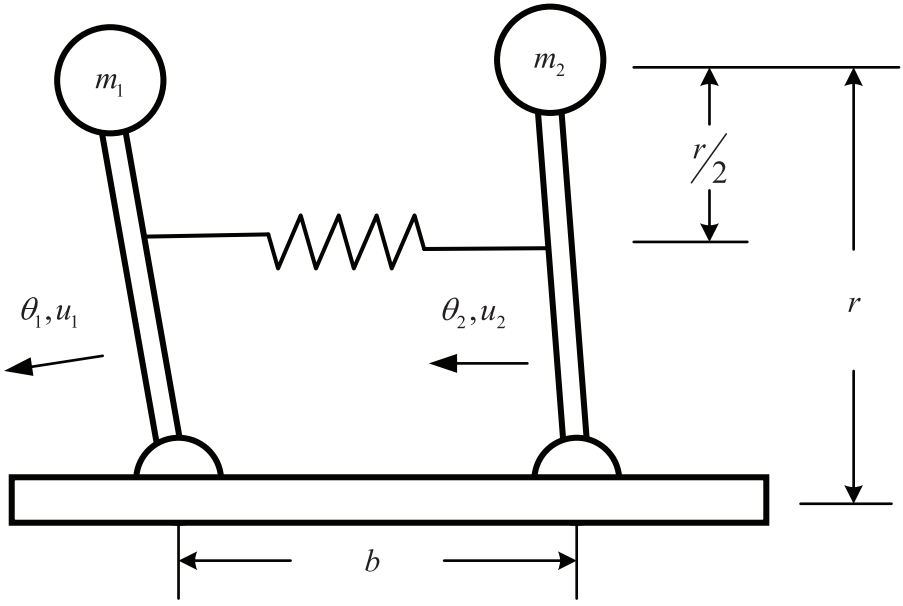

Fig. 8.1. The structure of two inverted pendulums.

8.4 Simulation Example

In this section, a simulation example is conducted with the proposed distributed policy iteration algorithm to illustrate the corresponding performance. The simulation of two inverted pendulums connected by a spring is investigated, whose structure is shown in Fig. 8.1. The dynamics of the two inverted pendulum system can be described as the following equations

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \left(\frac{m_1 gr}{J_1} - \frac{kr^2}{4J_1}\right)\sin(x_1) + \frac{kr}{2J_1}(l-b) \\
&\quad + \frac{u_1}{J_1} + \frac{kr^2}{4J_1}\sin(x_3)
\end{aligned}$$

$$\dot{x}_3 = x_4$$
$$\dot{x}_4 = \left(\frac{m_2 gr}{J_2} - \frac{kr^2}{4J_2}\right)\sin(x_3) + \frac{kr}{2J_2}(l-b) + \frac{u_2}{J_2} + \frac{kr^2}{4J_2}\sin(x_1) \tag{8.56}$$

where $x_{1,1}$ and $x_{2,1}$ represent the angular displacements of the pendulums from vertical. The initial conditions of the systems are $x_0 = [0.3, -0.5, 0.2, 0.1]^{\mathsf{T}}$. m_1 and m_2 denote the masses of the end of two pendulums, and they are considered as $m_1 = 2$ kg and $m_2 = 2.5$ kg in this example. The moments of inertia are denoted by J_1 and J_2, which are adopted as $J_1 = 0.5$ kg·m^2 and $J_2 = 0.625$ kg·m^2 here. The spring constant and natural length of the spring are represented by $k = 100$ N/m and $l = 0.5$ m, respectively. The pendulum height and the distance between the pendulum are defined as $r = 0.5$ m and $b = 0.4$ m. $g = 9.81$ m/s^2 stands for gravitational acceleration.

Define the performance index function as

$$J_1(x) = \int_t^\infty \left(x^{\mathsf{T}} Q x + u_1 R_1 u_1 + u_2 R_2 u_2\right) \mathrm{d}s, \tag{8.57}$$

where Q, R_1, and R_2 represent identity matrices with suitable dimensions.

To implement the proposed distributed policy iteration algorithm, one critic network and two action networks are adopted with BP algorithms, which all have three-layers structure of 4-10-1. Let the learning rate be $\alpha = 0.02$ and the training error be 10^{-5}. In this example, the initial control laws are chosen as $v_1^0(x) = -K_1 x$ and $v_2^0(x) = -K_2 x$, where $K_1 = [8.07,\ 2.13,\ 10.04,\ 1.93]$ and $K_2 = [8.63,\ 1.59,\ 10.69,\ 2.73]$, respectively, which are admissible control laws for the two inverted pendulum systems. $\eta = 0, 1, \ldots$ represents a non-negative integer series and let $\mathcal{N} = \{1, 2\}$. Let $\tau_k = 1$ for $k = 2\eta$ and let $\tau_k = 2$ for $k = 2\eta + 1$. Figure 8.2 shows the trajectory of the iterative value function $V^k(x)$ at $x = x_0$ with 30 iterations by implementing the developed continuous-time distributed policy iteration algorithm. As shown in Fig. 8.2, the iterative value function is monotonically non-increasing as iteration index increases and finally it converges to the optimum, which verifies the validity of theory analyses.

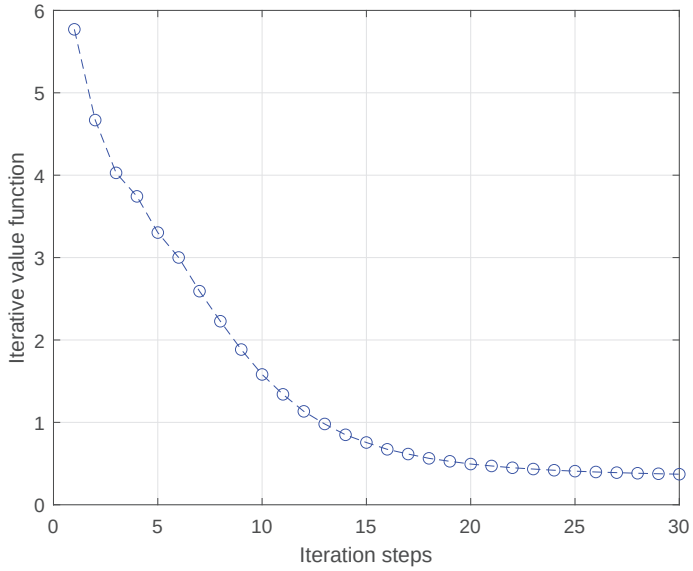

Fig. 8.2. Iterative value function with 30 iterations.

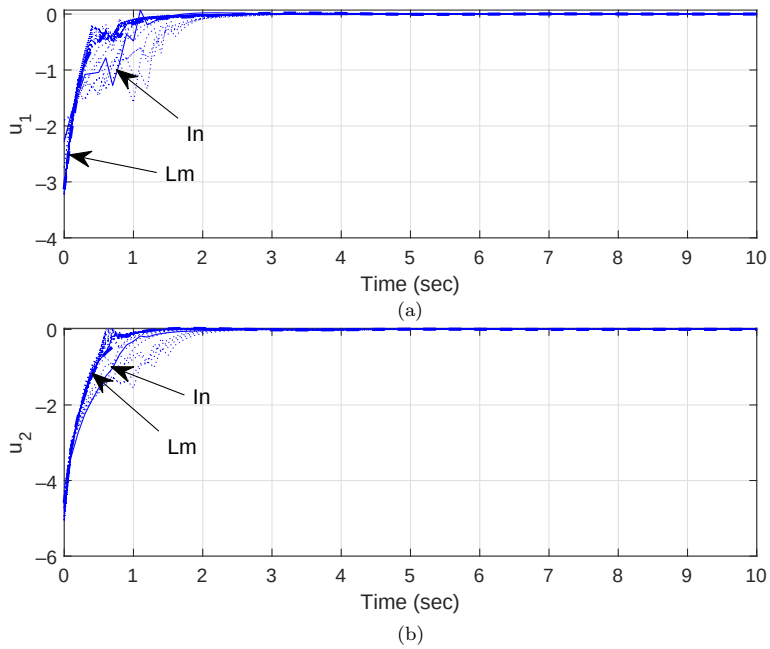

Fig. 8.3. The trajectories of the iterative control laws.

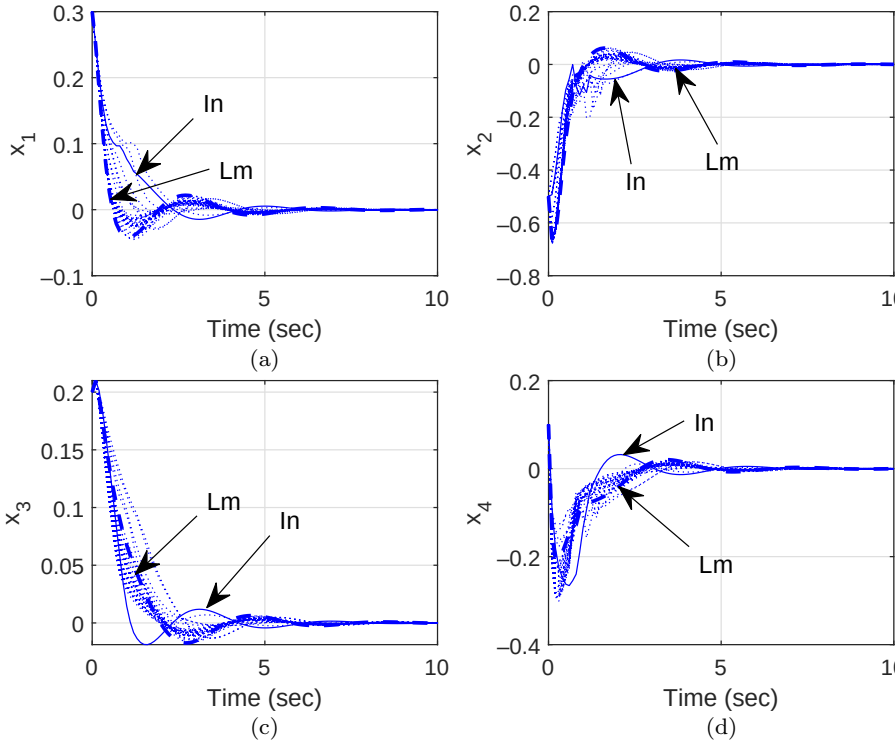

Fig. 8.4. State trajectories of the two inverted pendulum systems.

The trajectories of the iterative multi-control laws are illustrated in Fig. 8.3. Although for each iteration, only one of the iterative control laws is updated for the system and the other control laws remain unchanged, the system can still be maintained stable by the distributed iterative control laws. In Fig. 8.4, the states of the two inverted pendulum system are illustrated. Thus, the correctness of the theoretical analysis can be verified.

8.5 Conclusion

In this chapter, a novel continuous-time distributed policy iteration algorithm is proposed to be applied in multi-controller nonlinear systems for achieving the infinite horizon optimal control. In each iteration of the proposed algorithms, only one of the multi-control laws is

updated instead of all the control laws, which implies that the control laws are improved one by one. First, the detailed iterative methods of the distributed policy iteration are introduced. Second, this paper also discusses the admissibility of the proposed multi-control laws. In addition, the iterative value function can converge to optimum, which is the solution of HJB equations. Finally, numerical simulation is conducted to verify the effectiveness of the presented methods.

References

[1] Murray, J.J., Cox, C.J., Lendaris, G.G. and Saeks, R. (2002). Adaptive dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews* 32(2), 140–153.

[2] Lewis, F.L., Vrabie, D. and Vamvoudakis, K.G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine* 32(6), 76–105.

[3] Abu-Khalaf, M. and Lewis, F.L. (2005). Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5), 779–791.

[4] Modares, H. and Lewis, F.L. (2014). Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica* 50(7), 1780–1792.

[5] Liu, D., Li, H. and Wang, D. (2014). Online synchronous approximate optimal learning algorithm for multiplayer nonzero-sum games with unknown dynamics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44(8), 1015–1027.

[6] Bertsekas, D.P. (2017). Value and policy iterations in optimal control and adaptive dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems* 28(3), 500–509.

This page intentionally left blank

Chapter 9

A Novel Data-Based Fault-Tolerant Control Method for Multicontroller Linear Systems Via Distributed Policy Iteration

9.1 Introduction

This chapter presents a novel data-based fault-tolerant control method for multicontroller linear systems via distributed policy iteration. Traditional fault-tolerant control methods based on policy iteration update all the control laws in each iteration, which causes huge computational burden in the situation of high-dimension control laws. In order to solve this problem, a novel distributed policy iteration method is presented, where only one iterative control law is updated in each iteration, to realize the fault-tolerant control of multicontroller linear systems. First, the model-based fault-tolerant control via distributed policy iteration is provided. Based on the model-based method, a data-based fault-tolerant control method is presented. Finally, numerical experiment is given to show the performance of the presented method.

9.2 Problem Formulation

Consider the following multicontroller linear system:

$$\dot{x} = Ax + \sum_{i=1}^{N} B_i \left(u_i - f_i\right), \tag{9.1}$$

where $x \in \mathbb{R}^n$, $u_i \in \mathbb{R}^m, i = 1, 2, \ldots, N$ represent system state and control inputs, respectively. $f_i \in \mathbb{R}^m, i = 1, 2, \ldots, N$ denote constant actuator faults. A and $B_i, i = 1, 2, \ldots, N$ are the system matrices with suitable dimensions.

For the system (9.1) with no faults, i.e., $f_i = 0, \forall i = 1, 2, \ldots, N$, we can define performance index function as follows

$$J\left(x\right) = \int_0^\infty \left(x^\mathsf{T} Q x + \sum_{i=1}^{N} u_i^\mathsf{T} R_i u_i\right) \mathrm{d}t, \tag{9.2}$$

where Q and $R_i, i = 1, 2, \ldots, N$ are symmetric positive definite matrices with suitable dimensions. The matrix Q can also be written as $Q = C^\mathsf{T} C$, where $C \in \mathbb{R}^{p \times n}$. We can obtain the optimal control laws for $i = 1, 2, \ldots, N$ as

$$u_i^* = -K_i^* x = -R_i^{-1} B_i^\mathsf{T} P^* x, \tag{9.3}$$

where P^* is the solution to the following algebraic Riccati equation (ARE)

$$A^\mathsf{T} P + PA + Q - \sum_{i=1}^{N} P B_i R_i^{-1} B_i^\mathsf{T} P = 0. \tag{9.4}$$

The following assumption is provided.

Assumption 9.1. The system (9.1) is controllable, and (A, C) is observable.

Assumption 9.2. The faults $f_i, i = 1, 2, \ldots, N$ are unknown and bounded constant actuator faults.

It is hard to obtain $u_i^*, i = 1, 2, \ldots, N$ because of the existence of the actuator faults. Therefore, a fault-tolerant control method via distributed policy iteration will be introduced to handle this problem.

9.3 Model-Based Fault-Tolerant Control via Distributed Policy Iteration

In this section, the distributed fault-tolerant control method will be discussed with full system information. The distributed policy iteration method will firstly be presented in the fault-free situation, and then, the fault-tolerant control design will be realized by fault compensation method.

9.3.1 *Derivations of the distributed policy iteration method*

Let $K_i^0, i = 1, 2, \ldots, N$ be initial stabilizing state feedback gain matrices. We can obtain P^0 by the following Lyapunov equation

$$\begin{aligned}\left(A - \sum_{i=1}^{N} B_i K_i^0\right)^{\mathsf{T}} P^0 + P^0 \left(A - \sum_{i=1}^{N} B_i K_i^0\right) \\ + Q + \sum_{i=1}^{N} \left(K_i^0\right)^{\mathsf{T}} R_i K_i^0 = 0.\end{aligned} \tag{9.5}$$

For iteration index $j = 1$, choose $\tau_1 \in \mathcal{N}, \mathcal{N} = \{1, 2, \ldots, N\}$ randomly. We can obtain state feedback gain matrix $K_{\tau_1}^1$ by the following equation

$$K_{\tau_1}^1 = R_{\tau_1}^{-1} B_{\tau_1}^{\mathsf{T}} P^0. \tag{9.6}$$

For $i = 1, 2, \ldots, N$ and $i \neq \tau_1$, let $K_i^1 = K_i^0$. We can obtain P^1 by the following Lyapunov equation

$$\begin{aligned}\left(A - \sum_{i=1}^{N} B_i K_i^1\right)^{\mathsf{T}} P^1 + P^1 \left(A - \sum_{i=1}^{N} B_i K_i^1\right) \\ + Q + \sum_{i=1}^{N} \left(K_i^1\right)^{\mathsf{T}} R_i K_i^1 = 0.\end{aligned} \tag{9.7}$$

For $j = 1, 2, \ldots,$ choose $\tau_j \in \mathcal{N}$ randomly. We can obtain $K_{\tau_j}^j$ by

$$K_{\tau_j}^j = R_{\tau_j}^{-1} B_{\tau_j}^{\mathsf{T}} P^{j-1}. \tag{9.8}$$

For $i = 1, 2, \ldots, N$ and $i \neq \tau_j$, let $K_i^j = K_i^{j-1}$. We can obtain P^j by the following Lyapunov equation

$$\left(A - \sum_{i=1}^{N} B_i K_i^j\right)^{\mathsf{T}} P^j + P^j \left(A - \sum_{i=1}^{N} B_i K_i^j\right) + Q + \sum_{i=1}^{N} \left(K_i^j\right)^{\mathsf{T}} R_i K_i^j = 0. \tag{9.9}$$

The distributed policy iteration algorithm can be shown as Algorithm 9.1.

Algorithm 9.1 Distributed Policy Iteration Algorithm

Require:

Choose initial stabilizing state feedback gain matrices $K_i^0, i = 1, 2, \ldots, N$ randomly.

Ensure:

1: Let iteration index $j = 0$. Solve the Lyapunov equation (9.5).

2: Let iteration index $j = j + 1$, choose $\tau_j \in \mathcal{N}$ randomly. Calculate the state feedback gain matrix $K_{\tau_j}^j$ as

$$K_{\tau_j}^j = R_{\tau_j}^{-1} B_{\tau_j}^{\mathsf{T}} P^{j-1}. \tag{9.10}$$

3: For $i = 1, 2, \ldots, N$ and $i \neq \tau_j$, let $K_i^j = K_i^{j-1}$. Solve the following equation

$$\left(A - \sum_{i=1}^{N} B_i K_i^j\right)^{\mathsf{T}} P^j + P^j \left(A - \sum_{i=1}^{N} B_i K_i^j\right) + Q + \sum_{i=1}^{N} \left(K_i^j\right)^{\mathsf{T}} R_i K_i^j = 0. \tag{9.11}$$

4: Repeat Steps 2–3 until convergence.

5: **return** P^j, $K_i^j, i = 1, 2, \ldots, N$.

The properties of Algorithm 9.1 will be analyzed in the following theorem.

Theorem 9.1. *For $j = 0, 1, \ldots,$ the state feedback gain matrices $K_i^j, i = 1, 2, \ldots, N$ and the solution P^j of Lyapunov equation can be calculated by* (9.5)–(9.9)*. We can obtain the following properties*:

1. *If $A - \sum_{i=1}^{N} B_i K_i^j$ is Hurwitz, then $A - \sum_{i=1}^{N} B_i K_i^{j+1}$ is Hurwitz.*
2. *$P^{j+1} \leq P^j$.*
3. *Define $\Pi_i = \{j | \tau_j = i, i \in \mathcal{N}\}$ and π_i as the number of elements of Π_i. If $\pi_i \to \infty, i = 1, 2, \ldots, N,$ as $j \to \infty,$ then $K_i^j \to K_i^*, i = 1, 2, \ldots, N$ and $P^j \to P^*$.*

Proof. (1) If $A - \sum_{i=1}^{N} B_i K_i^j$ is Hurwitz, we can obtain P^j from (9.9). Selecting $\tau_{j+1} \in \mathcal{N}$ randomly, we can obtain that

$$K_{\tau_{j+1}}^{j+1} = R_{\tau_{j+1}}^{-1} B_{\tau_{j+1}}^{\mathsf{T}} P^j \tag{9.12}$$

and $K_i^{j+1} = K_i^j$ for $i = 1, 2, \ldots, N$ and $i \neq \tau_{j+1}$. According to (9.9), it can be derived that

$$\begin{aligned} &\left(A - \sum_{i=1}^{N} B_i K_i^{j+1}\right)^{\mathsf{T}} P^j + P^j \left(A - \sum_{i=1}^{N} B_i K_i^{j+1}\right) \\ &\quad + Q + \sum_{i=1}^{N} \left(K_i^{j+1}\right)^{\mathsf{T}} R_i K_i^{j+1} \leq 0. \end{aligned} \tag{9.13}$$

We can construct Lyapunov function as

$$L_j = x^{\mathsf{T}} P^j x. \tag{9.14}$$

Taking the derivative of (9.14) along $K_i^{j+1}, i = 1, 2, \ldots, N$, we have

$$\begin{aligned} \dot{L}_j &= \dot{x}^{\mathsf{T}} P^j x + x^{\mathsf{T}} P^j \dot{x} \\ &= x^{\mathsf{T}} \left(A - \sum_{i=1}^{N} B_i K_i^{j+1}\right)^{\mathsf{T}} P^j x \\ &\quad + x^{\mathsf{T}} P^j \left(A - \sum_{i=1}^{N} B_i K_i^{j+1}\right) x \end{aligned}$$

$$\leq -x^{\mathsf{T}}\left(Q+\sum_{i=1}^{N}\left(K_i^{j+1}\right)^{\mathsf{T}}R_iK_i^{j+1}\right)x$$
$$<0. \tag{9.15}$$

Therefore, $A-\sum_{i=1}^{N}B_iK_i^{j+1}$ is Hurwitz.

(2) Let $K_i^0, i=1,2,\ldots,N$ be initial stabilizing state feedback gain matrices. According to the idea from (9.12)–(9.15) and Algorithm 9.1, it can be derived that $K_i^1, i=1,2,\ldots,N$ are stabilizing state feedback gain matrices. Construct Lyapunov function as

$$L_0=x^{\mathsf{T}}P^0x \tag{9.16}$$

and

$$L_1=x^{\mathsf{T}}P^1x. \tag{9.17}$$

Taking the derivative of (9.16) and (9.17) along $K_i^1, i=1,2,\ldots,N$, according to (9.9) and (9.15), we have

$$\begin{aligned}\dot{L}_0&=\dot{x}^{\mathsf{T}}P^0x+x^{\mathsf{T}}P^0\dot{x}\\&\leq -x^{\mathsf{T}}\left(Q+\sum_{i=1}^{N}\left(K_i^1\right)^{\mathsf{T}}R_iK_i^1\right)x\\&=x^{\mathsf{T}}\left(A-\sum_{i=1}^{N}B_iK_i^1\right)^{\mathsf{T}}P^1x\\&\quad+x^{\mathsf{T}}P^1\left(A-\sum_{i=1}^{N}B_iK_i^1\right)x\\&=\dot{x}^{\mathsf{T}}P^1x+x^{\mathsf{T}}P^1\dot{x}\\&=\dot{L}_1.\end{aligned} \tag{9.18}$$

Furthermore, $\lim_{t\to\infty}L_0=\lim_{t\to\infty}L_1=0$. Therefore, it can be derived that $L_1\leq L_0$, which indicates that $P^1\leq P^0$. Based on mathematical induction, we have $P^{j+1}\leq P^j$ for $j=0,1,\ldots$.

(3) As P^j satisfies $P^{j+1} \leq P^j$ for $j = 0, 1, \ldots$, we can define that $\lim_{j\to\infty} P^j = P^\infty$, which satisfies the ARE

$$A^\mathsf{T} P^\infty + P^\infty A + Q - \sum_{i=1}^{N} P^\infty B_i R_i^{-1} B_i^\mathsf{T} P^\infty = 0. \tag{9.19}$$

Based on Assumption 9.1, it can be derived that the ARE (9.4) has a unique positive definite solution [1]. Therefore, we have

$$\lim_{j\to\infty} P^j = P^\infty = P^* \tag{9.20}$$

and

$$\lim_{j\to\infty} K_i^j = K_i^*, \quad i = 1, 2, \ldots, N. \tag{9.21}$$

The proofs are complete. □

9.3.2 *Fault-tolerant control method*

The fault-tolerant control method is presented in this subsection. The fault compensation method is utilized to eliminate the effects of system faults. For the system (9.1), the control laws can be designed as

$$u_i = u_i^* + \hat{f}_i, \quad i = 1, 2, \ldots, N, \tag{9.22}$$

where $\hat{f}_i, i = 1, 2, \ldots, N$ are fault compensators which have properties that

$$\dot{\hat{f}}_i = \alpha_i(-x^\mathsf{T} B_i + 2(u_i^*)^\mathsf{T} R_i)^\mathsf{T}, \quad i = 1, 2, \ldots, N, \tag{9.23}$$

where $\alpha_i, i = 1, 2, \ldots, N$ are arbitrary positive constants. The effectiveness of fault compensation method can be verified by the following theorem.

Theorem 9.2. *For the system* (9.1) *with faults, define* $\lambda_{\max}(\cdot)$ *and* $\lambda_{\min}(\cdot)$ *as maximum eigenvalue and minimum eigenvalue, respectively. If the control laws are designed as* (9.22)–(9.23) *and the matrices* Q *and* $R_i, i = 1, 2, \ldots, N$ *are selected as follows*

$$\begin{aligned} &\lambda_{\min}(Q) > \lambda_{\max}(A) + \frac{1}{2}, \\ &\lambda_{\min}(R_i) > \frac{N}{2}\lambda_{\max}\left(B_i^\mathsf{T} B_i\right), \quad i = 1, 2, \ldots, N \end{aligned} \tag{9.24}$$

then the system (9.1) *is asymptotical stable.*

Proof. Construct Lyapunov function as

$$L = \frac{1}{2}x^{\mathsf{T}}x + x^{\mathsf{T}}P^*x + \sum_{i=1}^{N}\frac{1}{2\alpha_i}\tilde{f}_i^{\mathsf{T}}\tilde{f}_i \tag{9.25}$$

where $\tilde{f}_i = f_i - \hat{f}_i$. Taking the derivative of (9.25), we can obtain that

$$\begin{aligned}
\dot{L} &= x^{\mathsf{T}}\dot{x} + \dot{x}^{\mathsf{T}}P^*x + x^{\mathsf{T}}P^*\dot{x} + \sum_{i=1}^{N}\frac{1}{\alpha_i}\tilde{f}_i^{\mathsf{T}}\dot{\tilde{f}}_i \\
&= x^{\mathsf{T}}\left(Ax + \sum_{i=1}^{N}B_i(u_i^* + \hat{f}_i - f_i)\right) \\
&\quad + \left(Ax + \sum_{i=1}^{N}B_i(u_i^* + \hat{f}_i - f_i)\right)^{\mathsf{T}}P^*x \\
&\quad + x^{\mathsf{T}}P^*\left(Ax + \sum_{i=1}^{N}B_i(u_i^* + \hat{f}_i - f_i)\right) \\
&\quad - \sum_{i=1}^{N}\tilde{f}_i^{\mathsf{T}}\left(-x^{\mathsf{T}}B_i + 2(u_i^*)^{\mathsf{T}}R_i\right)^{\mathsf{T}} \\
&= x^{\mathsf{T}}\left(Ax + \sum_{i=1}^{N}B_iu_i^*\right) \\
&\quad - \left(x^{\mathsf{T}} + 2x^{\mathsf{T}}P^*\right)\sum_{i=1}^{N}B_i\tilde{f}_i \\
&\quad + \left(Ax + \sum_{i=1}^{N}B_iu_i^*\right)^{\mathsf{T}}P^*x
\end{aligned}$$

$$+ x^{\mathsf{T}} P^* \left(Ax + \sum_{i=1}^{N} B_i u_i^* \right)$$

$$- \sum_{i=1}^{N} \left(-x^{\mathsf{T}} B_i + 2(u_i^*)^{\mathsf{T}} R_i \right) \tilde{f}_i. \tag{9.26}$$

According to (9.3), we have

$$(u_i^*)^{\mathsf{T}} R_i = -x^{\mathsf{T}} P^* B_i. \tag{9.27}$$

Based on (9.9) and (9.27), it can be derived that

$$\dot{L} = x^{\mathsf{T}} Ax + x^{\mathsf{T}} \sum_{i=1}^{N} B_i u_i^* - x^{\mathsf{T}} Qx - \sum_{i=1}^{N} (u_i^*)^{\mathsf{T}} R_i u_i^*$$

$$\leq \lambda_{\max}(A) \|x\|^2 + \frac{1}{2}\|x\|^2 + \frac{1}{2}\left\| \sum_{i=1}^{N} B_i u_i^* \right\|^2$$

$$- \lambda_{\min}(Q) \|x\|^2 - \sum_{i=1}^{N} \lambda_{\min}(R_i) \|u_i^*\|^2. \tag{9.28}$$

It can be derived that

$$\begin{aligned} \frac{1}{2}\left\| \sum_{i=1}^{N} B_i u_i^* \right\|^2 &= \frac{1}{2}\left(\sum_{i=1}^{N} B_i u_i^* \right)^{\mathsf{T}} \sum_{i=1}^{N} B_i u_i^* \\ &= \frac{1}{2} \sum_{i=1,k=1}^{N} (B_i u_i^*)^{\mathsf{T}} B_k u_k^* \\ &\leq \frac{N}{2} \sum_{i=1}^{N} (B_i u_i^*)^{\mathsf{T}} B_i u_i^* \\ &\leq \frac{N}{2} \sum_{i=1}^{N} \lambda_{\max}\left(B_i^{\mathsf{T}} B_i \right) \|u_i^*\|^2. \end{aligned} \tag{9.29}$$

Therefore, we have

$$\dot{L} \leq \left(\lambda_{\max}(A) + \frac{1}{2} - \lambda_{\min}(Q)\right)\|x\|^2 + \sum_{i=1}^{N}\left(\frac{N}{2}\lambda_{\max}\left(B_i^{\mathrm{T}}B_i\right) - \lambda_{\min}(R_i)\right)\|u_i^*\|^2. \quad (9.30)$$

It can be derived that $\dot{L} < 0$ can be guaranteed if the condition (9.24) is satisfied.

The proof is complete. □

According to Algorithm 9.1 and Theorem 9.2, the full system information is still required for distributed policy iteration algorithm and the fault compensator design. In the next section, a data-based method will be presented to solve this problem. The presented method only requires the information of the system matrices $B_i, i = 1, 2, \ldots, N$.

9.4 Data-Based Fault-Tolerant Control via Distributed Policy Iteration

In this section, a data-based fault-tolerant control method via distributed policy iteration is presented. In Ref. [2], a data-based policy iteration method was proposed which required the partial information of control systems. However, the method proposed in Ref. [2] may cause huge computational burden with high-dimension control laws, and the effects of faults were not considered in the control systems. Inspired by Ref. [2]. a data-based fault-tolerant control method via distributed policy iteration is presented, which can eliminate the effects of actuator faults and reduce the computational burden.

9.4.1 *Data-based distributed policy iteration*

To derive the data-based distributed policy iteration method, we can rewrite (9.2) as

$$\begin{aligned} J(x(t)) &= \int_t^{\infty} \left(x^{\mathsf{T}} Q x + \sum_{i=1}^{N} u_i^{\mathsf{T}} R_i u_i \right) \mathrm{d}\tau \\ &= \int_t^{t+\delta t} \left(x^{\mathsf{T}} Q x + \sum_{i=1}^{N} u_i^{\mathsf{T}} R_i u_i \right) \mathrm{d}\tau \\ &\quad + \int_{t+\delta t}^{\infty} \left(x^{\mathsf{T}} Q x + \sum_{i=1}^{N} u_i^{\mathsf{T}} R_i u_i \right) \mathrm{d}\tau \\ &= \int_t^{t+\delta t} \left(x^{\mathsf{T}} Q x + \sum_{i=1}^{N} u_i^{\mathsf{T}} R_i u_i \right) \mathrm{d}\tau \\ &\quad + J\left(x\left(t+\delta t\right)\right). \end{aligned} \tag{9.31}$$

According to (9.31), we have

$$\begin{aligned} x^{\mathsf{T}}\left(t\right) P x\left(t\right) = \int_t^{t+\delta t} x^{\mathsf{T}} \left(Q + \sum_{i=1}^{N} K_i^{\mathsf{T}} R_i K_i \right) x \mathrm{d}\tau \\ + x^{\mathsf{T}}\left(t+\delta t\right) P x\left(t+\delta t\right). \end{aligned} \tag{9.32}$$

Then, the following lemma can be provided.

Lemma 9.1. *If* $A - \sum_{i=1}^{N} B_i K_i^j$ *is Hurwitz, the solution* P^j *of equation* (9.11) *is equivalent to solving the following equation.*

$$\begin{aligned} x^{\mathsf{T}}\left(t\right) P^j x\left(t\right) = \int_t^{t+\delta t} x^{\mathsf{T}} \left(Q + \sum_{i=1}^{N} \left(K_i^j\right)^{\mathsf{T}} R_i K_i^j \right) x \mathrm{d}\tau \\ + x^{\mathsf{T}}\left(t+\delta t\right) P^j x\left(t+\delta t\right). \end{aligned} \tag{9.33}$$

Proof. For the system (9.1) with no faults, we have

$$\dot{x} = \left(A - \sum_{i=1}^{N} B_i K_i^j \right) x. \tag{9.34}$$

Construct Lyapunov function as

$$L_j = x^{\mathsf{T}} P^j x. \tag{9.35}$$

It can be derived that

$$\begin{aligned}\dot{L}_j &= \dot{x}^{\mathsf{T}} P^j x + x^{\mathsf{T}} P^j \dot{x} \\ &= x^{\mathsf{T}} \left(\left(A - \sum_{i=1}^{N} B_i K_i^j \right)^{\mathsf{T}} P^j \right. \\ &\quad \left. + P^j \left(A - \sum_{i=1}^{N} B_i K_i^j \right) \right) x \\ &= -x^{\mathsf{T}} \left(Q + \sum_{i=1}^{N} \left(K_i^j \right)^{\mathsf{T}} R_i K_i^j \right) x. \end{aligned} \tag{9.36}$$

We have

$$\begin{aligned}\int_t^{t+\delta t} \dot{L}_j \mathrm{d}\tau &= x^{\mathsf{T}}(t+\delta t) P^j x(t+\delta t) - x^{\mathsf{T}}(t) P^j x(t) \\ &= -\int_t^{t+\delta t} x^{\mathsf{T}} \left(Q + \sum_{i=1}^{N} \left(K_i^j \right)^{\mathsf{T}} R_i K_i^j \right) x \mathrm{d}\tau. \end{aligned} \tag{9.37}$$

The proof is complete. □

For (9.33), we have

$$x^{\mathsf{T}} Q^j x = \left(x^{\mathsf{T}} \otimes x^{\mathsf{T}} \right) \mathrm{vec}\left(Q^j \right), \tag{9.38}$$

where $Q^j = Q + \sum_{i=1}^{N} (K_i^j)^{\mathsf{T}} R_i K_i^j$. For positive integer l, we define σ_{xx} and Φ_{xx} as

$$\begin{aligned}\sigma_{xx} &= \left[\mathrm{vecv}\left(x(t_1) \right) - \mathrm{vecv}\left(x(t_0) \right) \cdots \right. \\ &\qquad \left. \mathrm{vecv}\left(x(t_l) \right) - \mathrm{vecv}\left(x(t_{l-1}) \right) \right]^{\mathsf{T}}, \\ \Phi_{xx} &= \left[\int_{t_0}^{t_1} x \otimes x d\tau \cdots \int_{t_{l-1}}^{t_l} x \otimes x d\tau \right]^{\mathsf{T}}, \end{aligned}$$

where $0 \leq t_0 < t_1 < \cdots < t_l$, and $\mathrm{vecv}(x) = [x_1^2, x_1 x_2, \ldots, x_1 x_n, x_2^2, x_2 x_3, \ldots, x_{n-1} x_n, x_n^2]^{\mathsf{T}}$. According to (9.37)–(9.38), we

can obtain that

$$\sigma_{xx}\text{vecs}\left(P^j\right) = \Psi^j, \tag{9.39}$$

where

$$\text{vecs}(P^j) = [p_{11}^j, 2p_{12}^j, \ldots, 2p_{1n}^j, p_{22}^j, 2p_{23}^j, \ldots, 2p_{n-1,n}^j, p_{nn}^j]^\mathsf{T}$$
$$\Psi^j = -\Phi_{xx}\text{vec}(Q^j).$$

Based on (9.39), it can be derived that

$$\text{vecs}(P^j) = ((\sigma_{xx})^\mathsf{T}\sigma_{xx})^{-1}(\sigma_{xx})^\mathsf{T}\tilde{\Psi}^j. \tag{9.40}$$

A data-based distributed policy iteration algorithm can be provided as shown in Algorithm 9.2.

Algorithm 9.2 Data-Based Distributed Policy Iteration Algorithm

Require:

Choose initial stabilizing state feedback gain matrices K_i^0, $i = 1, 2, \ldots, N$ randomly.

Choose a computation precision ε.

Ensure:

1: Let iteration index $j = 0$. Employ the control laws $u_i^0 = -K_i^0 x + \xi_i^0, i = 1, 2, \ldots, N$ to the system (9.1) on the time interval $[t_0, t_l]$, where ξ_i^0 represents exploration noise. Compute σ_{xx}, Φ_{xx} and solve the equation (9.40).

2: Let iteration index $j = j + 1$, choose $\tau_j \in \mathcal{N}$ randomly. Calculate the state feedback gain matrix $K_{\tau_j}^j$ as

$$K_{\tau_j}^j = R_{\tau_j}^{-1} B_{\tau_j}^\mathsf{T} P^{j-1}. \tag{9.41}$$

3: For $i = 1, 2, \ldots, N$ and $i \neq \tau_j$, let $K_i^j = K_i^{j-1}$. Employ the control laws $u_i^j = -K_i^j x + \xi_i^j, i = 1, 2, \ldots, N$ to the system (9.1) on the time interval $[t_0, t_l]$, where ξ_i^j represents exploration noise. Compute σ_{xx}, Φ_{xx} and solve the equation (9.40).

4: Repeat Steps 2–3 until $\left\|P^j - P^{j-1}\right\| \leq \varepsilon$ for $j \geq 1$.

5: **return** P^j, $K_i^j, i = 1, 2, \ldots, N$.

9.4.2 *Data-based fault-tolerant control method*

According to Subsection 9.3.2, the fault compensators (9.22) can be designed to cope with the system faults. The effectiveness of the fault compensators (9.22) is demonstrated as follows.

Theorem 9.3. *For the system* (9.1) *with faults, if the control laws are designed as* (9.22) *and* (9.23) *and the matrices* Q *and* $R_i, i = 1, 2, \ldots, N$ *are chosen as* (9.24), *where* P^* *and* u_i^* *can be obtained from Algorithm* 9.2, *then the system* (9.1) *is asymptotical stable.*

Proof. According to Lemma 9.1, it can be derived that (9.11) is equivalent to (9.33). Given stabilizing state feedback gain matrices $K_i^j, i = 1, 2, \ldots, N$, we can obtain P^j by the Lyapunov equation (9.11), which also satisfies (9.33). $K_i^{j+1}, i = 1, 2, \ldots, N$ can be determined by (9.10). By (9.40), we can obtain the unique solution P^j, which satisfies (9.33) for the given matrices $K_i^j, i = 1, 2, \ldots, N$. $K_i^{j+1}, i = 1, 2, \ldots, N$ can be determined by (9.41). Therefore, solving (9.40)–(9.41) is equal to (9.10)–(9.11). Based on the system states, we can obtain P^* and $K_i^*, i = 1, 2, \ldots, N$ from Algorithm 9.2. According to Theorems 1 and 2, it can be derived that the system (9.1) is asymptotical stable under the condition (9.24) by utilizing the control laws (9.22) and (9.23).

The proof is complete. □

9.5 Simulation Example

In this section, a simulation experiment is given to testify the correctness of the presented method. The power system with modifications is provided [3] as follows:

$$\dot{x} = \begin{bmatrix} -0.0665 & 8 & 0 & 0 \\ 0 & -3.663 & 3.663 & 0 \\ -6.86 & 0 & -13.736 & -13.736 \\ 0.6 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 3.736 \\ 0 \end{bmatrix} (u_1 - f_1) + \begin{bmatrix} 0 \\ 3.736 \\ 0 \\ 0 \end{bmatrix} (u_2 - f_2), \tag{9.42}$$

where $f_1 = -15$ and $f_2 = -10$ are actuator faults. The matrices Q, R_1 and R_2 are selected to be $Q = 15I_4$, $R_1 = 15$ and $R_2 = 15$, respectively. The simulation is performed with $x_0 = \begin{bmatrix}-1.2 & -3 & 4.5 & 0.5\end{bmatrix}^{\mathsf{T}}$, $\mathrm{K}_1^0 = \begin{bmatrix}0.5 & 0.2 & 0.5 & 0.2\end{bmatrix}$ and $\mathrm{K}_2^0 = \begin{bmatrix}0.2 & 0.1 & 0.2 & 0.1\end{bmatrix}$. The simulation results are displayed in Figs. 9.1–9.4. The convergence of P^j, K_1^j and K_2^j are shown in Figs. 9.1–9.2. It can be obtained that the Algorithm 9.2 converges in 15 iterations. We can obtain the matrices P^{15}, K_1^{15} and K_2^{15} as

$$P^{15} = \begin{bmatrix}5.6807 & 4.2096 & 0.4026 & 6.5997\\ 4.2096 & 6.2035 & 1.0859 & 4.0149\\ 0.4026 & 1.0859 & 0.7753 & 0.0000\\ 6.5997 & 4.0149 & 0.0000 & 36.1602\end{bmatrix}$$

$$\mathrm{K}_1^{15} = \begin{bmatrix}0.1003 & 0.2705 & 0.1931 & 0.0000\end{bmatrix}$$

$$\mathrm{K}_2^{15} = \begin{bmatrix}1.0485 & 1.5451 & 0.2705 & 1.0000\end{bmatrix}.$$

By solving ARE (9.4), we can obtain the optimal solutions as

$$P^* = \begin{bmatrix}5.6807 & 4.2096 & 0.4026 & 6.5997\\ 4.2096 & 6.2035 & 1.0859 & 4.0149\\ 0.4026 & 1.0859 & 0.7753 & 0.0000\\ 6.5997 & 4.0149 & 0.0000 & 36.1602\end{bmatrix}$$

$$\mathrm{K}_1^* = \begin{bmatrix}0.1003 & 0.2705 & 0.1931 & 0.0000\end{bmatrix}$$

$$\mathrm{K}_2^* = \begin{bmatrix}1.0485 & 1.5451 & 0.2705 & 1.0000\end{bmatrix}.$$

The convergence of Algorithm 9.2 can be verified. The curves of system states are shown in Figs. 9.3–9.4. The correctness of the presented method can be demonstrated.

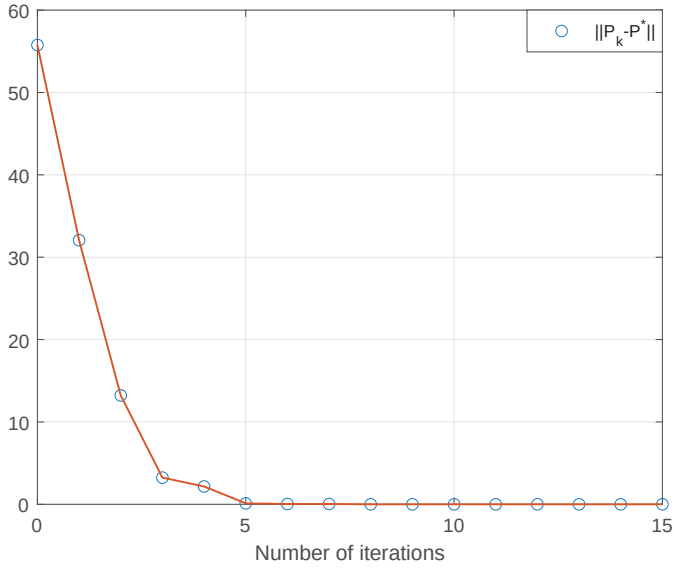

Fig. 9.1. The convergence of P^j.

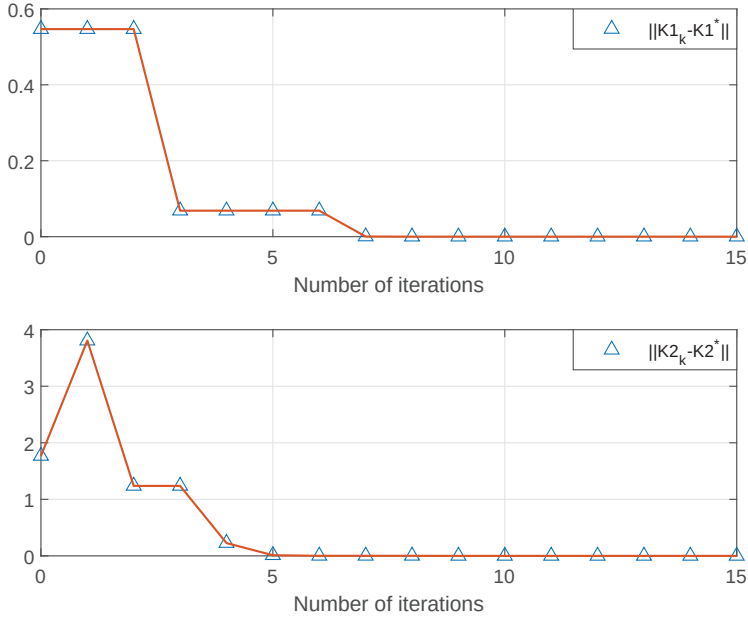

Fig. 9.2. The convergence of K_1^j and K_2^j.

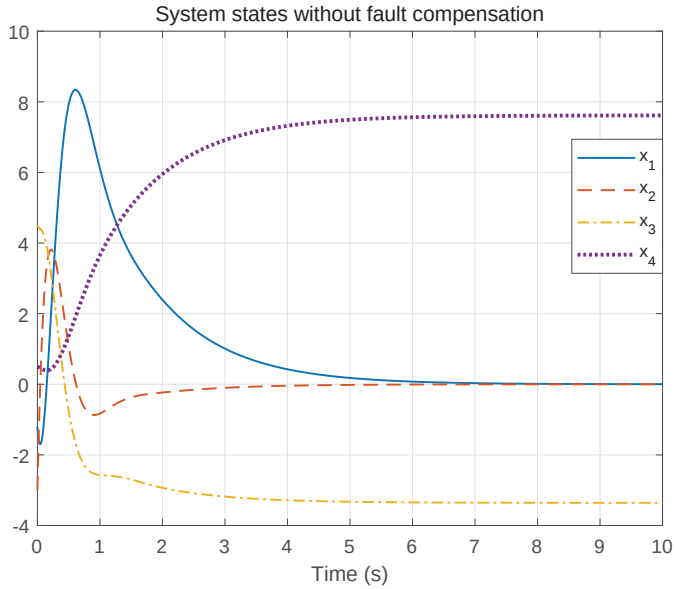

Fig. 9.3. System states without fault compensation.

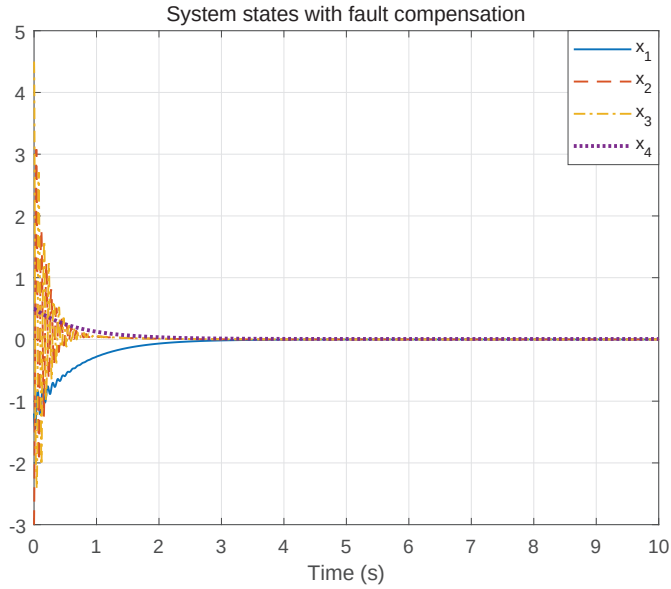

Fig. 9.4. System states with fault compensation.

9.6 Conclusion

A novel data-based fault-tolerant control method based on distributed policy iteration is presented for multicontroller linear systems. Comparing with traditional fault-tolerant control method based on adaptive dynamic programming, the distributed policy iteration method reduces the computational burden by updating only one control law in each iteration, which is the main advantage of the presented method. An online fault-tolerant control method is presented which requires only the partial system information. A simulation example is provided to verify the performance of the presented method.

References

[1] Kleinman, D.L. (1968). On an iterative technique for Riccati equation computations. *IEEE Transactions on Automatic Control* AC13(1), 114–115.

[2] Vrabie, D., Pastravanu, O., Abu-Khalaf M. and Lewis, F.L. (2009). Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica* 45(2), 477–484.

[3] Rizvi, S.A.A. and Lin, Z. (2020). Reinforcement learning-based linear quadratic regulation of continuous-time systems using dynamic output feedback. *IEEE Transactions on Cybernetics* 50(11), 4670–4679.

Chapter 10

A Novel Dual Iterative Q-Learning Method for Optimal Battery Management in Smart Residential Environments

10.1 Introduction

With the rising cost, environmental concerns, and reliability issues, the need to develop optimal control and management systems in residential environments is continuously increasing. Smart residential energy systems, composed of power grids, battery systems, and residential loads which are interconnected over a power management unit, provide end users with the optimal management of energy usage to improve the operation efficiency of power systems. On the other hand, with the rapidly evolving technology of electric storage devices, energy storage based optimal management has attracted much attention. Along with the development of smart grids, increasing intelligence is required in the optimal design of residential energy systems. Hence, the intelligent optimization of battery management becomes a key tool for saving the power expense in smart residential environments.

In this chapter, a novel iterative Q-learning method called "dual iterative Q-learning algorithm" is developed to solve the optimal battery management and control problem in smart residential environments. The main idea is to use ADP technique to obtain the optimal battery management and control scheme iteratively for residential

energy systems. In the developed dual iterative Q-learning algorithm, two iterations are introduced, which are respectively global and local iterations, where local iteration minimizes the total cost of power load in each period and the global iteration makes the iterative Q function converge to the optimum. Based on the dual iterative Q-learning algorithm, the convergence property of iterative Q-learning method for the optimal battery management and control problem is proven for the first time, which guarantees that both the iterative Q function and the iterative control law reach the optimum. Neural networks are introduced to implement the developed dual iterative Q-learning algorithm. Finally, numerical results are given to illustrate the performance of the developed algorithm.

10.2 Problem Formulation

In this section, smart residential energy systems with batteries will be described. The optimization objective of our research will be defined and the corresponding principle of optimality will be introduced.

10.2.1 *Smart residential energy systems*

The smart residential energy system described in Ref. [1] is composed of the power grid, the residential load, the battery system (including a battery and a sinewave inverter) and the power management unit (controller). The battery is connected to the power management unit through an inverter. The schematic diagram of the smart residential energy system can be described in Fig. 10.1.

In this chapter, the optimal battery management and control problem is treated as a discrete time problem with the time step of 1 hour and it is assumed that the residential load varies hourly. The battery will make decisions to meet the demand of the residential load and according to the real-time electricity rate. There are three operational modes for the battery of the residential energy system. (1) Charging mode: when the residential load is low and the electricity rate is inexpensive, the power grid will supply the residential load directly and, at the same time, charge the battery. (2) Idle mode: the power grid will directly supply the residential load at certain hours while the battery energy remains fixed. (3) Discharging mode: the

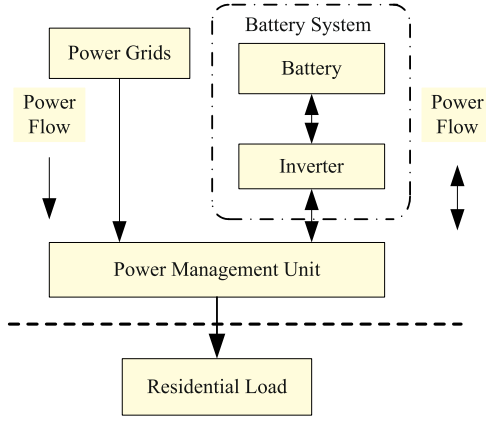

Fig. 10.1. Smart residential energy system.

battery supplies the residential load at hours when the residential load is high and the electricity rate is expensive.

10.2.2 *Battery model*

The battery model used in this work is based on [1–3] where battery efficiency is considered to extend the battery's lifetime as far as possible. Let E_{bt} be the battery energy at time t and let $\eta(\cdot)$ be the charging/discharging efficiency of the battery. Then, the battery model can be expressed as

$$E_{b(t+1)} = E_{bt} - P_{bt} \times \eta(P_{bt}), \tag{10.1}$$

where P_{bt} is the battery power output at time t. Let $P_{bt} > 0$ denote battery discharging. Let $P_{bt} < 0$ denote battery charging and let $P_{bt} = 0$ denote that the battery is idle. Let the efficiency of battery charging/discharging be derived as

$$\eta(P_{bt}) = 0.898 - 0.173|P_{bt}|/P_{\text{rate}}, \tag{10.2}$$

where $P_{\text{rate}} > 0$ is the rated power output of the battery. To extend the battery's lifetime, two constraints need to be considered:

1. The storage limit is considered:

$$E_b^{\min} \leq E_{bt} \leq E_b^{\max}, \tag{10.3}$$

where $E_b^{\min}$ and $E_b^{\max}$ are the minimum and maximum storage energy of the battery, respectively.

2. The charging and discharging power limits are considered:

$$P_b^{\min} \leq P_{bt} \leq P_b^{\max}, \tag{10.4}$$

where $P_b^{\min}$ and $P_b^{\max}$ are the minimum and maximum charging/discharging powers of the battery, respectively.

10.2.3 *Optimization objectives*

Given the residential load and the real-time electricity rate, the objective of the optimal control is to find the optimal battery charging/discharging/idle control law at each time step which minimizes the total expense of the power from the grid while considering the battery limitations. To find the optimal control law, the load balance should be considered. Let P_{Tt} be the power of the residential load at time t and let P_{gt} be the the power from the power grid. For convenience of analysis, we introduce a delay in P_{bt} and we have $P_{Tt} = P_{b(t-1)} + P_{gt}$. We denote $P_{L(t-1)} = P_{Tt}$ and then we can define the load balance as

$$P_{L(t-1)} = P_{b(t-1)} + P_{gt}. \tag{10.5}$$

In this chapter, the power flow from the battery to the grid is not permitted, i.e., we define $P_{gt} \geq 0$, to guarantee the power quality of the grid. To extend the lifetime of the battery, according to (10.3), we desire that the stored energy of the battery is close to the middle of storage limit E_b^o, where

$$E_b^o = \frac{1}{2}(E_b^{\min} + E_b^{\max}). \tag{10.6}$$

We also do not expect frequent battery charging/discharging. Thus, the total performance index function expected to be minimized can be written as

$$\sum_{t=0}^{\infty} \left(m_1(C_t P_{gt})^2 + m_2(E_{bt} - E_b^o)^2 + r(P_{bt})^2\right), \tag{10.7}$$

where m_1, m_2, and r are given positive constants. Let $x_{1t} = P_{gt}$ and $x_{2t} = E_{bt} - E_b^o$ be the two system states. Let $u_t = P_{bt}$ be the control

input. According to (10.1) and (10.5), letting $x_t = [x_{1t}, x_{2t}]^{\mathsf{T}}$, the equation of the residential energy system can be written as

$$x_{t+1} = F(x_t, u_t, t) = \begin{pmatrix} P_{Lt} - u_t \\ x_{2t} - u_t \eta(u_t) \end{pmatrix}. \tag{10.8}$$

Let $\underline{u}_t = (u_t, u_{t+1}, \ldots)$ denote the control sequence from t to ∞. Let $M_t = \begin{bmatrix} m_1 C_t^2 & 0 \\ 0 & m_2 \end{bmatrix}$. Let x_0 be the initial states. Then, the performance index function (10.7) can be written as $J(x_0, \underline{u}_0, 0) = \sum_{t=0}^{\infty} U(x_t, u_t, t)$, where $U(x_t, u_t, t) = x_t^{\mathsf{T}} M_t x_t + r u_t^2$. Generally, functions of the residential load and the real-time electricity rate are periodic. For convenience of analysis, our discussion is based on the following assumption.

Assumption 10.1. The residential load P_{Lt} and the electricity rate C_t are periodic functions with the period $\lambda = 24$ hours.

10.2.4 *The optimality principle*

Define the control sequence set as $\mathfrak{U}_t = \{\underline{u}_t \colon \underline{u}_t = (u_t, u_{t+1}, \ldots), \forall\, u_{t+i} \in \mathbb{R}^m, i = 0, 1, \ldots\}$. Then, for an arbitrary control sequence $\underline{u}_t \in \mathfrak{U}_t$, the optimal performance index function can be defined as

$$J^*(x_t, t) = \inf_{\underline{u}_t} \{J(x_t, \underline{u}_t, t) \colon \underline{u}_t \in \mathfrak{U}_t\}. \tag{10.9}$$

The optimal Q function satisfies the following equation

$$Q^*(x_t, u_t, t) = U(x_t, u_t, t) + \inf_{u_{t+1}} Q^*(x_{t+1}, u_{t+1}, t+1), \tag{10.10}$$

where the optimal performance index function satisfies $J^*(x_t, t) = \inf_{u_t} Q^*(x_t, u_t, t)$. From (10.10), we can define $Q^*(x_t, u_t, t)$ as the optimal Q function. Generally, the optimal Q function $Q^*(x_t, u_t, t)$ is a highly nonlinear and non-analytic function and it is almost impossible to obtain the optimal control by directly solving equation (10.10). In this chapter, a new dual iterative Q-learning algorithm will be developed to obtain the optimal solution indirectly.

10.3 Dual Iterative *Q*-Learning Algorithm of ADP

In this section, a new dual iterative Q-learning algorithm is developed to obtain the optimal control law for residential energy systems. A novel convergence analysis method will be developed in this section.

10.3.1 *Derivation of the dual iterative Q-learning algorithm*

From (10.10), we can see that the optimal Q function $Q^*(x_t, u_t, t)$ is a time-varying function, which means that for different time t, the optimal Q function is different. This makes it difficult to obtain $Q^*(x_t, u_t, t)$. According to Assumption 10.1, for $t = 0, 1, \ldots,$ there exist $\varsigma = 0, 1, \ldots$ and $\theta = 0, 1, \ldots, 23$ that satisfies $t = \varsigma\lambda + \theta$. Let $k = \varsigma\lambda$. Then, we have $P_{Lt} = P_{L(k+\theta)} = P_{L\theta}$ and $C_t = C_{k+\theta} = C_\theta$, respectively. Define $\mathcal{U}_k$ as the control sequence from k to $k + \lambda - 1$, i.e., $\mathcal{U}_k = (u_k, u_{k+1}, \ldots, u_{k+\lambda-1})$. Then, for $k \in \{0, \lambda, 2\lambda, \ldots\}$, we can define a new utility function as

$$\Pi(x_k, \mathcal{U}_k) = \sum_{\theta=0}^{\lambda-1} U(x_{k+\theta}, u_{k+\theta}, \theta). \tag{10.11}$$

Then, according to (10.11), equation (10.10) can be expressed as

$$Q^*(x_k, \mathcal{U}_k) = \Pi(x_k, \mathcal{U}_k) + \min_{\mathcal{U}_{k+\lambda}} Q^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda}). \tag{10.12}$$

The optimal control law sequence can be expressed by

$$\mathcal{U}^*(x_k) = \arg\min_{\mathcal{U}_k} \{Q^*(x_k, \mathcal{U}_k)\}. \tag{10.13}$$

Based on the preparations above, the new dual iterative Q-learning algorithm of ADP can be developed. In the developed algorithm, two iterations are introduced, which are global and local iterations, respectively. The objective of local iteration is to obtain the iterative control law sequence that minimizes the total cost in each period. The objective of global iteration is to update the iterative performance index function to achieve the optimum.

Let $i = 0, 1, \ldots$ be the global iteration index. Let $\Psi(x_k, u_k)$ be an arbitrary positive semi-definite function. Define the initial Q function $Q_0(x_k, \mathcal{U}_k)$ as

$$Q_0(x_k, \mathcal{U}_k) = \Pi(x_k, \mathcal{U}_k) + \min_{u_{k+\lambda}} \Psi(x_{k+\lambda}, u_{k+\lambda}). \tag{10.14}$$

The iterative control law sequence $\mathcal{U}_0$ can be computed as follows

$$\mathcal{U}_0(x_k) = \arg\min_{\mathcal{U}_k} Q_0(x_k, \mathcal{U}_k). \tag{10.15}$$

The iterative Q function can be updated as

$$\begin{aligned} Q_1(x_k, \mathcal{U}_k) &= \Pi(x_k, \mathcal{U}_k) + \min_{\mathcal{U}_{k+\lambda}} Q_0(x_{k+\lambda}, \mathcal{U}_{k+\lambda}) \\ &= \Pi(x_k, \mathcal{U}_k) + Q_0(x_{k+\lambda}, \mathcal{U}_0(x_{k+\lambda})). \end{aligned} \tag{10.16}$$

For $i = 1, 2, \ldots$, the global iteration will proceed between

$$\mathcal{U}_i(x_k) = \arg\min_{\mathcal{U}_k} Q_i(x_k, \mathcal{U}_k) \tag{10.17}$$

and

$$\begin{aligned} Q_{i+1}(x_k, \mathcal{U}_k) &= \Pi(x_k, \mathcal{U}_k) + \min_{\mathcal{U}_{k+\lambda}} Q_i(x_{k+\lambda}, \mathcal{U}_{k+\lambda}) \\ &= \Pi(x_k, \mathcal{U}_k) + Q_i(x_{k+\lambda}, \mathcal{U}_i(x_{k+\lambda})). \end{aligned} \tag{10.18}$$

From (10.15)–(10.18), we can see that for $i = 0, 1, \ldots$, if $\mathcal{U}_i(x_k)$ is obtained, the iterative Q function can be updated. As $\mathcal{U}_i(x_k)$ is an iterative control law sequence, local iteration is introduced to obtain $\mathcal{U}_i(x_k)$ iteratively. For $i = 0, 1, \ldots$, let $j = 0, 1, \ldots, 23$ be the local iteration index. For $i = 0$ and $j = 0$, let the initial iterative performance index be

$$Q_0^0(x_k, u_k) = \Psi(x_k, u_k). \tag{10.19}$$

Then, for $j = 0, 1, \ldots, 23$, the local iteration will proceed between

$$u_0^j(x_k) = \arg\min_{u_k} Q_i^j(x_k, u_k), \tag{10.20}$$

and

$$Q_0^{j+1}(x_k, u_k) = U(x_k, u_k, j) + \min_{u_{k+1}} Q_0^j(x_{k+1}, u_{k+1})$$
$$= U(x_k, u_k, j) + Q_0^j(x_{k+1}, u_0^j(x_{k+1})), \tag{10.21}$$

where we let $x_{k+1} = \begin{pmatrix} P_{L(\lambda-1-j)} - u_k \\ x_{2k} - u_k\eta(u_k) \end{pmatrix}$. Let $U(x_k, u_k, j) = x_k^T M_{\lambda-1-j}$ $x_k + ru_k^2$, where $M_{\lambda-1-j} = \begin{bmatrix} m_1 C_{\lambda-1-j}^2 & 0 \\ 0 & m_2 \end{bmatrix}$. For $i = 1, 2, \ldots,$ we let $Q_i^0(x_k, u_k) = Q_{i-1}^{24}(x_k, u_k)$. For $j = 0, 1, \ldots, 23,$ the local iteration will proceed between

$$u_i^j(x_k) = \arg\min_{u_k} Q_i^j(x_k, u_k), \tag{10.22}$$

and

$$Q_i^{j+1}(x_k, u_k) = U(x_k, u_k, j) + \min_{u_{k+1}} Q_i^j(x_{k+1}, u_{k+1})$$
$$= U(x_k, u_k, j) + Q_0^j(x_{k+1}, u_0^j(x_{k+1})). \tag{10.23}$$

Then, for $i = 0, 1, \ldots,$ we can obtain the iterative control law sequence by

$$\mathcal{U}_i(x_k) = \{u_i^0(x_k), u_i^1(x_k), \ldots, u_i^{23}(x_k)\}. \tag{10.24}$$

From the dual iterative Q-learning algorithm (10.15)–(10.24), we can see that the local iteration (10.19)–(10.24) proceeds for a finite number of iterations, while the global iteration (10.15)–(10.18) proceeds for infinite number of iterations. Thus, as $i \to \infty$, the algorithm should be convergent which makes both $Q_{i+1}(x_k, \mathcal{U}_k)$ and $\mathcal{U}_i(x_k)$ converge to the optimum. In the next subsection, the convergence property of the dual iterative Q-learning algorithm will be investigated.

10.3.2 *Properties of the dual iterative Q-learning algorithm*

In the above subsection, we have developed the dual iterative Q-learning algorithm, where the iterative Q function $Q_i(x_k, \mathcal{U}_k)$ is

used to approximate the optimal Q function $Q^*(x_k, \mathcal{U}_k)$ and the iterative control law $\mathcal{U}_i(x_k)$ is used to approximate $\mathcal{U}^*(x_k)$. In this subsection, the convergence property will be developed to guarantee that both the iterative Q function and the iterative control law converge to the optimum.

Theorem 10.1. *For $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, 23$, let the iterative Q functions $Q_i(x_k, \mathcal{U}_k)$ and $Q_i^j(x_k, u_k)$ be obtained by* (10.14)–(10.24). *Then, we have*

$$\min_{\mathcal{U}_k} Q_i(x_k, \mathcal{U}_k) = \min_{u_k} Q_i^{24}(x_k, u_k). \tag{10.25}$$

Proof. The statement can be proven by mathematical induction. First, for $i = 0$, we have

$$\begin{aligned}
&\min_{u_k} Q_0^{j+1}(x_k, u_k) \\
&\quad = \min_{u_k} \left(U(x_k, u_k, j) + \min_{u_{k+1}} Q_0^j(x_{k+1}, u_{k+1}) \right) \\
&\quad = \min_{u_k} \Bigg(U(x_k, u_k, j) + \min_{u_{k+1}} \Big(U(x_{k+1}, u_{k+1}, j-1) + \cdots \\
&\qquad + \min_{u_{k+j}} \left(U(x_{k+j}, u_{k+j}, 0) + \min_{u_{k+j+1}} \Psi(x_{k+j+1}, u_{k+j+1}) \right) \Big) \Bigg) \\
&\quad = \min_{(u_k, u_{k+1}, \ldots, u_{k+j})} \Bigg(\sum_{l=0}^{j} U(x_{k+l}, u_{k+l}, j-l) \\
&\qquad + \min_{u_{k+j+1}} \Psi\left(x_{k+j+1}, u_{k+j+1}\right) \Bigg).
\end{aligned} \tag{10.26}$$

Let $j = 23$. According to (10.11) and (10.14), we have

$$\begin{aligned}
\min_{u_k} Q_0^{24}(x_k, u_k) &= \min_{\mathcal{U}_k} \left(\Pi\left(x_k, \mathcal{U}_k\right) + \min_{u_{k+\lambda}} \Psi(x_{k+\lambda}, u_{k+\lambda}) \right) \\
&= \min_{\mathcal{U}_k} Q_0(x_k, \mathcal{U}_k).
\end{aligned} \tag{10.27}$$

The conclusion holds for $i = 0$. Assume that the conclusion holds for $i = \tau - 1$, i.e.,

$$\min_{\mathcal{U}_k} Q_{\tau-1}(x_k, \mathcal{U}_k) = \min_{u_k} Q_{\tau-1}^{24}(x_k, u_k). \tag{10.28}$$

Then for $i=\tau$, we have

$$\begin{aligned}
&\min_{u_k} Q_\tau^{24}(x_k,u_k)\\
&\quad=\min_{u_k}\left(U(x_k,u_k,23)+\min_{u_{k+1}} Q_\tau^{23}(x_{k+1},u_{k+1})\right)\\
&\quad=\min_{u_k}\Bigg(U(x_k,u_k,23)+\min_{u_{k+1}}\Bigg(U(x_{k+1},u_{k+1},22)\\
&\quad+\cdots+\min_{u_{k+23}}\left(U(x_{k+23},u_{k+23},0)+\min_{u_{k+\lambda}} Q_\tau^0(x_{k+\lambda},u_{k+\lambda})\right)\Bigg)\Bigg)\\
&\quad=\min_{\mathcal{U}_k}\left(\Pi\left(x_k,\mathcal{U}_k\right)+\min_{u_{k+\lambda}} Q_{\tau-1}^{24}(x_{k+\lambda},u_{k+\lambda})\right)\\
&\quad=\min_{\mathcal{U}_k}\left(\Pi\left(x_k,\mathcal{U}_k\right)+\min_{\mathcal{U}_{k+\lambda}} Q_{\tau-1}(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right)\\
&\quad=\min_{\mathcal{U}_k} Q_\tau(x_k,\mathcal{U}_k). \qquad (10.29)
\end{aligned}$$

The mathematical induction is complete. □

From Theorem 10.1, we can obtain the following corollary.

Corollary 10.1. *Let $\mu(x_k)$ be an arbitrary control law. For $i=0,1,\ldots$ and $j=0,1,\ldots,23$, define a new performance index function as*

$$\mathcal{Q}_i^{j+1}(x_k,u_k)=U(x_k,u_k,j)+\mathcal{Q}_i^j(x_{k+1},\mu(x_{k+1})), \qquad (10.30)$$

and define $Q_i^{j+1}(x_k,u_k)$ as in (10.23). *For $i=0,1,\ldots,$ let*

$$\mathcal{Q}_i^0(x_k,u_k)=Q_i^0(x_k,u_k). \qquad (10.31)$$

Then, for $j=0,1,\ldots,23,$ we have

$$Q_i^j(x_k,u_k)\le\mathcal{Q}_i^j(x_k,u_k). \qquad (10.32)$$

From Theorem 10.1 and Corollary 10.1, for $i=0,1,\ldots,$ we can say that the total cost in each period can be minimized by the iterative control law sequence $\mathcal{U}_i(x_k)$ according to the local iteration (10.19)–(10.24). Next, the convergence property of the global iteration will be developed.

Theorem 10.2. *For $i = 0, 1, \ldots,$ let $Q_{i+1}(x_k, \mathcal{U}_k)$ and $\mathcal{U}_i(x_k)$ be obtained by the global iteration* (10.15)–(10.18)*. Then, the iterative Q function $Q_i(x_k, \mathcal{U}_k)$ converges to the optimal Q function, i.e.,*

$$\lim_{i \to \infty} Q_i(x_k, \mathcal{U}_k) = Q^*(x_k, \mathcal{U}_k). \tag{10.33}$$

Proof. For functions $Q^*(x_k, \mathcal{U}_k)$, $\Pi(x_k, \mathcal{U}_k)$, and $Q_0(x_k, \mathcal{U}_k)$, let $\underline{\gamma}$, $\overline{\gamma}$, $\underline{\delta}$ and $\overline{\delta}$ be constants that satisfy

$$\underline{\gamma}\, \Pi(x_k, \mathcal{U}_k) \leq \min_{\mathcal{U}_{k+\lambda}} Q^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda}) \leq \overline{\gamma}\, \Pi(x_k, \mathcal{U}_k), \tag{10.34}$$

and

$$\underline{\delta} Q^*(x_k, \mathcal{U}_k) \leq Q_0(x_k, \mathcal{U}_k) \leq \overline{\delta} Q^*(x_k, \mathcal{U}_k), \tag{10.35}$$

respectively, where $0 < \underline{\gamma} \leq \overline{\gamma} < \infty$ and $0 \leq \underline{\delta} \leq \overline{\delta} < \infty$. Since $Q^*(x_k, \mathcal{U}_k)$ is unknown, the values of $\underline{\gamma}$, $\overline{\gamma}$, $\underline{\delta}$ and $\overline{\delta}$ cannot be obtained directly. In the following, we will prove that for arbitrary constants $\underline{\gamma}$, $\overline{\gamma}$, $\underline{\delta}$ and $\overline{\delta}$, the iterative Q function $Q_i(x_k, \mathcal{U}_k)$ will converge to the optimum and the estimations for the values of these constants can be omitted. The proof proceeds in four steps. First, we show that if $0 \leq \underline{\delta} \leq \overline{\delta} < 1$, then for $i = 0, 1, \ldots,$ the iterative performance index function $Q_i(x_k, \mathcal{U}_k)$ satisfies

$$\begin{aligned} &\left(1 + \frac{\underline{\delta} - 1}{(1 + \overline{\gamma}^{-1})^i}\right) Q^*(x_k, \mathcal{U}_k) \leq Q_i(x_k, \mathcal{U}_k) \\ &\leq \left(1 + \frac{\overline{\delta} - 1}{(1 + \underline{\gamma}^{-1})^i}\right) Q^*(x_k, \mathcal{U}_k). \end{aligned} \tag{10.36}$$

The inequality (10.36) can be proven by mathematical induction. Let $i = 0$ and we have

$$\begin{aligned} Q_1(x_k, \mathcal{U}_k) &= \Pi(x_k, \mathcal{U}_k) + \min_{\mathcal{U}_{k+\lambda}} \left\{ Q_0(x_{k+\lambda}, \mathcal{U}_{k+\lambda}) \right\} \\ &\geq \Pi(x_k, \mathcal{U}_k) + \underline{\delta} \min_{\mathcal{U}_{k+\lambda}} \left\{ Q^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda}) \right\} \end{aligned}$$

$$\geq \left(1+\overline{\gamma}\frac{\underline{\delta}-1}{1+\overline{\gamma}}\right)\Pi\left(x_k,\mathcal{U}_k\right)$$
$$+\left(\underline{\delta}-\frac{\underline{\delta}-1}{1+\overline{\gamma}}\right)\min_{\mathcal{U}_{k+\lambda}}\left\{Q^*(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\}$$
$$=\left(1+\frac{\underline{\delta}-1}{(1+\overline{\gamma}^{-1})}\right)\left\{\Pi\left(x_k,\mathcal{U}_k\right)+\min_{\mathcal{U}_{k+\lambda}}\left\{Q^*(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\}\right\}$$
$$=\left(1+\frac{\underline{\delta}-1}{(1+\overline{\gamma}^{-1})}\right)Q^*(x_k,\mathcal{U}_k). \tag{10.37}$$

From the idea of (10.37), we can also get $Q_1(x_k,\mathcal{U}_k) \leq \left(1+\frac{\overline{\delta}-1}{(1+\underline{\gamma}^{-1})}\right)Q^*(x_k,\mathcal{U}_k)$. Thus, (10.36) holds for $i=0$. Assume that (10.36) holds for $i=l-1$, $l=1,2,\ldots$. Then for $i=l$, we have

$$Q_{l+1}(x_k,\mathcal{U}_k)$$
$$=\Pi\left(x_k,\mathcal{U}_k\right)+\min_{\mathcal{U}_{k+\lambda}}\left\{Q_l(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\}$$
$$\geq \Pi\left(x_k,\mathcal{U}_k\right)+\left(1+\frac{\underline{\delta}-1}{(1+\overline{\gamma}^{-1})^{l-1}}\right)\min_{\mathcal{U}_{k+\lambda}}\left\{Q^*(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\}$$
$$\geq \left(1+\frac{\underline{\delta}-1}{(1+\overline{\gamma}^{-1})^{l}}\right)\left\{\Pi\left(x_k,\mathcal{U}_k\right)+\min_{\mathcal{U}_{k+\lambda}}\left\{Q^*(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\}\right\}$$
$$=\left(1+\frac{\underline{\delta}-1}{(1+\overline{\gamma}^{-1})^{l}}\right)Q^*(x_k,\mathcal{U}_k). \tag{10.38}$$

We can also get

$$Q_{l+1}(x_k,\mathcal{U}_k)$$
$$=\Pi\left(x_k,\mathcal{U}_k\right)+\min_{\mathcal{U}_{k+\lambda}}\left\{Q_l(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\}$$
$$\leq \left(1+\frac{\overline{\delta}-1}{(1+\underline{\gamma}^{-1})^{l}}\right)\left\{\Pi\left(x_k,\mathcal{U}_k\right)+\min_{\mathcal{U}_{k+\lambda}}\left\{Q^*(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\}\right\}$$
$$=\left(1+\frac{\overline{\delta}-1}{(1+\underline{\gamma}^{-1})^{l}}\right)Q^*(x_k,\mathcal{U}_k). \tag{10.39}$$

Hence, we obtain that (10.36) holds for $i = 0, 1, \ldots$ The mathematical induction is complete. Second, we show that if $0 \leq \underline{\delta} \leq 1 \leq \overline{\delta} < \infty$, then the iterative Q function $Q_i(x_k, \mathcal{U}_k)$ satisfies

$$\left(1+\frac{\underline{\delta}-1}{(1+\overline{\gamma}^{-1})^i}\right)Q^*(x_k,\mathcal{U}_k) \leq Q_i(x_k,\mathcal{U}_k) \leq \left(1+\frac{\overline{\delta}-1}{(1+\overline{\gamma}^{-1})^i}\right)Q^*(x_k,\mathcal{U}_k). \tag{10.40}$$

The left hand side of (10.40) can be proven according to (10.37). Now, we prove the right-hand side of (10.40) by mathematical induction. Let $i = 0$. We have

$$\begin{aligned} &Q_1(x_k,\mathcal{U}_k) \\ &\quad = \Pi(x_k,\mathcal{U}_k) + \min_{\mathcal{U}_{k+\lambda}} \left\{Q_0(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\} \\ &\quad \leq \Pi(x_k,\mathcal{U}_k) + \overline{\delta} \min_{\mathcal{U}_{k+\lambda}} \left\{Q^*(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\} \\ &\qquad + \frac{\overline{\delta}-1}{(1+\overline{\gamma})}\left(\overline{\gamma}\,\Pi(x_k,\mathcal{U}_k) - \min_{\mathcal{U}_{k+\lambda}} \left\{Q^*(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\}\right) \\ &\quad \leq \left(1+\frac{\overline{\delta}-1}{(1+\overline{\gamma}^{-1})}\right) Q^*(x_k,\mathcal{U}_k). \end{aligned} \tag{10.41}$$

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

$$\begin{aligned} &Q_{l+1}(x_k,\mathcal{U}_k) \\ &\quad = \Pi(x_k,\mathcal{U}_k) + \min_{\mathcal{U}_{k+\lambda}} \left\{Q_0(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\} \\ &\quad \leq \Pi(x_k,\mathcal{U}_k) + \left(1+\frac{\overline{\delta}-1}{(1+\overline{\gamma}^{-1})^{l-1}}\right) \min_{\mathcal{U}_{k+\lambda}} \left\{Q^*(x_{k+\lambda},\mathcal{U}_{k+\lambda})\right\} \\ &\quad \leq \left(1+\frac{\overline{\delta}-1}{(1+\overline{\gamma}^{-1})^{l}}\right) Q^*(x_k,\mathcal{U}_k). \end{aligned} \tag{10.42}$$

Hence, we obtain that (10.40) holds for $i = 0, 1, \ldots$ The mathematical induction is complete. Third, for the situation $1 \leq \underline{\delta} \leq \overline{\delta} < \infty$, according to (10.37)–(10.39), we can prove that for $i = 0, 1, \ldots$, the iterative performance index function $Q_i(x, \mathcal{U})$ satisfies (10.36).

Finally, considering the three situations above, for arbitrary constants $\underline{\gamma}$, $\overline{\gamma}$, $\underline{\delta}$ and $\overline{\delta}$, according to (10.36) and (10.40), we can easily obtain (10.33), as $i \to \infty$. □

Corollary 10.2. *For $i = 0, 1, \ldots$, let $Q_{i+1}(x_k, \mathcal{U}_k)$ and $\mathcal{U}_i(x_k)$ be obtained by global iteration* (10.15)–(10.18). *Then the iterative control law sequence $\mathcal{U}_i(x_k)$ converges to the optimal control law sequence, i.e.,* $\lim_{i\to\infty} \mathcal{U}_i(x_k) = \mathcal{U}^*(x_k)$.

10.4 Neural Network Implementation for the Dual Iterative *Q*-Learning Algorithm

In this section, neural networks are introduced to implement the dual iterative Q-learning algorithm. There are two neural networks, which are critic and action networks, respectively, in the dual iterative Q-learning algorithm. Both neural networks are chosen as three-layer back-propagation (BP) networks. The whole structure diagram is shown in Fig. 10.2.

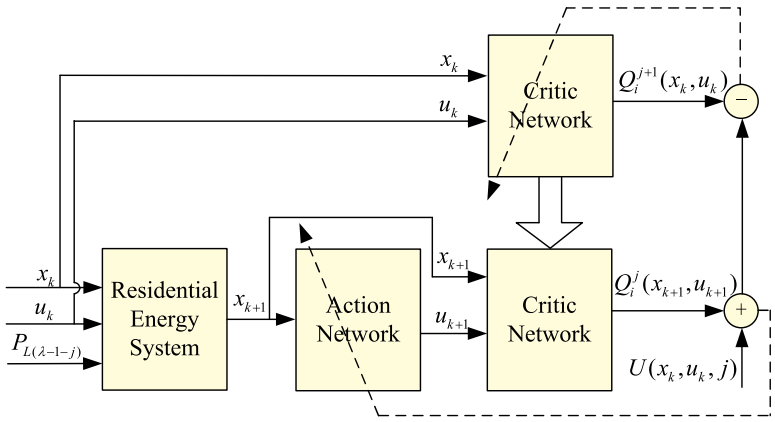

Fig. 10.2. The structure diagram of the dual iterative Q-learning algorithm.

10.4.1 *The action network*

For $i = 0, 1, \ldots$, the role of the action network is to approximate the iterative control law sequence $\mathcal{U}_i(x_k)$ defined in (10.17). To obtain the sequence, the action network should be trained to obtain the iterative control in local iteration. According to (10.19)–(10.23), the target of the action network can be defined as (10.20) and (10.22), where constraints in (10.3) and (10.4) are considered.

The action network can be constructed by 2 input neurons, 10 sigmoidal hidden neurons and 1 linear output neuron. Let $l = 0, 1, \ldots$ be the training step. Then, the output of the action network can be expressed as

$$\hat{u}_i^{j,l}(x_k) = W_{ai}^{jT}(l)\sigma(\mathcal{Z}_a(x_k)), \tag{10.43}$$

where $\mathcal{Z}_a(x) = Y_a^T x_k$ and $\sigma(\cdot)$ is a sigmoid function. Let $W_{ai}^j(l)$ and Y_a be the hidden-output and input-hidden weight matrices, respectively. To enhance the training speed, only the hidden-output weight $W_{ai}^j(l)$ is updated during the neural network training, while the input-hidden weight is fixed. The action network weight update is expressed as follows

$$W_{ai}^j(l+1) = W_{ai}^j(l) - \beta_a \left[\frac{\partial E_{ai}^j(l)}{\partial W_{ai}^j(l)}\right], \tag{10.44}$$

where $E_{ai}^j(l) = \frac{1}{2}(e_{ai}^j(l))^2$ and $e_{ai}^j(l) = \hat{u}_i^{j,l}(x_k) - u_i^j(x_k)$. Let $\beta_a > 0$ be the learning rate of the action network.

10.4.2 *The critic network*

For $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, 23$, the goal of the critic network is to obtain $Q_i(x_k, \mathcal{U}_k)$ by updating $Q_i^{j+1}(x_k, u_k)$ in (10.23), iteratively. The critic network can be constructed by 3 input neurons, 15 sigmoidal hidden neurons and 1 linear output neuron. Let $Z_{ck} = [x_k^T, u_k]^T$ be the input vector of the critic network. Then, the output of the critic network can be expressed as

$$\hat{Q}_i^{j+1,l}(x_k, u_k) = W_{ci}^{jT}(l)\sigma(\mathcal{Z}_{ck}), \tag{10.45}$$

where $\mathcal{Z}_{ck} = Y_c^T Z_{ck}$ and $\sigma(\cdot)$ is a sigmoid function. Let $W_{ci}^j(l)$ and Y_c be the hidden-output and input-hidden weight matrices, respectively. During the neural network training, the hidden-output weight

$W_{ci}^{j}(l)$ is updated, while the input-hidden weight Y_c is fixed. The critic network weight update is expressed as follows

$$W_{ci}^{j}(l+1) = W_{ci}^{j}(l) - \alpha_c \left[\frac{\partial E_{ci}^{j}(l)}{\partial W_{ci}^{j}(l)} \right], \tag{10.46}$$

where $E_{ci}^{j}(l) = \frac{1}{2}(e_{ci}^{j}(l))^2$ and $e_{ci}^{j}(l) = \hat{Q}_i^{j+1,l}(x_k, u_k) - Q_i^{j+1}(x_k, u_k)$. Let $\alpha_c > 0$ be the learning rate of the critic network.

10.4.3 *Training phase*

In this subsection, the dual iterative Q-learning algorithm implemented by action and critic networks is explained step by step and shown in Algorithm 10.1.

Algorithm 10.1 Dual iterative Q-learning algorithm.

Initialization:

1: Collect an array of system data for the residential energy system (10.8).
2: Give a positive semi-definite function $\Psi(x_k, u_k)$.
3: Give the computation precision $\varepsilon > 0$.

Iteration:

4: Let $i = 0$.
5: For $j = 0$, let $Q_0^0(x_k, u_k) = \Psi(x_k, u_k)$.
6: For $j = 0, 1, \ldots, 23$, train the action and critic networks to obtain $u_0^j(x_k)$ and $Q_0^{j+1}(x_k, u_k)$ that satisfy (10.20) and (10.21), respectively.
7: Let $Q_0(x_k, \mathcal{U}_k) = Q_0^{24}(x_k, u_k)$. Obtain $\mathcal{U}_0(x_k)$ and $Q_1(x_k, \mathcal{U}_k)$ by (10.15) and (10.16), respectively.
8: Let $i = i + 1$.
9: For $j = 0, 1, \ldots, 23$, train the action and critic networks to obtain $u_i^j(x_k)$ and $Q_i^{j+1}(x_k, u_k)$ that satisfy (10.22) and (10.23), respectively.
10: Let $Q_i(x_k, \mathcal{U}_k) = Q_i^{24}(x_k, u_k)$. Obtain $\mathcal{U}_i(x_k)$ and $Q_{i+1}(x_k, \mathcal{U}_k)$ by (10.17) and (10.18), respectively.
11: If $|Q_i(x_k, \mathcal{U}_k) - Q_{i-1}(x_k, \mathcal{U}_k)| \leq \varepsilon$, then goto next step. Otherwise, goto Step 8.
12: For $j = 0, 1, \ldots, 23$, solve $u_i^j(x_k)$ by (10.22) and obtain $\mathcal{U}_i(x_k) = (u_i^0(x_k), \ldots, u_i^{23}(x_k))$.
13: **return** $Q_i(x_k, \mathcal{U}_k)$ and $\mathcal{U}_i(x_k)$.

10.5 Simulation Example

In this section, the performance of the dual iterative Q-learning algorithm will be examined by numerical experiments. The profiles of the residential load demand (kW) and the real-time electricity rate (cents) for one week (168 hours) are shown in Figs. 10.3(a) and (c), respectively. We can see that the residential load demand and the real-time electricity rate are both periodic-like functions with the period $\lambda = 24$. The average trajectories of the residential load demand and the electricity rate are shown in Figs. 10.3(b) and (d).

We assume that the supply from the power grid guarantees the residential load demand at any time. Define the capacity of the

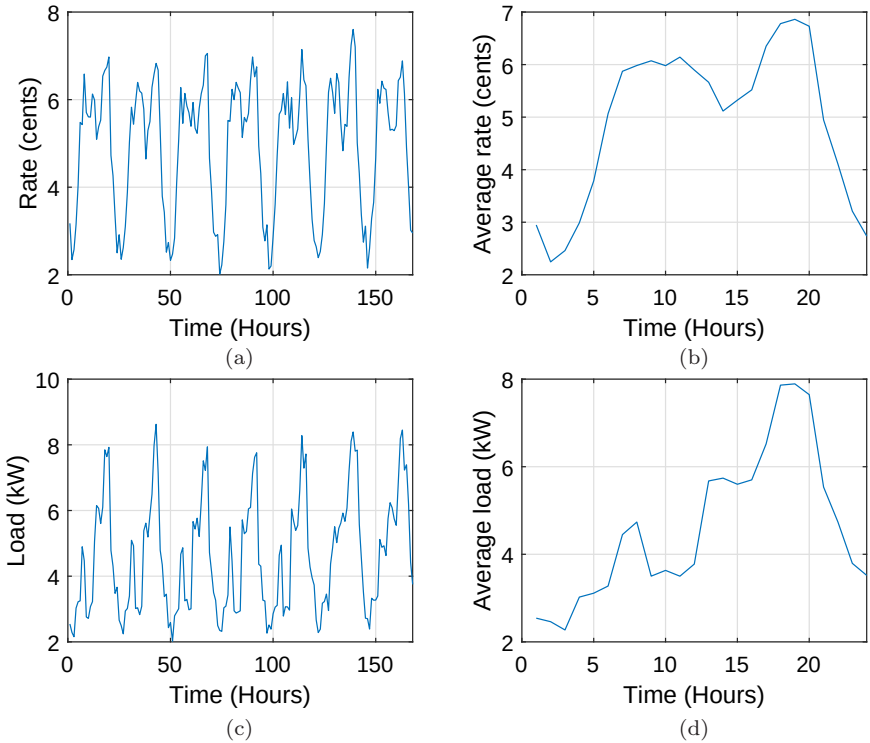

Fig. 10.3. Residential load demand and electricity rate. (a) Residential load demand for 168 hours. (b) Average residential load demand. (c) Real-time electricity rate for 168 hours. (d) Average electricity rate.

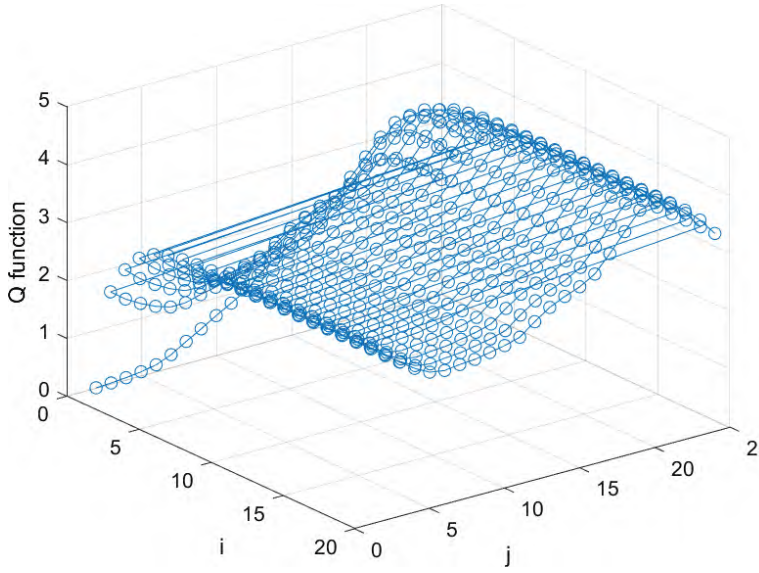

Fig. 10.4. The trajectory of the iterative Q function.

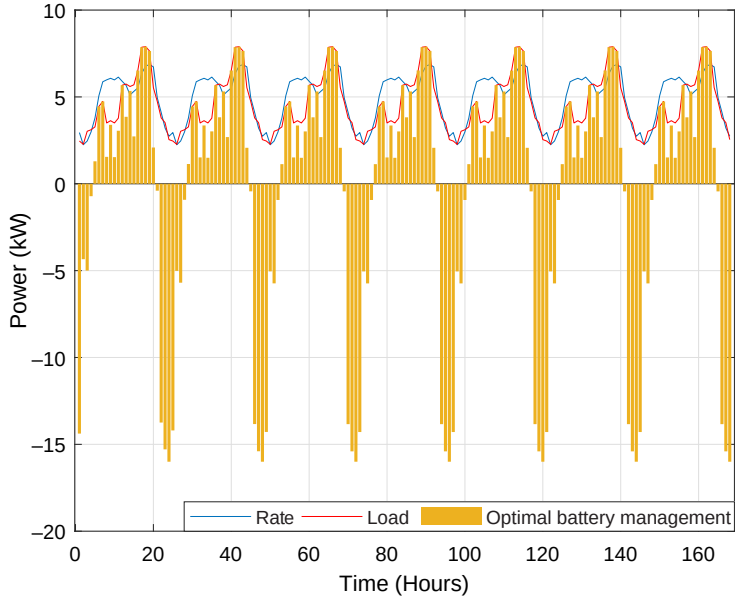

Fig. 10.5. Optimal control of the battery in one week.

battery as 100 kWh. Let the upper and lower storage limits of the battery be $E_b^{\min} = 20$ kWh and $E_b^{\max} = 80$ kWh, respectively. The rated power output of the battery and the maximum charge/discharge rate is 16 kW. The initial level of the battery is 30 kWh. Let the performance index function be expressed as in (10.7), where we set $m_1 = 1$, $m_2 = 0.2$ and $r = 0.1$. Let the initial function $\Psi(x_k, u_k) = [x_k^T, u_k] I [x_k^T, u_k]^T$, where I is the identity matrix with a suitable dimension. Let the initial state be $x_0 = [8, 30]^T$. After normalizing the data of the residential load demand and the electricity rate, we implement the developed dual iterative Q-learning algorithm by neural networks for $i = 20$ iterations to guarantee the computation precision $\varepsilon = 10^{-4}$. The learning rates of the action and critic networks are 0.01 and the training precisions of the neural networks are 10^{-6}. Let $Q_i^j(x_0, \bar{u}) = \min_u Q_i^j(x_0, u)$. The trajectory of $Q_i^j(x_0, \bar{u})$ is shown in Fig. 10.4, where we can see that the iterative Q function is convergent to the optimum after $i = 20$ iterations. According to the one week's residential load demand and electricity rate, the optimal control of the battery is shown in Fig. 10.5.

10.6 Conclusion

In this chapter, a dual iterative Q-learning algorithm is developed to solve the optimal battery management and control problem in smart residential environments. Global and local iterations are introduced in the developed algorithm. The convergence property of the dual iterative Q-learning algorithm is developed for the first time to guarantee that both the iterative Q function and the iterative control law sequence reach the optimum. Neural networks are introduced to implement the developed dual iterative Q-learning algorithm. Finally, numerical results are given to illustrate the superiority of the developed algorithm.

References

[1] Huang, T. and Liu, D. (2013). A self-learning scheme for residential energy system control and management. *Neural Computing and Applications* 22(2), 259–269.

[2] Lee, T.Y. (2007). Operating schedule of battery energy storage system in a time-of-use rate industrial user with wind turbine generators: a multipass iteration particle swarm optimization approach. *IEEE Transactions on Energy Conversion* 22(3), 774–782.

[3] Yau, T., Walker, L.N., Graham, H.L., and Raithel, R. (1981). Effects of battery storage devices on power system dispatch. *IEEE Transactions on Power Apparatus and Systems* PAS-100(1), 375–383.

Chapter 11

Mixed Iterative Adaptive Dynamic Programming for Optimal Battery Energy Control in Microgrids

11.1 Introduction

In this chapter, a novel mixed iterative ADP algorithm is developed to solve the optimal battery energy management and control problem in microgrid systems. Based on the data of the load and electricity rate, two iterations are constructed, which are P-iteration and V-iteration, respectively. The V-iteration is implemented based on value iteration, which aims to obtain the iterative control law sequence in each period. The P-iteration is implemented based on policy iteration, which updates the iterative value function according to the iterative control law sequence. Properties of the developed mixed iterative ADP algorithm are analyzed. It is shown that the iterative value function is monotonically non-increasing and converges to the solution of the HJB equation. In each iteration, it is proven that the performance index function is finite under the iterative control law sequence. Finally, numerical results are given to illustrate the performance of the developed algorithm.

11.2 Problem Formulation

In this section, the microgrid systems will be described. The performance index function will be defined and the optimality principle will be introduced.

11.2.1 *System descriptions*

The microgrid system discussed in this paper is described in Refs. [1, 2], which is composed of the power grid, the load, and the battery system. It is expected to optimize the energy management of the microgrid system by suitable designing the charging/discharging power of the battery. In this chapter, the optimal battery management and control problem is treated as a discrete time problem with the time step of 1 hour and it is assumed that the load of the user varies hourly. The battery will make decisions to meet the demand of the residential load and according to the real-time electricity rate. There are three operational modes for the battery of the microgrid system.

- **Charging mode:** when the load is low and the electricity rate is inexpensive, the power grid will supply the load directly and, at the same time, charge the battery;
- **Idle mode:** the power grid will directly supply the load at certain hours while the battery energy remains fixed;
- **Discharging mode:** the battery supplies the load at hours when the load is high and the electricity rate is expensive.

The battery model used in this chapter is based on [1, 2], where battery efficiency is considered to extend the battery's lifetime as far as possible. Let E_{bt} be the battery energy at time t and let $\eta(\cdot)$ be the charging/discharging efficiency of the battery. Then, the battery model can be expressed as

$$E_{b(t+1)} = E_{bt} - P_{bt} \times \eta(P_{bt}), \tag{11.1}$$

where P_{bt} is the battery power output at time t. Let $P_{bt} > 0$ denote battery discharging. Let $P_{bt} < 0$ denote battery charging and let $P_{bt} = 0$ denote that the battery is idle. The efficiency of battery

charging/discharging is derived by Refs. [1, 2], which is expressed as

$$\eta(P_{bt}) = 0.898 - 0.173|P_{bt}|/P_{\text{rate}}, \tag{11.2}$$

where $P_{\text{rate}} > 0$ is the rated power output of the battery. To extend the battery's lifetime, two constraints need to be considered. The storage limits satisfy $E_b^{\min} \leq E_{bt} \leq E_b^{\max}$, where $E_b^{\min}$ and $E_b^{\max}$ are the minimum and maximum storage energy of the battery, respectively. The charging and discharging power limits satisfy $P_b^{\min} \leq P_{bt} \leq P_b^{\max}$, where $P_b^{\min}$ and $P_b^{\max}$ are the minimum and maximum charging/discharging powers of the battery, respectively.

11.2.2 *Optimization objectives*

First, inspired by Ref. [2], we define the system states and control. Let $x_{1t} = P_{gt}$ and $x_{2t} = E_{bt} - E_b^o$ be the two system states, where P_{gt} is the power from the power grid and E_b^o is the middle storage limit of the battery. Let the system control be the charging/discharging power of the battery, i.e., $u_t = P_{bt}$. Letting $x_t = [x_{1t}, x_{2t}]^{\mathsf{T}}$ be the system state and letting P_{Lt} be the power of the load, the equation of the microgrid system is derived as

$$x_{t+1} = F(x_t, u_t, t), \tag{11.3}$$

where the system function is expressed as $F(x_t, u_t, t) = \begin{pmatrix} P_{Lt} - u_t \\ x_{2t} - u_t\eta(u_t) \end{pmatrix}$. Let $\underline{u}_t = (u_t, u_{t+1}, \ldots)$ denote the control sequence from t to ∞. Let x_0 be the initial state. Then, the total performance index function expected to be minimized can be written as

$$J(x_0, \underline{u}_0, 0) = \sum_{t=0}^{\infty} \gamma^t \Big(m_1 (C_t x_{1t})^2 + m_2 x_{2t}^2 + r u_t^2 \Big), \tag{11.4}$$

where C_t is the electricity rate, and m_1, m_2, r are given positive constants. Let γ denote the discount factor, which satisfies $0 < \gamma < 1$. The physical meaning of the first term of the performance index function is to minimize the total cost from the grid. The second term aims to make the stored energy of the battery close to the middle of storage limit, which avoids fully charging/discharging of the battery. The third term is to prevent large charging/discharging power of the

battery. Hence, the second and third terms aim to extend the lifetime of the battery. Let $M_t = \begin{bmatrix} m_1 C_t^2 & 0 \\ 0 & m_2 \end{bmatrix}$ and let the utility function be $U(x_t, u_t, t) = x_t^T M_t x_t + r u_t^2$. Then, the performance index function (11.4) can be expressed as

$$J(x_0, \underline{u}_0, 0) = \sum_{t=0}^{\infty} \gamma^t U(x_t, u_t, t). \tag{11.5}$$

Generally, the functions of the load and the electricity rate are periodic. For convenience of analysis, in this chapter, our discussion is based on the following assumption.

Assumption 11.1. The residential load P_{Lt} and the electricity rate C_t are periodic functions with the period $\lambda = 24$ hours.

Define the control sequence set as $\underline{\mathfrak{A}}_t = \left\{\underline{u}_t \colon \underline{u}_t = (u_t, u_{t+1}, \ldots), u_{t+i} \in \mathbb{R}, i = 0, 1, \ldots\right\}$. Then, for an arbitrary control sequence $\underline{u}_t \in \underline{\mathfrak{A}}_t$, the optimal performance index function can be defined as

$$J^*(x_t, t) = \inf_{\underline{u}_t} \left\{J(x_t, \underline{u}_t, t) \colon \underline{u}_t \in \underline{\mathfrak{A}}_t\right\}. \tag{11.6}$$

According to Bellman's principle of optimality, we can obtain the following discrete-time Bellman equation

$$J^*(x_t, t) = \min_{u_t} \left\{U(x_t, u_t, t) + \gamma J^*(x_{t+1}, t+1)\right\}. \tag{11.7}$$

Generally, the optimal performance index function $J^*(x_t, t)$ is a nonlinear non-analytic function and it is almost impossible to obtain the optimal control by directly solving equation (11.7). In Ref. [2], a dual iterative Q-learning algorithm was developed to obtain the optimal solution indirectly, which was proven that the iterative Q function converged to the optimum, as the iteration index increased to infinity. However, properties of the iterative control laws were not considered in Ref. [2]. It implied that the dual iterative Q-learning algorithm could not stop until infinite iterations to reach the optimum. Hence, the computation efficiency of dual iterative Q-learning algorithm is low. To overcome this disadvantage, a new mixed iterative ADP algorithm will be developed.

11.3 Mixed Iterative ADP Algorithm

In this section, inspired by Ref. [2], a new mixed iterative iterative ADP algorithm will be developed to obtain the optimal control law for microgrid systems. New property analysis will also be developed in this section.

11.3.1 *Derivations*

As P_{Lt} and C_t are both periodic functions with the period $\lambda = 24$ hours, for any $t = 0, 1, \ldots$, there exist $\varrho = 0, 1, \ldots$ and $\theta = 0, 1, \ldots, 23$ that satisfy $t = \varrho\lambda + \theta$. Let $k = \varrho\lambda$. Then, we have $P_{Lt} = P_{L(k+\theta)} = P_{L\theta}$ and $C_t = C_{k+\theta} = C_\theta$, respectively. Define $\mathscr{U}_k$ as the control sequence from k to $k+\lambda-1$, i.e., $\mathscr{U}_k = (u_k, u_{k+1}, \ldots, u_{k+\lambda-1})$. Then $\forall\, k \in \{0, \lambda, 2\lambda, \ldots\}$, we can define a new utility function as

$$\Pi\left(x_k, \mathscr{U}_k\right) = \sum_{\theta=0}^{\lambda-1} \gamma^\theta U(x_{k+\theta}, u_{k+\theta}, \theta). \tag{11.8}$$

Letting $\mathcal{J}^*(x_k) = J^*(x_k, k)$, the Bellman equation (11.7) can be derived as

$$\mathcal{J}^*(x_k) = \min_{\mathscr{U}_k} \left\{\Pi\left(x_k, \mathscr{U}_k\right) + \tilde{\gamma}\mathcal{J}^*(x_{k+\lambda})\right\}, \tag{11.9}$$

where $\tilde{\gamma} = \gamma^\lambda$. The optimal control law sequence can be expressed by

$$\mathscr{U}^*(x_k) = \arg\min_{\mathscr{U}_k} \left\{\Pi\left(x_k, \mathscr{U}_k\right) + \tilde{\gamma}\mathcal{J}^*(x_{k+\lambda})\right\}. \tag{11.10}$$

Let $\mu^l(x_k)$, $l = 0, 1, \ldots, \lambda - 1$ be the control law in hour l. Let $\Lambda(x_k) = \{\mu^0(x_k), \mu^1(x_k), \ldots, \mu^{\lambda-1}(x_k)\}$ be a control law sequence in a period, which makes the following performance index function

$$\mathcal{P}(x_k) = \sum_{\tau=0}^{\infty} \gamma^{\lambda\tau} \Pi(x_{k+\lambda\tau}, \Lambda(x_{k+\lambda\tau})) \tag{11.11}$$

finite. Let $\mathfrak{U}(x_k)$ be a set of control law sequences, which is defined as

$$\mathfrak{U}(x_k) = \left\{ \Lambda(x_k) \,\middle|\, \sum_{\tau=0}^{\infty} \tilde{\gamma}^\tau \Pi(x_{k+\lambda\tau}, \Lambda(x_{k+\lambda\tau})) < \infty \right\}. \tag{11.12}$$

Based on the above preparations, the new mixed iterative ADP algorithm, which is referred to HDP, can be developed. There are two iterations in the developed algorithm, which are V-iteration and P-iteration, respectively. Let $i = 0, 1, \ldots$ be the P-iteration index. For $i = 0$. Let

$$\mathcal{U}_0(x_k) = \left\{u_0^0(x_k), u_0^1(x_k), \ldots, u_0^{23}(x_k)\right\} \tag{11.13}$$

be an arbitrary control law sequence, which satisfies $\mathcal{U}_0(x_k) \in \mathfrak{U}(x_k)$. Let $V_0(x_k)$ be the iterative value function constructed by the control law sequence $\mathcal{U}_0(x_k)$ that satisfies the following generalized Bellman equation

$$V_0(x_k) = \Pi\left(x_k, \mathcal{U}_0(x_k)\right) + \tilde{\gamma} V_0(x_{k+\lambda}). \tag{11.14}$$

The iterative control law sequence $\mathcal{U}_1(x_k)$ can be computed as

$$\mathcal{U}_1(x_k) = \arg\min_{\mathscr{U}_k} \left\{\Pi\left(x_k, \mathscr{U}_k\right) + \tilde{\gamma} V_0(x_{k+\lambda})\right\}. \tag{11.15}$$

For $i = 1, 2, \ldots$, let $V_i(x_k)$ be the iterative value function constructed by the control law sequence $\mathcal{U}_i(x_k)$ that satisfies the following generalized Bellman equation

$$V_i(x_k) = \Pi\left(x_k, \mathcal{U}_i(x_k)\right) + \tilde{\gamma} V_i(x_{k+\lambda}). \tag{11.16}$$

Then, we update the iterative control law sequence $\mathcal{U}_{i+1}(x_k)$ by

$$\mathcal{U}_{i+1}(x_k) = \arg\min_{\mathscr{U}_k} \left\{\Pi\left(x_k, \mathscr{U}_k\right) + \tilde{\gamma} V_i(x_{k+\lambda})\right\}. \tag{11.17}$$

Equations (11.13)–(11.17) can be called as the P-iteration. For any P-iteration index $i = 0, 1, \ldots$, it requires the iterative control law sequence $\mathcal{U}_i(x_k)$ to construct the iterative value function $V_i(x_k)$. However, $\mathcal{U}_i(x_k)$ cannot be obtained by solving (11.15) and (11.17) directly. Thus, the V-iteration is implemented to obtain $\mathcal{U}_i(x_k)$ iteratively. For any $i = 0, 1, \ldots$, let $j = 0, 1, \ldots, \lambda - 1$ be the V-iteration index. For $i = 0, 1, \ldots$, and $j = 0$, let the initial value function be

$$\mathcal{V}_{i+1}^0(x_k) = V_i(x_k). \tag{11.18}$$

Then, for $j = 0, 1, \ldots, \lambda - 1$, the iterative control law is solved by

$$\begin{aligned} u_{i+1}^{j}(x_k) &= \arg\min_{u_k} \left\{ U(x_k, u_k, j) + \gamma \mathcal{V}_{i+1}^{j}(x_{k+1}) \right\} \\ &= \arg\min_{u_k} \left\{ U(x_k, u_k, j) + \gamma \mathcal{V}_{i+1}^{j}(\mathcal{F}(x_k, u_k, j)) \right\}, \end{aligned} \tag{11.19}$$

where $x_{k+1} = \mathcal{F}(x_k, u_k, j) = \begin{pmatrix} P_{L(\lambda-1-j)} - u_k \\ x_{2k} - u_k \eta(u_k) \end{pmatrix}$, $U(x_k, u_k, j) = x_k^T M_{\lambda-1-j} x_k + r u_k^2$, and $M_{\lambda-1-j} = \begin{bmatrix} m_1 C_{\lambda-1-j}^2 & 0 \\ 0 & m_2 \end{bmatrix}$. The iterative value function is updated by

$$\begin{aligned} \mathcal{V}_{i+1}^{j+1}(x_k) &= \min_{u_k} \left\{ U(x_k, u_k, j) + \gamma \mathcal{V}_{i+1}^{j}(x_{k+1}) \right\} \\ &= U(x_k, u_{i+1}^{j}(x_k), j) + \gamma \mathcal{V}_{i+1}^{j}(\mathcal{F}(x_k, u_{i+1}^{j}(x_k), j)). \end{aligned} \tag{11.20}$$

Then, we construct the iterative control law sequence by

$$\mathcal{U}_{i+1}(x_k) = \{ u_{i+1}^{\lambda-1}(x_k), u_{i+1}^{\lambda-2}(x_k), \ldots, u_{i+1}^{0}(x_k) \}. \tag{11.21}$$

Remark 11.1. For the microgrid system in this chapter, the load and electricity rate are different in each hour. Hence, the microgrid system (11.3) is a time-variant system. Then, the system control law for microgrid system is also time-variant, which implies that the optimal battery control law in each hour is also different. On the other hand, in Assumption 11.1, the residential load and the electricity rate are periodic functions with the period $\lambda = 24$ hours. Thus, we can find a periodical control sequence $\mathcal{U}_k$ that satisfies the Bellman equation can be derived as (11.9), where each control in the control sequence processes different control law. Thus, there are 24 control laws in the control sequence, i.e., $\mathcal{U}(x_k) = (u^0(x_k), u^1(x_k), \ldots, u^{23}(x_k))$. For any $j = 0, 1, \ldots, 23$, according to the state x_{k+j}, we can easily obtain the control input $u_{k+j} = u^j(x_{k+j})$. Hence, the mixed iterative ADP algorithm is initialized by an arbitrary control law sequence (11.13), instead of a single control law.

Remark 11.2. There are two blocks in the iterative ADP algorithm (11.13)–(11.21), which are P-iteration and V-iteration, respectively. The P-iteration (11.13)–(11.17) is implemented based on policy

iteration, which objective is to update the iterative value function that satisfies the generalized Bellman equation. This iteration procedure is called "policy evaluation" procedure [3]. The V-iteration (11.18)–(11.21) is implemented based on value iteration, which objective is to obtain the iterative control law sequence that minimizes the total cost in each period. From V-iteration, we can always use it to find another control law sequence that is better, or at least no worse. This iteration procedure is known as "policy improvement" procedure. As we can see that the developed iterative ADP algorithm contains both value and policy iterations, and the developed iterative ADP algorithm can be called mixed iterative ADP algorithm.

On the other hand, the policy evaluation and policy improvement procedures are different from the procedures in Ref. [3]. For the policy evaluation procedure in Ref. [3], the iterative value function is updated for a single control law. In this chapter, the iterative value function is updated under the iterative control law sequence. In the policy improvement procedure in Ref. [3], a single iterative control law is obtained. In the developed mixed iterative value function, we use (11.18)–(11.21) in the policy improvement procedure to update a new iterative control law sequence instead of a single control law. Thus, the policy evaluation and policy improvement procedures are different from the procedures in Ref. [3].

Remark 11.3. The developed mixed iterative ADP algorithm (11.13)–(11.21) is different from the dual iterative Q-learning algorithm [2]. First, the initial conditions are different. In Ref. [2], the dual iterative Q-learning algorithm initialized by a positive semi-definite function, while the mixed iterative ADP algorithm initialized by a control law sequence. Second, in each iteration of the dual iterative Q-learning algorithm, the iterative control law sequence is used to improve the iterative Q function, while in each P-iteration of the mixed iterative ADP algorithm, it is required to solve a generalized Bellman equation by the iterative control law sequence. Thus, the property analysis in Ref. [2] is not available for the developed mixed iterative ADP algorithm and new property analysis method will be constructed.

Remark 11.4. If we have obtained the iterative control law $\mathcal{U}_i(x_k)$, $i = 0, 1, \ldots$, we can construct an iterative value function $V_i(x_k)$ to satisfy generalized Bellman equation (11.14) and (11.16). The method can be explained as follows. Let $l = 0, 1, \ldots$ be an iteration

index. Letting $V_i^0(x_k) = V_{i-1}(x_k)$, for any $l = 0, 1, \ldots$, update the iterative value function by the following equation

$$V_i^{l+1}(x_k) = \Pi(x_k, \mathcal{U}_i(x_k)) + \tilde{\gamma} V_i^l(x_{k+\lambda}) \tag{11.22}$$

until the iterative value function is convergent, i.e., $V_i^{l+1}(x_k) = V_i^l(x_k)$. Then, we can obtain $V_i(x_k) = V_i^l(x_k)$. According to (11.22), the iterative control law sequence $\mathcal{U}_i(x_k)$ is kept unchanged. Hence, we say that in each P-iteration of the mixed iterative ADP algorithm, the iterative control law sequence is kept unchanged. However, in each iteration of the dual Q-learning algorithm [2], including i-iteration and j-iteration, the iterative control law of the system is updated for any i and j. Thus, we say that the P-iteration of the mixed iterative ADP algorithm is simpler than the dual Q-learning algorithm [2].

Remark 11.5. Generally, traditional ADP algorithms are investigated in the framework of Markov decision process which are implemented along time $k = 0, 1, \ldots$. For any time $k = 0, 1, \ldots$, the performance index function and control law are updated for the state x_k, where the state x_k is a stochastic state in the state space in the framework of Markov decision process. When the time increases to infinity, the performance index function and the control law are desired to satisfy the following HJB equation (11.7). Hence, we say that traditional ADP algorithms are implemented in stochastic process. On the other hand, in this chapter, we focus on deterministic systems. The developed mixed iterative ADP algorithm is implemented according to the iteration indices $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, 23$, which is not along to the time index k. In each iteration, it is required that the iterative control law and the iterative value function are updated for the whole state space instead of being updated for a single state x_k. Thus, we say that the developed mixed iterative ADP algorithm is obviously different from the traditional ADP algorithms.

11.3.2 *Properties*

In this subsection, the property of the V-iteration will first be shown. Then, the properties of the iterative control law sequences and the iterative value function will be proven.

Theorem 11.1. *For $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$, let the iterative value function $\mathcal{V}_{i+1}^{j}(x_k)$ and the iterative control law $u_{i+1}^{j}(x_k)$ be obtained by the V-iteration* (11.18)–(11.21)*. Then, the iterative control law sequence $\mathcal{U}_{i+1}(x_k)$ in* (11.21) *satisfies* (11.17) *for any $i = 0, 1, \ldots$.*

Proof. For any $i = 0, 1, \ldots$, according to (11.20), we can derive that

$$\mathcal{V}_{i+1}^{\lambda}(x_k) = \min_{u_k} \left\{ U(x_k, u_k, \lambda - 1) + \gamma \mathcal{V}_{i+1}^{\lambda-1}(x_{k+1}) \right\}, \tag{11.23}$$

where $U(x_k, u_k, \lambda - 1) = x_k^T \begin{bmatrix} m_1 (C_0)^2 & 0 \\ 0 & m_2 \end{bmatrix} x_k + r u_k^2$, and $x_{k+1} = \mathcal{F}(x_k, u_k, \lambda - 1) = \begin{pmatrix} P_{L0} - u_k \\ x_{2k} - u_k \eta(u_k) \end{pmatrix}$. According to (11.20) and (11.23), we can derive

$$\begin{aligned}
\mathcal{V}_{i+1}^{\lambda}(x_k) &= \min_{u_k} \Big\{ U(x_k, u_k, \lambda - 1) + \gamma \min_{u_{k+1}} \Big\{ U(x_{k+1}, u_{k+1}, \lambda - 2) \\
&\quad + \gamma \mathcal{V}_{i+1}^{\lambda-2}(x_{k+2}) \Big\} \Big\} \\
&= \min_{u_k} \Big\{ U(x_k, u_k, \lambda - 1) + \gamma \min_{u_{k+1}} \Big\{ U(x_{k+1}, u_{k+1}, \lambda - 2) \\
&\quad + \cdots + \gamma \min_{u_{k+\lambda-1}} \Big\{ U(x_{k+\lambda-1}, u_{k+\lambda-1}, 0) \\
&\quad + \gamma \mathcal{V}_{i+1}^{0}(x_{k+\lambda}) \Big\} \cdots \Big\} \Big\} \\
&= \min_{\underline{u}_k^{k+\lambda-1}} \left\{ \sum_{j=0}^{\lambda-1} \gamma^j U(x_k, u_k, \lambda - j - 1) + \gamma^{\lambda} \mathcal{V}_{i+1}^{0}(x_{k+\lambda}) \right\} \\
&= \min_{\mathscr{U}_k} \Big\{ \Pi(x_k, \mathscr{U}_k) + \tilde{\gamma} V_i(x_{k+\lambda}) \Big\}.
\end{aligned} \tag{11.24}$$

Thus, we know that the iterative control law sequence $\mathcal{U}_{i+1}(x_k)$ in (11.21) satisfies (11.17) for any $i = 0, 1, \ldots$. □

Theorem 11.2. *Let the mixed iterative ADP algorithm be implemented according to* (11.13)–(11.21)*. For $i = 0, 1, \ldots$, if the iterative control law sequence $\mathcal{U}_i(x_k)$ satisfies $\mathcal{U}_i(x_k) \in \mathfrak{U}(x_k)$, where $\mathfrak{U}(x_k)$ is defined in* (11.12)*, then we have*

$$\mathcal{U}_{i+1}(x_k) \in \mathfrak{U}(x_k). \tag{11.25}$$

Proof. If $\mathcal{U}_i(x_k) \in \mathfrak{U}(x_k)$, then there exists an iterative value function $V_i(x_k)$ that satisfies (11.16). Let $\mathcal{P}_\ell(x_k)$, $\ell = 0, 1, \ldots$, be expressed as

$$\mathcal{P}_{\ell+1}(x_k) = \Pi\left(x_k, \mathcal{U}_{i+1}(x_k)\right) + \tilde{\gamma}\mathcal{P}_\ell(x_{k+\lambda}), \tag{11.26}$$

where $\mathcal{P}_0(x_k) = V_i(x_k)$. As $\mathcal{U}_{i+1}(x_k)$ satisfies (11.17), we know that

$$\begin{aligned}
\mathcal{P}_1(x_k) &= \Pi\left(x_k, \mathcal{U}_{i+1}(x_k)\right) + \tilde{\gamma}\mathcal{P}_0(x_{k+\lambda}) \\
&= \min_{\mathscr{U}_k}\{\Pi(x_k, \mathscr{U}_k) + \tilde{\gamma}V_i(x_{k+\lambda})\} \\
&\leq \Pi\left(x_k, \mathcal{U}_i(x_k)\right) + \tilde{\gamma}V_i(x_{k+\lambda}) \\
&= V_i(x_k).
\end{aligned} \tag{11.27}$$

Thus, $\mathcal{P}_1(x_k) \leq V_i(x_k)$. Using mathematical induction, we can derive

$$\mathcal{P}_{\ell+1}(x_k) \leq V_i(x_k) \tag{11.28}$$

for any $\ell = 0, 1, \ldots$. According to (11.26), for $\ell = 0, 1, \ldots$, we can derive that

$$\begin{aligned}
\mathcal{P}_{\ell+1}(x_k) = &\sum_{\tau=0}^{\ell} \tilde{\gamma}^\tau \Pi(x_{k+\lambda\tau}, \mathcal{U}_{i+1}(x_{k+\lambda\tau})) \\
&+ \tilde{\gamma}^{\ell+1} V_i(x_{k+\lambda(\ell+1)}).
\end{aligned} \tag{11.29}$$

Letting $\ell \to \infty$, we can derive

$$\sum_{\tau=0}^{\infty} \tilde{\gamma}^\tau \Pi(x_{k+\lambda\tau}, \mathcal{U}_{i+1}(x_{k+\lambda\tau})) \leq V_i(x_k) < \infty. \tag{11.30}$$

Thus, we have $\mathcal{U}_{i+1}(x_k) \in \mathfrak{U}(x_k)$. The proof is complete. □

Corollary 11.1. *Let the mixed iterative ADP algorithm be implemented according to* (11.13)–(11.21). *If the iterative control law sequence* $\mathcal{U}_0(x_k)$ *satisfies* $\mathcal{U}_0(x_k) \in \mathfrak{U}(x_k)$, *where* $\mathfrak{U}(x_k)$ *is defined in* (11.12), *then we have* (11.25) *holds for any* $i = 0, 1, \ldots$.

In Theorems 11.1–11.2, the properties of the iterative control law sequence have been derived, i.e., $\mathcal{U}_i(x_k) \in \mathfrak{U}(x_k)$, $i = 0, 1, \ldots$. In the following, the convergence of the iterative value function will

be analyzed in two steps. First, the monotonicity of the iterative value function is discussed. The iterative value function is proven to be monotonically non-decreasing as the iteration index i increases. Second, the optimality of the converged iterative value function will be presented.

Theorem 11.3. *Let the mixed iterative ADP algorithm be implemented according to* (11.13)–(11.21). *For* $i = 0, 1, \ldots,$ *the iterative value function* $V_i(x_k)$ *is monotonically non-increasing, which satisfies*

$$V_{i+1}(x_k) \leq V_i(x_k). \tag{11.31}$$

Proof. For $\ell = 0, 1, \ldots,$ define $\mathcal{P}_{\ell+1}(x_k)$ as in (11.26). Then, $\mathcal{P}_{\ell+1}(x_k)$ can be written as (11.29). Letting $\mathcal{P}_\infty(x_k) = \lim_{\ell\to\infty} \mathcal{P}_\ell(x_k)$, we have

$$\begin{aligned}\mathcal{P}_\infty(x_k) = &\lim_{\ell\to\infty}\left(\sum_{\tau=0}^{\ell}\tilde{\gamma}^\tau \Pi(x_{k+\lambda\tau}, \mathcal{U}_{i+1}(x_{k+\lambda\tau}))\right)\\ &+ \lim_{\ell\to\infty}\left(\tilde{\gamma}^{\ell+1} V_i(x_{k+\lambda(\ell+1)})\right).\end{aligned} \tag{11.32}$$

According to Corollary 11.1, we know that $\mathcal{U}_{i+1}(x_k) \in \mathfrak{U}(x_k)$. Thus, we know that

$$\sum_{\tau=0}^{\infty}\tilde{\gamma}^\tau \Pi(x_{k+\lambda\tau}, \mathcal{U}_{i+1}(x_{k+\lambda\tau})) < \infty. \tag{11.33}$$

Letting

$$V_{i+1}(x_k) = \sum_{\tau=0}^{\infty}\tilde{\gamma}^\tau \Pi(x_{k+\lambda\tau}, \mathcal{U}_{i+1}(x_{k+\lambda\tau})), \tag{11.34}$$

we can derive that the iterative value function $V_{i+1}(x_k)$ satisfies (11.16) for any $i = 0, 1, \ldots$. According to (11.32) and (11.34), we can obtain

$$V_{i+1}(x_k) \leq \mathcal{P}_\infty(x_k). \tag{11.35}$$

According to (11.28), letting $\ell \to \infty$, we can get

$$\mathcal{P}_\infty(x_k) \leq V_i(x_k). \tag{11.36}$$

According to (11.35) and (11.36), we can obtain (11.31). The proof is complete. □

Remark 11.6. From Theorems 11.2–11.3, the advantages of the developed mixed iterative ADP algorithm can be recognized comparing with the dual iterative Q-learning algorithm [2]. First, from Theorem 11.2 and Corollary 11.1, we know that the performance index function cannot increase to infinity under $\mathcal{U}_i(x_k)$, $\forall\, i = 0, 1, \ldots$. In Ref. [2], however, the properties of the iterative control laws were not considered. Second, the monotonicity of the iterative value function by the mixed iterative ADP algorithm is shown in Theorem 11.3, which means that the performance of the microgrid system in the present iteration is optimized than the one in previous iteration. However, in Ref. [2] the monotonicity of the iterative Q function cannot be guaranteed, which means the performance of the microgrid system cannot be justified before the Q-learning algorithm implemented for infinite iterations. Hence, the developed mixed iterative ADP algorithm is superior than the dual iterative Q-learning algorithm in Ref. [2].

In Theorem 11.3, we have proven that the iterative value function is monotonically non-increasing as the iteration index i increases. As $V_i(x_k)$ is a lower bounded sequence, the limit of the iterative value function $V_i(x_k)$ exists as $i \to \infty$. However, in Theorem 11.3, it cannot guarantee the iterative value function converge to the optimum. Hence, in the following theorem, the optimality of the mixed iterative ADP algorithm will be developed.

Theorem 11.4. *Let the mixed iterative ADP algorithm be implemented according to* (11.13)–(11.21)*. Then, the iterative value function $V_i(x_k)$ converges to the optimum as $i \to \infty$, i.e.,*

$$\lim_{i\to\infty} V_i(x_k) = \mathcal{J}^*(x_k), \tag{11.37}$$

which satisfies the Bellman equation (11.9)*.*

Proof. Define the performance index function $V_\infty(x_k)$ as the limit of the iterative function $V_i(x_k)$, i.e.,

$$V_\infty(x_k) = \lim_{i\to\infty} V_i(x_k). \tag{11.38}$$

For $i = 0, 1, \ldots$, define $\mathscr{P}_{i+1}(x_k)$ as

$$\mathscr{P}_{i+1}(x_k) = \Pi(x_k, \mathcal{U}_{i+1}(x_k)) + \tilde{\gamma} V_i(x_{k+\lambda}). \tag{11.39}$$

Then, we can derive

$$\begin{aligned}\mathscr{P}_{i+1}(x_k) &\geq \Pi(x_k, \mathcal{U}_{i+1}(x_k)) + \tilde{\gamma} V_{i+1}(x_{k+\lambda}) \\ &= V_{i+1}(x_k). \end{aligned} \tag{11.40}$$

According to the definition of $\mathcal{U}_{i+1}(x_k)$ in (11.17), we know that

$$\mathscr{P}_{i+1}(x_k) = \min_{\mathscr{U}_k} \left\{ \Pi(x_k, \mathscr{U}_k) + \tilde{\gamma} V_i(x_{k+\lambda}) \right\}. \tag{11.41}$$

Let $i \to \infty$. We can obtain

$$\begin{aligned} V_\infty(x_k) &= \lim_{i \to \infty} V_{i+1}(x_k) \leq V_{i+1}(x_k) \leq \mathscr{P}_{i+1}(x_k) \\ &= \min_{u_k} \left\{ \Pi(x_k, \mathscr{U}_k) + \tilde{\gamma} V_i(x_{k+1}) \right\}. \end{aligned} \tag{11.42}$$

Thus, we can obtain

$$V_\infty(x_k) \leq \min_{u_k} \left\{ \Pi(x_k, \mathscr{U}_k) + \tilde{\gamma} V_\infty(x_{k+1}) \right\}. \tag{11.43}$$

Let $\varepsilon > 0$ be an arbitrary positive number. Since $V_i(x_k)$ is non-increasing for $i \geq 1$ and $\lim_{i \to \infty} V_i(x_k) = V_\infty(x_k)$, there exists a positive integer p such that

$$V_p(x_k) - \varepsilon \leq V_\infty(x_k) \leq V_p(x_k). \tag{11.44}$$

Hence, we can get

$$\begin{aligned} V_\infty(x_k) &\geq \Pi(x_k, \mathcal{U}_p(x_k)) + \tilde{\gamma} V_p(x_{k+\lambda}) - \varepsilon \\ &\geq \Pi(x_k, \mathcal{U}_p(x_k)) + \tilde{\gamma} V_\infty(x_{k+\lambda}) - \varepsilon \\ &\geq \min_{\mathscr{U}_k} \left\{ \Pi(x_k, \mathscr{U}_k) + \tilde{\gamma} V_\infty(x_{k+\lambda}) \right\} - \varepsilon. \end{aligned} \tag{11.45}$$

Since ε is arbitrary, we have

$$V_\infty(x_k) \geq \min_{\mathscr{U}_k} \left\{ \Pi(x_k, \mathscr{U}_k) + \tilde{\gamma} V_\infty(x_{k+\lambda}) \right\}. \tag{11.46}$$

Combining (11.43) and (11.46), we can obtain

$$V_\infty(x_k) = \min_{\mathscr{U}_k} \left\{ \Pi(x_k, \mathscr{U}_k) + \tilde{\gamma} V_\infty(x_{k+\lambda}) \right\}. \tag{11.47}$$

According to the definition of the Bellman equation (11.9), we know that $V_\infty(x_k) = \mathcal{J}^*(x_k)$ and we can draw the conclusion. □

Now, we summarize the mixed iterative ADP algorithm in Algorithm 11.1.

Algorithm 11.1 The value iteration ADP algorithm

Initialization:

Choose randomly an array of initial states x_0;
Choose a computation precision ε;
Give a control law sequence $\mathcal{U}_0(x_k) \in \mathfrak{U}(x_k)$;

Iteration:

1: Let the iteration index $i = 0$ and let $V_{i-1}(x_k) = 0$;
P-**Iteration**
2: For $i = 0, 1, \ldots,$ construct iterative value function by the control law sequence $\mathcal{U}_i(x_k)$, which expressed as

$$V_i(x_k) = \Pi\left(x_k, \mathcal{U}_i(x_k)\right) + \tilde{\gamma} V_i(x_{k+\lambda});$$

3: If $\forall\, x_k$, $|V_i(x_k) - V_{i-1}(x_k)| \leq \varepsilon$, then goto Step 9; otherwise, goto Step 5;
4: Construct the iterative control law

$$\mathcal{U}_{i+1}(x_k) = \{u_{i+1}^{\lambda-1}(x_k), u_{i+1}^{\lambda-2}(x_k), \ldots, u_{i+1}^{0}(x_k)\}$$

and then let $i = i + 1$ and goto Step 2;
V-**Iteration**
5: Let $j = 0$ and let

$$\mathcal{V}_{i+1}^{j}(x_k) = V_i(x_k);$$

6: For $j = 0, 1, \ldots, \lambda - 1$, compute the

$$u_{i+1}^{j}(x_k) = \arg\min_{u_k} \left\{U(x_k, u_k, j) + \gamma \mathcal{V}_{i+1}^{j}(x_{k+1})\right\};$$

7: Update the iterative value function

$$\mathcal{V}_{i+1}^{j+1}(x_k) = \min_{u_k} \left\{U(x_k, u_k, j) + \gamma \mathcal{V}_{i+1}^{j}(x_{k+1})\right\};$$

8: If $j < \lambda - 1$, then let $j = j + 1$ and goto Step 6; otherwise goto Step 4;
9: **return** $V_i(x_k)$ and $\mathcal{U}_i(x_k)$.

11.4 Simulation Example

In this section, the performance of the mixed iterative ADP algorithm for microgrid system will be examined by numerical experiments. Choose the residential load demand and the real-time electricity rate in Ref. [1], where the home load demand and the electricity rate in a week (168 hours) are shown in Figs. 11.1(a) and (c), respectively. The average trajectories of the residential load demand and the electricity rate are shown in Figs. 11.1(b) and (d) with the period $\lambda = 24$. If there is no optimization, the original electricity cost in a week is 4124.13 cents.

The upper and lower storage limits of the battery are defined as $E_b^{\min} = 20$ kWh and $E_b^{\max} = 80$ kWh, respectively. The rated

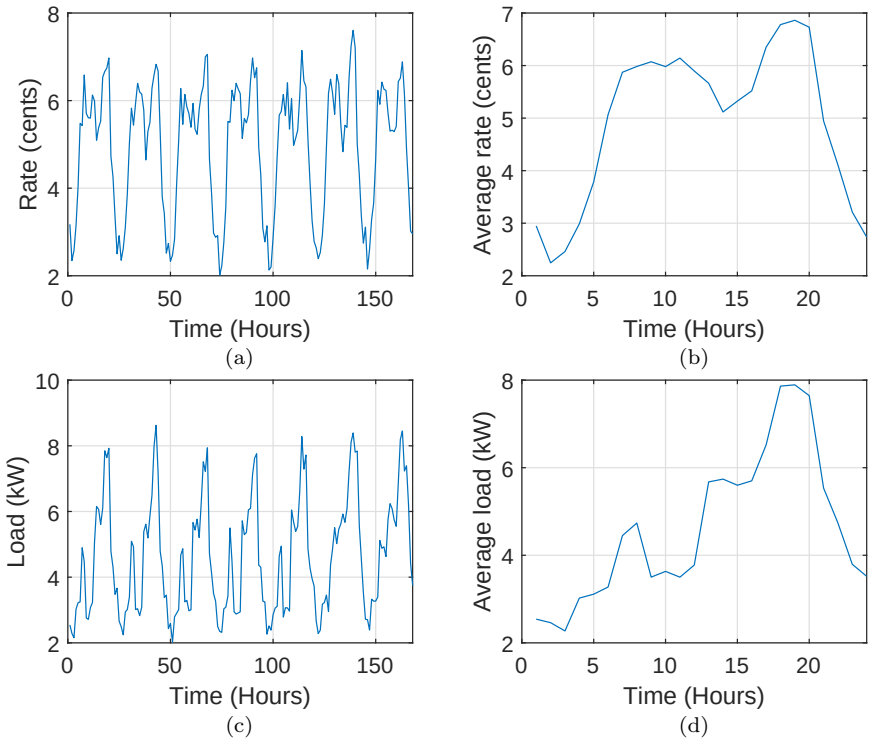

Fig. 11.1. Residential load demand and electricity rate. (a) Residential load demand in a week. (b) Average residential load demand in a day. (c) Real-time electricity rate in a week. (d) Average electricity rate in a day.

power output of the battery and the rated charging/discharging rate is 16 kW. The initial energy level of the battery is 30 kWh. The performance index function is chosen as in (11.5), with the parameters $m_1 = 1$, $m_2 = 0.2$, $r = 0.1$, and $\gamma = 0.97$. Let the initial state be $x_0 = [8, 30]^{\mathrm{T}}$. To implement the mixed iterative ADP algorithm, it is necessary to choose an initial control law $\mathcal{U}_0(x_k)$ which satisfies $\mathcal{U}_0(x_k) \in \mathfrak{U}(x_k)$. It can be chosen by the following experiments: The battery is charging if the load is low and the electricity rate is inexpensive and discharging if the load is high and the electricity rate is expensive. Otherwise, the battery control is arbitrary. Based on the above experiments, the initial control law is chosen as the one shown in Fig. 11.2. The electricity costs under $\mathcal{U}_0(x_k)$ in a week becomes 4006.39 cents.

Based on the initial control law sequence $\mathcal{U}_0(x_k)$, we implement the mixed iterative ADP algorithm (11.13)–(11.21). The iterative

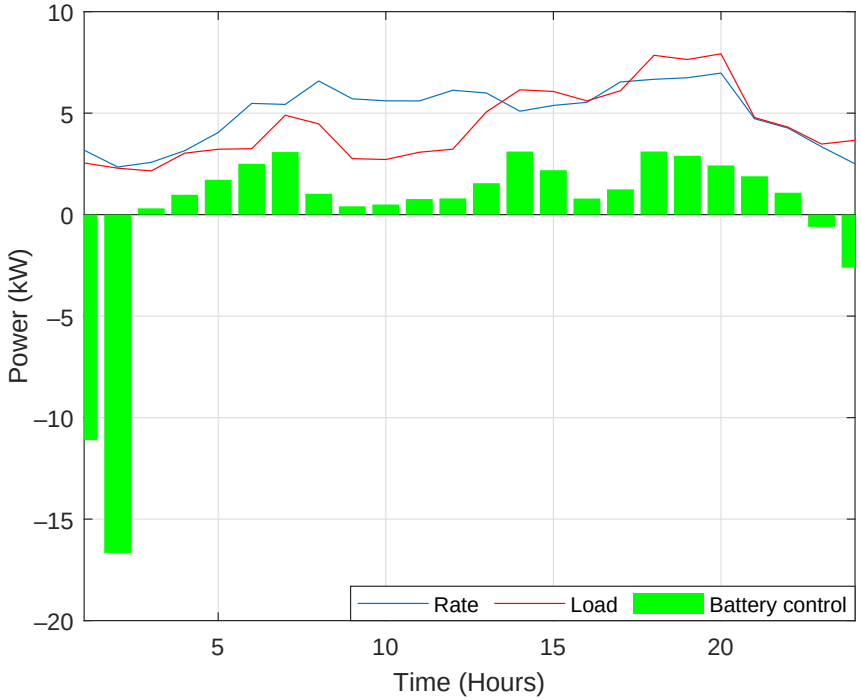

Fig. 11.2. The initial control law sequence $\mathcal{U}_0(x_k)$.

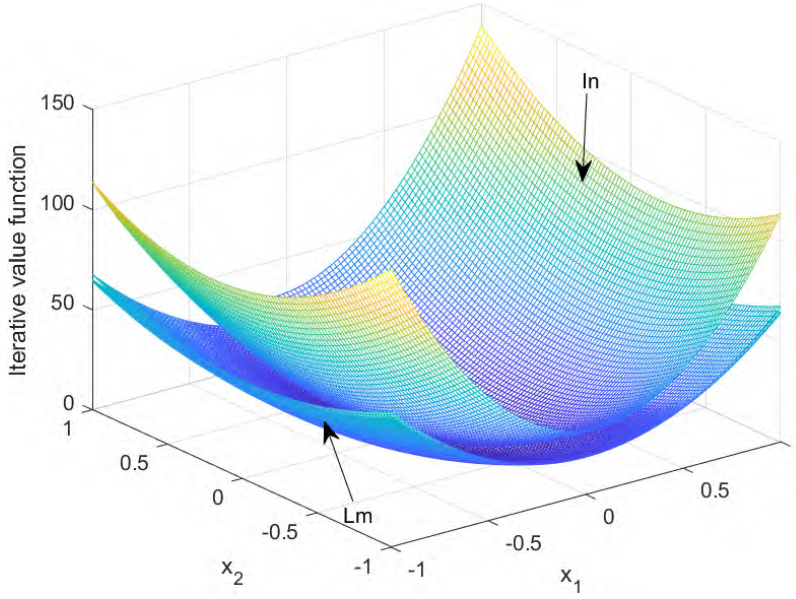

Fig. 11.3. The plots of the iterative value function.

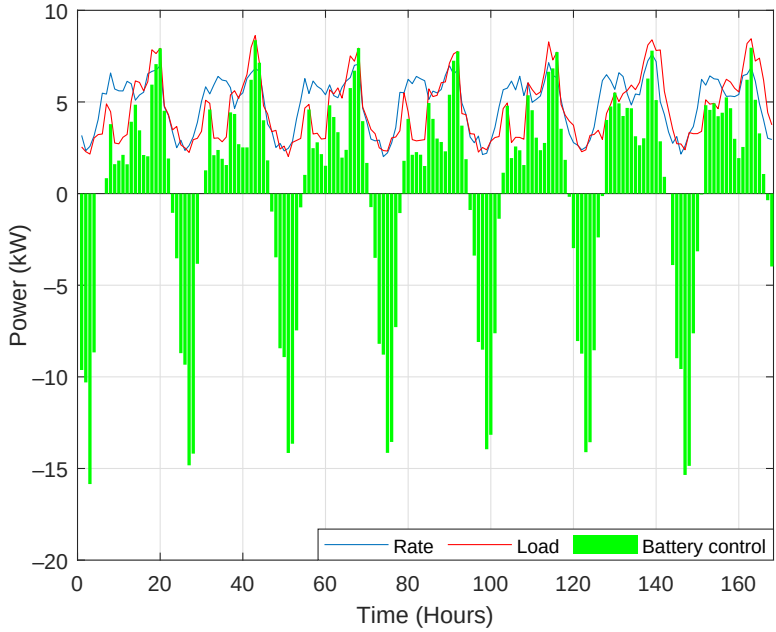

Fig. 11.4. Optimal control of the battery in a week.

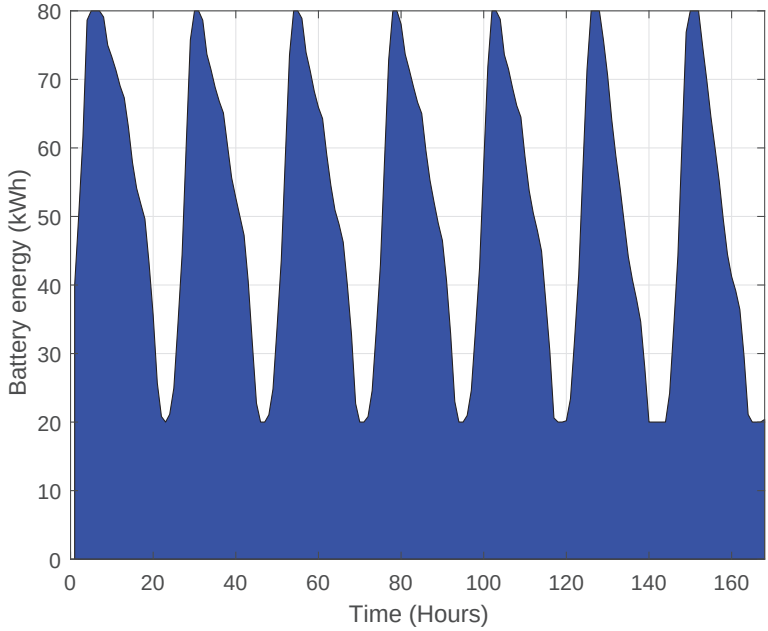

Fig. 11.5. Optimal battery energy in a week.

value function is displayed in Fig. 11.3. It takes 6 P-iterations to guarantee that the iterative value function converges to the computation precision $\xi = 10^{-2}$. From Fig. 11.3, we can see that the iterative value function is monotonically non-increasing to the optimum, which justifies the correctness of Theorem 11.3 in this chapter. The optimal battery control in a week is shown in Fig. 11.4. The optimal battery energy is shown in Fig. 11.5. The optimal electricity costs by mixed iterative ADP algorithm in a week reduces to 2944.72 cents.

11.5 Conclusion

In this chapter, two iterations in the mixed iterative ADP algorithm are employed, which are P-iteration and V-iteration, respectively. Based on value iteration, the V-iteration is implemented to obtain the iterative control law sequence, which aims to minimizes the total cost in a period. The iterative value function is improved by the P-iteration which guarantees that the iterative value function

is monotonically non-increasing and converges to the optimum. We emphasize that any of the iterative control law sequence can make the performance index function finite, which guarantees the finite electricity cost in each iteration.

References

[1] Huang, T. and Liu, D. (2013). A self-learning scheme for residential energy system control and management. *Neural Computing & Applications* 22(2), 259–269.

[2] Wei, Q., Liu, D. and Shi, G. (2015). A novel dual iterative Q-learning method for optimal battery management in smart residential environments. *IEEE Transactions on Industrial Electronics* 62(4), 2509–2518.

[3] Lewis, F.L., Vrabie, D. and Vamvoudakis, K.G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine* 32(6), 76–105.

Chapter 12

Error-Tolerant Iterative Adaptive Dynamic Programming for Optimal Renewable Home Energy Scheduling and Battery Management

12.1 Introduction

In this chapter, a novel error-tolerant iterative ADP algorithm is developed to solve optimal battery control and management problems in smart home environments with renewable energy. The algorithm can be initialized by an arbitrary positive semi-definite function. Based on the quasi-periodic data of the electricity rate, the home load demand, and the renewable energy, the error-tolerant iterative ADP algorithm can be implemented. Considering the differences of the data in different periods, the developed error-tolerant iterative ADP algorithm can adaptively regulate the discount factor in each iteration to make the iterative value function converge. A new analysis method is developed to guarantee the iterative value function to converge to a finite neighborhood of the optimal performance index function, in spite of the differences of the electricity rate, the home load demand, and the renewable energy in different periods. Numerical results are given to illustrate the performance of the developed algorithm.

12.2 Problem Formulation

The smart home energy system with renewable energy is described in Ref. [1]. The system is composed of the power grid, the renewable energy system, the home load demand, and the energy storage (battery) system, which is expressed in Fig. 12.1. The optimal battery control and management problem is treated as a discrete time problem with the time step of 1 hour and it is assumed that the electricity rate, the home load demand, and the renewable energy vary hourly. Regarding solar renewable energy, the estimated total solar radiation varies from a certain period to another and also depends on the position of the sun in the sky. A typical solar panel characteristic is chosen as in Refs. [1, 2]. The renewable energy in a week which is used in this chapter is shown in Fig. 12.2.

As the renewable energy is cost free, the energy of the renewable resource, first and foremost, is desired to meet the home load demand and the rest of the energy is desired to charge the battery. Letting $P_{L,k}$ and $P_{R,k}$ be the given power of the home load demand (kW) and the given power of the renewable resource (kW), respectively, we can define the new home load demand and renewable energy as

$$\mathscr{P}_{L,k} = \begin{cases} P_{L,k} - P_{R,k}, & P_{L,k} - P_{R,k} \geq 0, \\ 0, & P_{L,k} - P_{R,k} < 0, \end{cases} \tag{12.1}$$

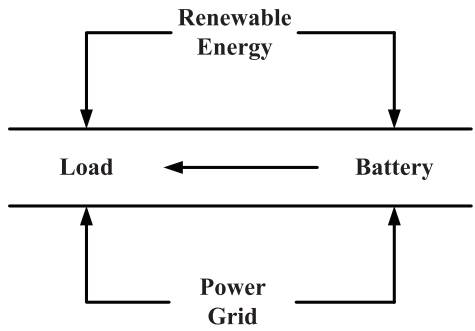

Fig. 12.1. Smart home energy system with renewable energy.

and

$$\mathscr{P}_{R,k} = \begin{cases} P_{R,k} - P_{L,k}, & P_{R,k} - P_{L,k} \geq 0, \\ 0, & P_{R,k} - P_{L,k} < 0. \end{cases} \tag{12.2}$$

Then, the home load balance can be rewritten as

$$\mathscr{P}_{L,k} = P_{G,k} + (P_{BL,k} - P_{GB,k}), \tag{12.3}$$

where $P_{G,k}$ is the power supply of the grid (kW), $P_{BL,k}$ is the power from the battery to the load (kW), and $P_{GB,k}$ is the power from the power grid to the battery (kW).

The battery model used in this work is based on Ref. [3], where the battery efficiency is considered to extend the battery's lifetime as far as possible. Let $E_{B,k}$ be the battery energy (kWh) at time k and let $\eta(\cdot)$ be the charging/discharging efficiency of the battery. Then,

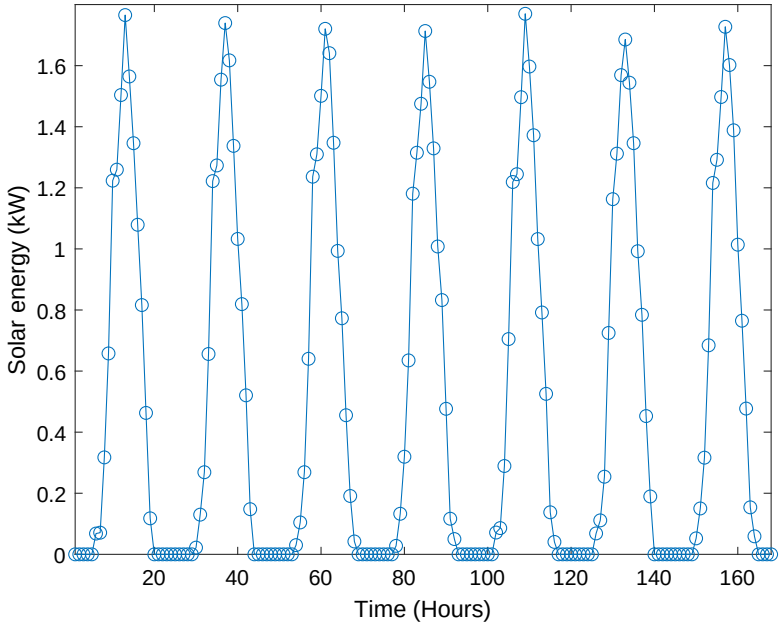

Fig. 12.2. Solar renewable energy.

the battery model can be expressed as

$$E_{B,k+1} = E_{B,k} - (P_{BL,k} - (P_{RB,k} + P_{GB,k})) \times \eta(P_{BL,k} - (P_{RB,k} + P_{GB,k})), \tag{12.4}$$

where $P_{RB,k}$ is the power from the renewable resource to the battery (kW). Let the efficiency of battery charging/discharging be derived as $\eta(P_{BL,k} - (P_{RB,k} + P_{GB,k})) = 0.898 - 0.173|P_{BL,k} - (P_{RB,k} + P_{GB,k})|/P_{\text{rate}}$, where $P_{\text{rate}} > 0$ is the rated power output of the battery (kW). The storage limit of the battery is considered, i.e., $E_b^{\min} \leq E_{B,k} \leq E_b^{\max}$, where $E_b^{\min}$ and $E_b^{\max}$ are the minimum and maximum storage energy of the battery, respectively.

Remark 12.1. We declare that the establishment of the PV panel and the battery system are not free. In this paper, the cost of the establishment of the PV panel and the battery system are not considered. After the smart home is established, the main contribution of this paper is to design an effective iterative ADP algorithm to find the optimal control law for the smart homes, which minimizes the total electricity cost and simultaneously extend the lifetime of the battery.

12.3 Error-Tolerant Iterative ADP Algorithm

In this section, a new error-tolerant iterative ADP algorithm is developed to obtain the optimal control law for home energy system with renewable energy, where a novel convergence analysis method will be developed in this section.

12.3.1 *Optimization objectives*

For convenience of analysis, results of this chapter are based on the following assumptions.

Assumption 12.1. The power flow from the battery to the grid is not permitted to guarantee the power quality of the grid.

Assumption 12.2. The electricity rate C_k (cents/kWh), the home load demand $P_{L,k}$, and the renewable energy $P_{R,k}$ are quasi-periodic functions with the period $\lambda = 24$ hours.

According to Assumption 12.1, we can get $P_{G,k} \geq 0$. The total cost expected to be minimized can be written as

$$\sum_{k=0}^{\infty} \gamma^k \left(m_1 (C_k P_{G,k})^2 + m_2 \left(E_{B,k} - \frac{1}{2}(E_b^{\min} + E_b^{\max}) \right)^2 + m_3 \big(P_{BL,k} - (P_{RB,k} + P_{GB,k}) \big)^2 \right), \tag{12.5}$$

where the discount factor $0 < \gamma < 1$ and m_1, m_2, and m_3 are positive constants. The first term of the performance index function aims to minimize the total cost from the grid. The second term avoids fully charging/discharging of the battery and the third term is to prevent large charging/discharging power of the battery. We introduce delays in $P_{BL,k}$ and $P_{GB,k}$. Define the load balance as $\mathscr{P}_{L,k-1} = P_{G,k} + P_{BL,k-1} - P_{GB,k-1}$. Let $x_{1,k} = P_{G,k}$ and $x_{2,k} = E_{B,k} - E_b^o$ be the two system states. From the model of the battery, we know that the power flows $P_{GB,k}$, and $P_{BL,k}$ are nonnegative values, i.e., $P_{GB,k} \geq 0$ and $P_{BL,k} \geq 0$. As the battery is not permitted to charge and discharge simultaneously, we know $P_{BL,k} = 0$, if $P_{GB,k} \geq 0$ and $P_{GB,k} = 0$, if $P_{BL,k} \geq 0$. Then, we can define the control input as $u_k = P_{BL,k} - P_{GB,k}$. Letting $x_k = [x_{1,k}, x_{2,k}]^{\mathsf{T}}$, the equation of the home energy system can be written as

$$\begin{aligned} x_{k+1} &= F(x_k, u_k, k) \\ &= \begin{pmatrix} \mathscr{P}_{L,k} - u_k \\ x_{2,k} - (u_k - \mathscr{P}_{R,k})\eta(u_k - \mathscr{P}_{R,k}) \end{pmatrix}. \end{aligned} \tag{12.6}$$

Let $\underline{u}_k = (u_k, u_{k+1}, \ldots)$ denote the control sequence from k to ∞. Let $M_k = \begin{bmatrix} m_1 C_k^2 & 0 \\ 0 & m_2 \end{bmatrix}$. Let x_0 be the initial state. According to the definitions of the state and control, the performance index function, which is expected to be minimized, can be written as

$$J(x_0, \underline{u}_0, 0) = \sum_{k=0}^{\infty} \gamma^k U(x_k, u_k, k), \tag{12.7}$$

where the utility function is expressed as $U(x_k, u_k, k) = x_k^{\mathsf{T}} M_k x_k + m_3 u_k^2$. Define the optimal performance index function can be

defined as

$$J^*(x_k,k)=\inf_{\underline{u}_k}\{J(x_k,\underline{u}_k,k)\}. \tag{12.8}$$

According to Bellman's principle of optimality, we can obtain the following discrete-time Bellman equation $J^*(x_k,k)=\inf_{u_k}\left\{U(x_k,u_k,k)+\gamma J^*(x_{k+1},k+1)\right\}$.

12.3.2 *Algorithm derivations*

In this section, detailed derivations of the error-tolerant iterative ADP algorithm are presented. In the iterative ADP algorithm, the value function and control law sequence are both updated at every iteration.

Let $i=0,1,\ldots$ be the iteration index. Let $\mathcal{C}_k^i=\{C_k^i,C_{k+1}^i,\ldots,C_{k+\lambda-1}^i\}$ be the collected rate data in the ith iteration. Let $\mathcal{P}_{L,k}^i=\{P_{L,k}^i,P_{L,k+1}^i,\ldots,P_{L,k+\lambda-1}^i\}$ and $\mathcal{P}_{R,k}^i=\{P_{R,k}^i,P_{R,k+1}^i,\ldots,P_{R,k+\lambda-1}^i\}$ be the collected home load demand and renewable energy data in the ith iteration, respectively. Then, we let

$$\mathscr{P}_{L,k}^i=\begin{cases}P_{L,k}^i-P_{R,k}^i, & P_{L,k}^i-P_{R,k}^i\geq 0,\\ 0, & P_{L,k}^i-P_{R,k}^i<0,\end{cases} \tag{12.9}$$

and

$$\mathscr{P}_{R,k}^i=\begin{cases}P_{R,k}^i-P_{L,k}^i, & P_{R,k}^i-P_{L,k}^i\geq 0,\\ 0, & P_{R,k}^i-P_{L,k}^i<0.\end{cases} \tag{12.10}$$

For $i=0,1,\ldots,$ and $j=0,1,\ldots,\lambda-1$, we let $U_i(x_k,u_k,\lambda-1-j)=x_k^{\mathsf{T}}M_{\lambda-1-j}^i x_k+m_3u_k^2$ and let

$$\begin{aligned}\mathscr{U}_i(x_k,u_k,j)&=U_i(x_k,u_k,\lambda-1-j)\\&=x_k^{\mathsf{T}}M_{\lambda-1-j}^i x_k+m_3u_k^2,\end{aligned} \tag{12.11}$$

where $M_{\lambda-1-j}^i=\begin{bmatrix}m_1(C_{\lambda-1-j}^i)^2 & 0\\ 0 & m_2\end{bmatrix}$. For $i=0,1,\ldots,$ and $j=0,1,\ldots,\lambda-1$, we let

$$\mathscr{F}_i(x_k,u_k,j)=F_i(x_k,u_k,\lambda-1-j), \tag{12.12}$$

where the function $F_i(x_k, u_k, \lambda - 1 - j)$ is defined as

$$F_i(x_k, u_k, \lambda - 1 - j) = \begin{pmatrix} \mathscr{P}^i_{L,\lambda-1-j} - u_k \\ x_{2,k} - (u_k - \mathscr{P}^i_{R,\lambda-1-j})\eta(u_k - \mathscr{P}^i_{R,\lambda-1-j}) \end{pmatrix}. \tag{12.13}$$

Define $\mathcal{U}_k = (u_k, u_{k+1}, \ldots, u_{k+\lambda-1})$ as the control sequence in a period. For any $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$, according to (12.12) and (12.13), the state under the control $\mathcal{U}_k$ can be derived as

$$\begin{cases} x_{k+1} = F_i(x_k, u_k, \lambda - 1 - j), \\ x_{k+2} = F_i(x_{k+1}, u_{k+1}, \lambda - j), \\ \vdots \\ x_{k+\lambda} = F_i(x_{k+\lambda-1}, u_{k+\lambda-1}, 2\lambda - j). \end{cases} \tag{12.14}$$

According to (12.14), there exists a function $\mathcal{F}_i(\cdot)$, such that the state $x_{k+\lambda}$ can be derived as

$$x_{k+\lambda} = \mathcal{F}_i(x_k, \mathcal{U}_k, j). \tag{12.15}$$

Now, we define a new iterative discount factor γ_i, $i = 0, 1, \ldots$, where we let γ_i is a positive constant which satisfies $0 < \gamma_i < 1$. Let the initial iterative value function $\Psi(x_k)$ be an arbitrary positive semi-definite function. For $i = 0, 1, \ldots$, let $j = 0, 1, \ldots, \lambda - 1$ be the local iteration index. For $i = 0$ and $j = 0$, the initial value function is defined as $V_0^0(x_k) = \Psi(x_k)$. Then, for $j = 0, 1, \ldots, \lambda - 1$, the iterative control law is obtained by

$$\begin{aligned} u_0^j(x_k) &= \arg\min_{u_k}\{\mathscr{U}_0(x_k, u_k, j) + \gamma_0 V_0^j(x_{k+1})\} \\ &= \arg\min_{u_k}\{\mathscr{U}_0(x_k, u_k, j) + \gamma_0 V_i^j(\mathscr{F}_0(x_k, u_k, j))\}, \end{aligned} \tag{12.16}$$

where $x_{k+1} = \mathscr{F}_0(x_k, u_k, j)$ is defined in (12.12) and the utility function $\mathscr{U}_0(x_k, u_k, j)$ is defined in (12.11). According to the iterative control law $u_0^j(x_k)$, we can update the iterative value function by

$$V_0^{j+1}(x_k) = \mathscr{U}_0(x_k, u_0^j(x_k), j) + \gamma_0 V_0^j(\mathscr{F}_0(x_k, u_0^j(x_k), j)). \tag{12.17}$$

For $i = 1, 2, \ldots$, we let $V_i^0(x_k) = V_{i-1}^{\lambda}(x_k)$. Then, for $i = 1, 2, \ldots$, $j = 0, 1, \ldots, \lambda - 1$, the iterative control law is obtained by

$$\begin{aligned} u_i^j(x_k) &= \arg\min_{u_k}\{\mathscr{U}_i(x_k, u_k, j) + \gamma_i V_i^j(x_{k+1})\}, \\ &= \arg\min_{u_k}\{\mathscr{U}_i(x_k, u_k, j) + \gamma_i V_i^j(\mathscr{F}_i(x_k, u_k, j))\}, \end{aligned} \tag{12.18}$$

where $\mathscr{F}_i(x_k, u_k, j)$ is defined in (12.12) and the utility function $\mathscr{U}_i(x_k, u_k, j)$ is defined in (12.11). Then, the iterative value function is updated by

$$V_i^{j+1}(x_k) = \mathscr{U}_i(x_k, u_i^j(x_k), j) + \gamma_i V_i^j(\mathscr{F}_i(x_k, u_i^j(x_k), j)), \tag{12.19}$$

where $V_i^j(x_k) = V_{i-1}^{\lambda}(x_k)$ for $j = 0$. Then, for any $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$, we can construct the periodic iterative control law sequence by

$$\begin{aligned} \mathcal{U}_i^{j+1}(x_k) = \Big\{ & u_i^j(x_k), u_i^{j-1}(x_k), \ldots, u_i^0(x_k), u_i^{\lambda-1}(x_k), \\ & u_i^{\lambda-2}(x_k), \ldots, u_i^{j+1}(x_k) \Big\}. \end{aligned} \tag{12.20}$$

From (12.16)–(12.20), we can see that in each iteration of the algorithm, the data of the electricity rate, the home load demand, and the renewable energy are different. In Ref. [4], the convergence of the iterative value function was presented by the accurate iterative ADP algorithms without considering the renewable energy. With the renewable energy and errors, convergence analysis for the previous iterative ADP algorithms [4] is invalid. In this chapter, a new iterative ADP algorithm (12.16)–(12.20) is established to obtain the optimal renewable home energy control law with errors. In the next subsection, new methods will be established to analyze the properties of the developed iterative ADP algorithm.

12.3.3 *Properties of the error-tolerant iterative ADP algorithm*

To analyze the properties of the error-tolerant iterative ADP algorithm (12.16)–(12.20), we first discuss a special case of the algorithm, where the electricity rate, the home load demand, and the

renewable energy are all accurate periodic functions. Let $\bar{C}_k$, $\bar{P}_{L,k}$, and $\bar{P}_{R,k}$ be the expected electricity rate, the expected home load demand, and the expected renewable energy, respectively, which satisfy $\bar{C}_k = \bar{C}_{k+\lambda}$, $\bar{P}_{L,k} = \bar{P}_{L,k+\lambda}$, and $\bar{P}_{R,k} = \bar{P}_{R,k+\lambda}$. We let

$$\bar{\mathscr{P}}_{L,k} = \begin{cases} \bar{P}_{L,k} - \bar{P}_{R,k}, & \bar{P}_{L,k} - \bar{P}_{R,k} \geq 0, \\ 0, & \bar{P}_{L,k} - \bar{P}_{R,k} < 0, \end{cases} \tag{12.21}$$

and

$$\bar{\mathscr{P}}_{R,k} = \begin{cases} \bar{P}_{R,k} - \bar{P}_{L,k}, & \bar{P}_{R,k} - \bar{P}_{L,k} \geq 0, \\ 0, & \bar{P}_{R,k} - \bar{P}_{L,k} < 0. \end{cases} \tag{12.22}$$

According to (12.21)–(12.22), for $j = 0, 1, \ldots, \lambda - 1$, we can define the expected system function $\bar{F}(\cdot)$ as

$$\begin{aligned} x_{k+1} &= \bar{F}(x_k, u_k, \lambda - 1 - j) \\ &= \begin{pmatrix} \bar{\mathscr{P}}_{L,\lambda-1-j} - u_k \\ x_{2,k} - (u_k - \bar{\mathscr{P}}_{R,\lambda-1-j})\eta(u_k - \bar{\mathscr{P}}_{R,\lambda-1-j}) \end{pmatrix}. \end{aligned} \tag{12.23}$$

Let $\bar{\mathscr{F}}(x_k, u_k, j) = \bar{F}(x_k, u_k, \lambda - 1 - j)$.

For $i = 0, 1, \ldots,$ and $j = 0, 1, \ldots, \lambda - 1$, we can define the expected utility function as $\bar{\mathscr{U}}(x_k, u_k, j) = x_k^{\mathrm{T}} \begin{bmatrix} m_1(\bar{C}_{\lambda-1-j})^2 & 0 \\ 0 & m_2 \end{bmatrix} x_k + m_3 u_k^2$. For $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$, we let

$$\begin{aligned} \bar{\Upsilon}(x_k, \mathcal{U}_k, j) = &\sum_{t=0}^{j} \gamma^t \bar{\mathscr{U}}(x_{k+t}, u_{k+t}, j - t) \\ &+ \sum_{t=0}^{\lambda-j-1} \gamma^{t+j} \bar{\mathscr{U}}(x_{k+j+t}, u_{k+j+t}, \lambda - t), \end{aligned} \tag{12.24}$$

where we let $\mathcal{U}_k = \underline{u}_k^{k+\lambda-1}$. According to (12.23), there exists a function $\bar{\mathcal{F}}(\cdot)$, such that the state $x_{k+\lambda}$ can be derived as

$$x_{k+\lambda} = \bar{\mathcal{F}}(x_k, \mathcal{U}_k, j). \tag{12.25}$$

Then, letting $\tilde{\gamma} = \gamma^{\lambda}$, the expected iterative value function can be written as

$$\bar{V}_{i+1}^{j+1}(x_k) = \min_{\mathcal{U}_k}\{\bar{\Upsilon}(x_k, \mathcal{U}_k, j) + \tilde{\gamma}\bar{V}_i^{j+1}(x_{k+\lambda})\}. \tag{12.26}$$

According to Ref. [4], it can be easily derived that the expected iterative value function $\bar{V}_i^j(x_k)$ converges to the optimal performance index function as $i \to \infty$ (see the proof of Theorem 2 in Ref. [4]). However, in this chapter, the electricity rate C_k, the home load demand $P_{L,k}$, and the renewable energy $P_{R,k}$, generally do not equal their expected ones. Then, for $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$, the iterative value function $V_i^j(x_k)$ cannot equal the expected one, i.e., $V_i^j(x_k) \neq \bar{V}_i^j(x_k)$. Hence, the convergence analysis in Ref. [4] is not applicable for $V_i^j(x_k)$ and new analysis will be developed.

Now, for $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$, we define a new iterative utility function as

$$\begin{aligned}\Upsilon_i(x_k, \mathcal{U}_k, j) = &\sum_{t=0}^{j} \gamma_i^t \mathscr{U}_i(x_{k+t}, u_{k+t}, j-t) \\ &+ \sum_{t=0}^{\lambda-j-1} \gamma_i^{t+j} \mathscr{U}_i(x_{k+j+t}, u_{k+j+t}, \lambda - t).\end{aligned} \tag{12.27}$$

According to (12.16)–(12.19), for $i = 0, 1, \ldots,$ and $j = 0, 1, \ldots, \lambda - 1$, the iterative value function is expressed as

$$\begin{aligned}V_i^{j+1}(x_k) &= \min_{u_k}\left\{\mathscr{U}_i(x_k, u_k, j) + \gamma_i V_i^{j+1}(x_{k+1})\right\} \\ &= \min_{u_k}\left\{\mathscr{U}_i(x_k, u_k, j) + \gamma_i \min_{u_{k+1}}\left\{\mathscr{U}_i(x_{k+1}, u_{k+1}, j-1)\right.\right. \\ &\quad \left.\left. + \gamma_i V_i^{j+1}(x_{k+1})\right\}\right\} \\ &= \min_{\mathcal{U}_k}\left\{\Upsilon_i(x_k, \mathcal{U}_k, j) + \tilde{\gamma}_i V_{i-1}^{j+1}(x_{k+\lambda})\right\},\end{aligned} \tag{12.28}$$

where $\mathcal{U}_k = \underline{u}_k^{k+\lambda-1}$, $\tilde{\gamma}_i = \gamma_i^{\lambda}$, and $x_{k+\lambda} = \mathcal{F}_i(x_k, \mathcal{U}_k, j)$ is defined in (12.15).

In this chapter, it is assumed that there exist three constants, ϕ_i, σ_i, and α_i, such that $0 < \phi_i < \infty$, $1 \leq \sigma_i < \infty$, and $1 \leq \alpha_i < \infty$,

respectively, which satisfy the following inequalities

$$\tilde{\gamma}\bar{V}_i^{j+1}(\bar{\mathcal{F}}(x_k,\mathcal{U}_k,j)) \leq \phi_i\bar{\Upsilon}(x_k,\mathcal{U}_k,j), \tag{12.29}$$

$$V_i^{j+1}(\mathcal{F}_i(x_k,\mathcal{U}_k,j)) \leq \sigma_i V_i^{j+1}(\bar{\mathcal{F}}(x_k,\mathcal{U}_k,j)), \tag{12.30}$$

and

$$\Upsilon_i(x_k,\mathcal{U}_k,j) \leq \alpha_i\bar{\Upsilon}(x_k,\mathcal{U}_k,j). \tag{12.31}$$

Based on the above preparations, we can derive the following main theorems.

Theorem 12.1. *For $i = 0,1,\ldots$ and $j = 0,1,\ldots,\lambda-1$, let the iterative value function $V_i^{j+1}(x_k)$ and the iterative control law $u_i^j(x_k)$ be obtained by* (12.16)–(12.19)*. Assume that the constants ϕ_i, σ_i, and α_i, such that $0 < \phi_i < \infty$, $1 \leq \sigma_i < \infty$, and $1 \leq \alpha_i < \infty$, which satisfy* (12.29)–(12.31)*, respectively. Then the iterative value function $V_i^{j+1}(x_k)$ satisfies*

$$V_{i+1}^{j+1}(x_k) \leq \alpha_i\left(1+\sum_{\tau=1}^{i}\left(\prod_{l=1}^{\tau-1}\left(\left(\sigma_{i-l}\frac{\gamma_{i-l}}{\gamma}\alpha_{i-l-1}\right)\frac{\phi_{i-l}}{(\phi_{i-l}+1)}\right)\right)\right.$$
$$\left.\times\left(\sigma_{i-\tau}\frac{\gamma_{i-\tau}}{\gamma}\alpha_{i-\tau-1}-1\right)\right)\bar{V}_{i+1}^{j+1}(x_k), \tag{12.32}$$

where we let $\alpha_i = 1$, for $i < 0$, and $\prod_{\tau}^{i}(\cdot) = 1$ for $\tau > i$.

Proof. The theorem can be proven by mathematical induction. First, let $i = 0$. For any $j = 0,1,\ldots,\lambda-1$, according to (12.28)–(12.31), we can derive

$$V_1^{j+1}(x_k) \leq \min_{\mathcal{U}_k}\left\{\alpha_0\bar{\Upsilon}(x_k,\mathcal{U}_k,j)+\alpha_0\frac{\tilde{\gamma}_0}{\tilde{\gamma}}\sigma_0\tilde{\gamma}\bar{V}_0^{j+1}(\bar{\mathcal{F}}(x_k,\mathcal{U}_k,j))\right.$$
$$+\frac{\alpha_0}{1+\phi_0}\left(\frac{\tilde{\gamma}_0}{\tilde{\gamma}}\sigma_0-1\right)\left(\phi_0\bar{\Upsilon}(x_k,\mathcal{U}_k,j)\right.$$
$$\left.\left.-\tilde{\gamma}V_0^{j+1}(\bar{\mathcal{F}}(x_k,\mathcal{U}_k,j))\right)\right\}$$

$$
\begin{aligned}
&= \alpha_0 \left(1 + \frac{\phi_0}{\phi_0 + 1} \left(\frac{\tilde{\gamma}_0}{\tilde{\gamma}} \sigma_0 - 1\right)\right) \\
&\quad \times \min_{\mathcal{U}_k} \left\{\bar{\Upsilon}(x_k, \mathcal{U}_k, j) + \tilde{\gamma} \bar{V}_0^{j+1}(\bar{\mathcal{F}}(x_k, \mathcal{U}_k, j))\right\} \\
&= \alpha_0 \left(1 + \frac{\phi_0}{\phi_0 + 1} \left(\frac{\tilde{\gamma}_0}{\tilde{\gamma}} \sigma_0 - 1\right)\right) \bar{V}_1^{j+1}(x_k), \qquad (12.33)
\end{aligned}
$$

where $x_{k+\lambda} = \mathcal{F}_0(x_k, \mathcal{U}_k, j)$ and $\tilde{\gamma}_0 = \gamma_0^{\lambda}$. Hence, the inequality (12.32) holds for $i = 0$. Assume that the inequality (12.32) holds for $i = \ell - 1$, $\ell = 1, 2, \ldots$. Then, for $i = \ell$, we have

$$
\begin{aligned}
&V_{\ell+1}^{j+1}(x_k) \\
&\quad = \min_{\mathcal{U}_k} \left\{\Upsilon_\ell(x_k, \mathcal{U}_k, j) + \tilde{\gamma}_\ell V_\ell^{j+1}(\mathcal{F}_\ell(x_k, \mathcal{U}_k, j))\right\} \\
&\quad \leq \min_{\mathcal{U}_k} \left\{\alpha_\ell \bar{\Upsilon}(x_k, \mathcal{U}_k, j) + \alpha_\ell \frac{\tilde{\gamma}_\ell}{\tilde{\gamma}} \sigma_\ell \tilde{\gamma} V_\ell^{j+1}(\bar{\mathcal{F}}(x_k, \mathcal{U}_k, j))\right\} \\
&\quad \leq \min_{\mathcal{U}_k} \left\{\alpha_\ell \bar{\Upsilon}(x_k, \mathcal{U}_k, j) + \alpha_\ell \alpha_{\ell-1} \frac{\tilde{\gamma}_\ell}{\tilde{\gamma}} \sigma_\ell \right. \\
&\qquad \times \left(1 + \sum_{\tau=1}^{\ell-1} \left(\prod_{l=1}^{\tau-1} \left(\sigma_{\ell-l-1} \frac{\tilde{\gamma}_{\ell-l-1}}{\tilde{\gamma}} \alpha_{\ell-l-2}\right) \frac{\phi_{\ell-l-1}}{(\phi_{\ell-l-1} + 1)}\right)\right. \\
&\qquad \left. \times \left(\sigma_{\ell-\tau-1} \frac{\tilde{\gamma}_{\ell-\tau-1}}{\tilde{\gamma}} \alpha_{\ell-\tau-2} - 1\right)\right) \tilde{\gamma} \bar{V}_\ell^{j+1}(\bar{\mathcal{F}}(x_k, \mathcal{U}_k, j)) \\
&\qquad + \frac{\alpha_\ell}{\phi_\ell + 1} \left(\alpha_{\ell-1} \frac{\tilde{\gamma}_\ell}{\tilde{\gamma}} \sigma_\ell \left(1 + \sum_{\tau=1}^{\ell-1} \left(\prod_{l=1}^{\tau-1} \left(\sigma_{\ell-l-1} \frac{\tilde{\gamma}_{\ell-l-1}}{\tilde{\gamma}} \alpha_{\ell-l-2}\right)\right.\right.\right. \\
&\qquad \left.\left.\left. \times \frac{\phi_{\ell-l-1}}{(\phi_{\ell-l-1} + 1)}\right) \left(\sigma_{\ell-\tau-1} \frac{\tilde{\gamma}_{\ell-\tau-1}}{\tilde{\gamma}} \alpha_{\ell-\tau-2} - 1\right)\right) - 1\right) \\
&\qquad \left. \times \left(\phi_\ell \bar{\Upsilon}(x_k, \mathcal{U}_k, j) - \tilde{\gamma} \bar{V}_\ell^{j+1}(\bar{\mathcal{F}}(x_k, \mathcal{U}_k, j))\right)\right. \\
&\quad = \alpha_\ell \left(1 + \sum_{\tau=1}^{\ell} \left(\prod_{l=1}^{\tau-1} \left(\sigma_{\ell-l} \frac{\gamma_{\ell-l}}{\gamma} \alpha_{\ell-l-1}\right) \frac{\phi_{\ell-l}}{(\phi_{\ell-l} + 1)}\right)\right. \\
&\qquad \left. \times \left(\sigma_{\ell-\tau} \frac{\gamma_{\ell-\tau}}{\gamma} \alpha_{\ell-\tau-1} - 1\right)\right) \bar{V}_{\ell+1}^{j+1}(x_k). \qquad (12.34)
\end{aligned}
$$

Then, the conclusion holds for $i = \ell$. Hence, inequality (12.32) holds for $i = 0, 1, \ldots$. The mathematical induction is complete. □

According to Theorem 12.1, the convergence of the iterative value function can be derived.

Theorem 12.2. *For $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$, let the iterative value function $V_i^{j+1}(x_k)$ and the iterative control law $u_i^j(x_k)$ be obtained by* (12.16)–(12.19)*. For $i = 0, 1, \ldots$, if the iterative discount factor $\tilde{\gamma}_i$ satisfies*

$$\tilde{\gamma}_i < \min\left\{1, \frac{\tilde{\gamma}(\phi_i + 1)}{\alpha_{i-1}\sigma_i\phi_i}\right\}, \tag{12.35}$$

where $\alpha_{i-1} = 1$ for $i = 0$, then the iterative value function $V_{i+1}^{j+1}(x_k)$ is convergent to a finite neighborhood of the optimal performance index function.

Proof. For (12.32) in Theorem 12.1, we let

$$\begin{aligned}\Theta_i = \sum_{\tau=1}^{i}\left(\prod_{l=1}^{\tau-1}\left(\left(\sigma_{i-l}\frac{\tilde{\gamma}_{i-l}}{\tilde{\gamma}}\alpha_{i-l-1}\right)\frac{\phi_{i-l}}{(\phi_{i-l}+1)}\right)\right)\\ \times\left(\sigma_{i-\tau}\frac{\tilde{\gamma}_{i-\tau}}{\tilde{\gamma}}\alpha_{i-\tau-1} - 1\right).\end{aligned} \tag{12.36}$$

For $\tau = 1, 2, \ldots, i$, letting

$$A_{i\tau} = \prod_{l=1}^{\tau}\left(\left(\sigma_{i-l}\frac{\tilde{\gamma}_{i-l}}{\tilde{\gamma}}\alpha_{i-l-1}\right)\frac{\phi_{i-l}}{(\phi_{i-l}+1)}\right) \tag{12.37}$$

and

$$B_{i\tau} = \prod_{l=1}^{\tau-1}\left(\left(\sigma_{i-l}\frac{\tilde{\gamma}_{i-l}}{\tilde{\gamma}}\alpha_{i-l-1}\right)\frac{\phi_{i-l}}{(\phi_{i-l}+1)}\right), \tag{12.38}$$

we can obtain $\Theta_i = \sum_{\tau=1}^{i} A_{i\tau} - \sum_{\tau=1}^{i} B_{i\tau}$. We can see that if $\sum_{\tau=1}^{i} A_{i\tau}$ is finite as $i \to \infty$, then we have $\sum_{\tau=1}^{\infty} B_{i\tau}$ is also finite, which means $\lim_{i\to\infty}\Theta_i$ is finite. According to (12.37), we can obtain

$$\frac{A_{i\tau}}{A_{(i-1)\tau}} = \sigma_i\frac{\tilde{\gamma}_i}{\tilde{\gamma}}\alpha_{i-1}\frac{\phi_i}{\phi_i + 1}. \tag{12.39}$$

For $i = 1, 2, \ldots$, if we choose the iterative discount factor $\tilde{\gamma}_i$ that satisfies $\tilde{\gamma}_i < \frac{\tilde{\gamma}(\phi_i+1)}{\alpha_{i-1}\sigma_i\phi_i}$, then we can obtain $\frac{A_{i\tau}}{A_{(i-1)\tau}} < 1$, which means

$\lim_{i\to\infty}\sum_{\tau=1}^{i} A_{i\tau} < \infty$. As the iterative discount factor $0 < \gamma_i < 1$, $i = 1, 2, \ldots$, we know that if we choose γ_i that satisfies (12.35), we can derive that the iterative value function $V_{i+1}^{j+1}(x_k)$ is convergent to a finite neighborhood of the optimal performance index function. □

12.4 Neural Network Implementation for the Error-Tolerant Iterative ADP Algorithm

In this section, neural networks are used to implement the error-tolerant iterative ADP algorithm. There are two neural networks, which are critic and action networks, respectively, which are both chosen as three-layer back-propagation (BP) networks. The whole structure diagram is shown in Fig. 12.3.

12.4.1 *The action and critic networks*

For $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$, the role of the action network is to approximate the iterative control law sequence $\mathcal{U}_i^{j+1}(x_k)$

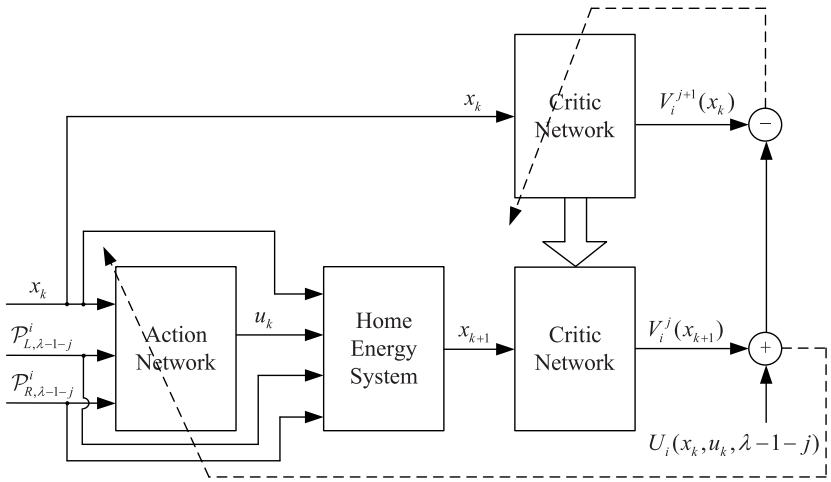

Fig. 12.3. The structure diagram of the error-tolerant iterative ADP algorithm.

in (12.20). The target of the action network can be defined as (12.16) and (12.18). The action network can be constructed by 4–15–1. Let $l = 0, 1, \ldots$ be the training step. The output of the action network can be expressed as $\hat{u}_i^{j,l}(x_k) = W_{ai}^j(l)\sigma(\mathcal{Z}_{a,i,k}^j)$, where $\mathcal{Z}_{a,i,k}^j = Y_a\left[x_k, \mathscr{P}_{L,\lambda-1-j}^i, \mathscr{P}_{R,\lambda-1-j}^i\right] + b_a$ and $\sigma(\cdot)$ is a sigmoid function. Here, $\mathscr{P}_{L,\lambda-1-j}^i$ and $\mathscr{P}_{R,\lambda-1-j}^i$ are both constants for any $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$. The action network weight update is expressed as follows

$$W_{ai}^j(l+1) = W_{ai}^j(l) - \beta_a \left[\frac{\partial E_{ai}^j(l)}{\partial W_{ai}^j(l)}\right], \tag{12.40}$$

where $E_{ai}^j(l) = \frac{1}{2}(e_{ai}^j(l))^2$, $e_{ai}^j(l) = \hat{u}_i^{j,l}(x_k) - u_i^j(x_k)$, and $\beta_a > 0$ is the learning rate of the action network. For $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$, the goal of the critic network is to obtain $V_i^{j+1}(x_k)$. The critic network can be constructed by 2–10–1. The output of the critic network can be expressed as $\hat{V}_i^{j+1,l}(x_k) = W_{ci}^j(l)\sigma(\mathcal{Z}_{ck})$, where $\mathcal{Z}_{ck} = Y_c x_k + b_c$ and $\sigma(\cdot)$ is a sigmoid function. The critic network weight update is expressed as follows

$$W_{ci}^j(l+1) = W_{ci}^j(l) - \alpha_c \left[\frac{\partial E_{ci}^j(l)}{\partial W_{ci}^j(l)}\right], \tag{12.41}$$

where $E_{ci}^j(l) = \frac{1}{2}(e_{ci}^j(l))^2$, $e_{ci}^j(l) = \hat{V}_i^{j+1,l}(x_k) - V_i^{j+1}(x_k)$, and $\alpha_c > 0$ is the learning rate of the critic network.

12.4.2 *Evaluation of the constants in the convergence criterion*

In Theorem 12.2, although convergence criterion (12.35) is presented for designing $\tilde{\gamma}_i$ to guarantee the convergence of the iterative value function, the constants ϕ_i, α_i, and σ_i are generally difficult to achieve directly by solving (12.29)–(12.31). In this section, numerical method is developed to evaluate the values of the constants, which is expressed in Algorithm 12.1.

Algorithm 12.1 Evaluate ϕ_i, α_i, and σ_i

Initialization:

Choose randomly an array of system states $(x_k^{(1)}, x_k^{(2)}, \ldots, x_k^{(p)})$ for a large integer p.

Collect enough data of C_k, $P_{L,k}$, and $P_{R,k}$.

Choose randomly control law sequences $(\mathcal{U}_k^{(1)}, \mathcal{U}_k^{(2)}, \ldots, \mathcal{U}_k^{(p)})$.

Iteration:

Block 1

1: Obtain the average electricity rate $\bar{C}_k$, the average load $\bar{P}_{L,k}$, and the average renewable energy $\bar{P}_{R,k}$.

2: Using $\bar{C}_k$, $\bar{P}_{L,k}$, and $\bar{P}_{R,k}$, construct $\bar{\mathcal{F}}(x_k, \mathcal{U}_k, j)$ and $\bar{\Upsilon}(x_k, \mathcal{U}_k, j)$, according to (12.24) and (12.25).

3: For $i = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$, solve (12.26) to obtain the expected iterative value function $\bar{V}_i^{j+1}(x_k)$ according to [4] until $\bar{V}_i^{j+1}(x_k)$ is convergent.

Block 2

4: Using $\mathcal{C}_k^i$, $\mathcal{P}_{L,k}^i$, and $\mathcal{P}_{R,k}^i$, construct $\Upsilon_i(x_k, \mathcal{U}_k, j)$, according to (12.27).

5: For $i = 0, 1, \ldots, j = 0, 1, \ldots, \lambda - 1$, obtain the iterative value function $V_i^{j+1}(x_k)$ by Algorithm 12.2.

6: Let $\ell = 1$.

7: According to $x_k^{(\ell)}$ and $\mathcal{U}_k^{(\ell)}$, compute $\bar{\mathcal{F}}(x_k^{(\ell)}, \mathcal{U}_k^{(\ell)}, j)$, $\bar{\Upsilon}(x_k^{(\ell)}, \mathcal{U}_k^{(\ell)}, j)$, $\mathcal{F}_i(x_k^{(\ell)}, \mathcal{U}_k^{(\ell)}, j)$, and $\Upsilon_i(x_k^{(\ell)}, \mathcal{U}_k^{(\ell)}, j)$ by (12.15), (12.25), (12.24), and (12.27), respectively.

8: Compute

$$\phi_i^{(\ell)} = \tilde{\gamma} \bar{V}_i^{j+1}(\bar{\mathcal{F}}(x_k^{(\ell)}, \mathcal{U}_k^{(\ell)}, j)) \Big/ \bar{\Upsilon}(x_k^{(\ell)}, \mathcal{U}_k^{(\ell)}, j), \quad (12.42)$$

$$\sigma_i^{(\ell)} = V_i^{j+1}(\mathcal{F}_i(x_k^{(\ell)}, \mathcal{U}_k^{(\ell)}, j)) \Big/ V_i^{j+1}(\bar{\mathcal{F}}(x_k^{(\ell)}, \mathcal{U}_k^{(\ell)}, j)), \quad (12.43)$$

and

$$\alpha_i^{(\ell)} = \Upsilon_i(x_k^{(\ell)}, \mathcal{U}_k^{(\ell)}, j) \Big/ \bar{\Upsilon}(x_k^{(\ell)}, \mathcal{U}_k^{(\ell)}, j). \quad (12.44)$$

9: If $\ell < p$, then go to Step 7; else go to next step.

10: Let $\phi_i = \max\{\phi_i^{(\ell)}\}$, $\sigma_i = \max\{\sigma_i^{(\ell)}\}$, and $\alpha_i = \max\{\alpha_i^{(\ell)}\}$, $\ell = 0, 1, \ldots, p$, and go to next step.

11: **return** ϕ_i, α_i, and σ_i.

12.4.3 *Training phase*

In this section, the error-tolerant iterative ADP algorithm implemented by action and critic networks is explained step by step and shown in Algorithm 12.2.

Algorithm 12.2 Error-Tolerant Iterative ADP Algorithm

Initialization:

Choose randomly an array of system states x_k.
Choose a semi-positive definite function $\Psi(x_k) \geq 0$.
Choose a convergence precision ζ.
Choose a positive constant $0 < \theta < 1$.

Iteration:

1: Let the iteration index $i = 0$ and let $V_0^0(x_k) = \Psi(x_k)$.
2: Let $\gamma_i = \gamma$.
3: For $j = 0, 1, \ldots, \lambda - 1$, train the action and critic networks to obtain $u_i^j(x_k)$ and $V_i^{j+1}(x_k)$ that satisfy (12.16)–(12.19).
4: Take $V_i^{j+1}(x_k)$ to Algorithm 12.1 and obtain ϕ_i, α_i, and σ_i.
5: If γ_i satisfies (12.35) then go to next step; else let $\gamma_i = \theta\gamma_i$ and go to Step 3.
6: If $\forall j = 0, 1, \ldots, \lambda - 1$, $|V_i^{j+1}(x_k) - V_{i-1}^{j+1}(x_k)| \leq \zeta$, then construct $\mathcal{U}_i^{j+1}(x_k)$ by (12.20) and go to next step; else let $i = i + 1$ and go to Step 2.
7: **return** $V_i^{j+1}(x_k)$ and $\mathcal{U}_i^{j+1}(x_k)$.

Remark 12.2. For $i = 0, 1, \ldots,$ Algorithm 12.2 computes $V_i^{j+1}(x_k)$ as the input of Algorithm 12.1. Then, Algorithm 12.1 outputs the evaluation value of ϕ_i, α_i, and σ_i which guarantees the effective improvement of the iterative value function. Thus, Algorithms 12.1 and 12.2 are interactive. On the other hand, for real smart home systems, the system model, especially for the battery efficiency in (12.4), is varying along with its implementation. This may cause the modeling for the system (12.13) inaccurate. Letting the varying smart home system be $\hat{F}_i(x_k, u_k, j)$, we have $\hat{F}_i(x_k, u_k, j) \neq F_i(x_k, u_k, j)$, $k = 0, 1, \ldots$ and $j = 0, 1, \ldots, \lambda - 1$. According to (12.14)–(12.15), we can derive $x_{k+\lambda} = \hat{\mathcal{F}}_i(x_k, \mathcal{U}_k, j)$. In this situation, it is desired to find a constant $\hat{\sigma}_i > 0$, which satisfies $V_i^{j+1}(\hat{\mathcal{F}}_i(x_k, \mathcal{U}_k, j)) \leq \hat{\sigma}_i V_i^{j+1}(\bar{\mathcal{F}}(x_k, \mathcal{U}_k, j))$, and Theorem 12.1 can be implemented.

12.5 Simulation Example

In this section, numerical experiments are displayed to show the performance of the error-tolerant iterative ADP algorithm. The trajectories of the electricity rate and the home load demand in 168 hours (a week) are shown in Figs. 12.4(a) and (b), respectively. The renewable energy is shown in Fig. 12.2. We can see that the electricity rate, the home load demand, and the renewable energy are all quasi-periodic functions with the period $\lambda = 24$. If there is no optimization,

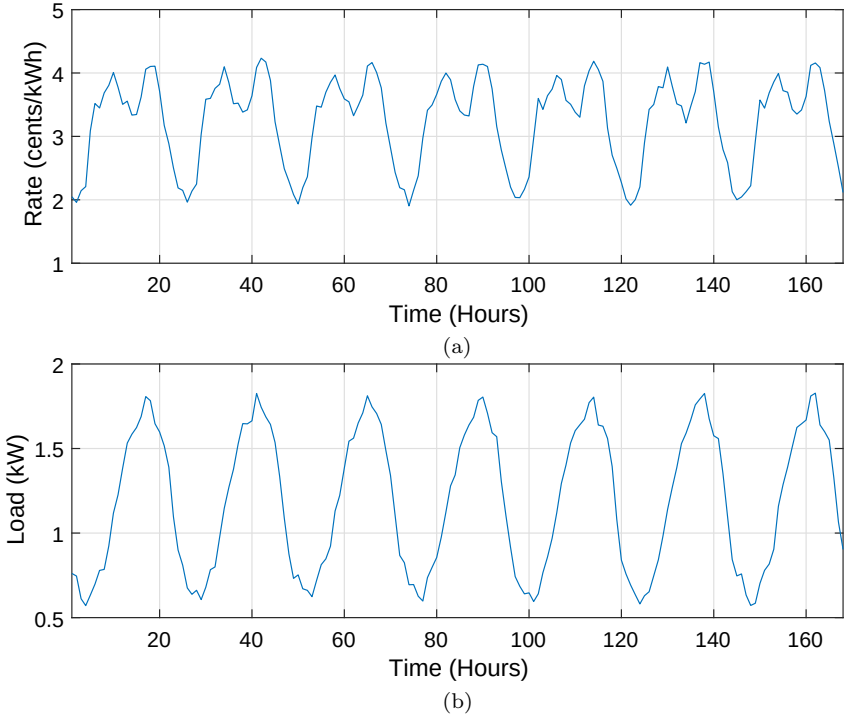

Fig. 12.4. Electricity rate and home load demand. (a) Real-time electricity rate in 168 hours. (b) Home load demand in 168 hours.

the original electricity cost, without considering the solar energy in a week, is 681.55 cents.

Let the upper and lower storage limits of the battery be $E_b^{\min} = 3$ kWh and $E_b^{\max} = 12$ kWh, respectively. The rated power output of the battery and the maximum charging/discharging rate is 3 kW. Let $\gamma = 0.98$. The initial level of the battery is 9 kWh. Let the performance index function be expressed as in (12.5), where we set $m_1 = 1$, $m_2 = 0.2$ and $m_3 = 0.2$. Let the initial function be $\Psi(x_k) = x_k^{\mathsf{T}} P x_k$, where $P = [0.11, -0.47; -0.47, 5.02]$. Let the initial state be $x_0 = [0.7, 9]^{\mathsf{T}}$. From Fig. 12.4(b), we can derive $0 \leq x_1 < 2$. According to the constraints of the battery, we have $3 \leq x_2 \leq 12$. We arbitrarily choose $p = 500$ states and control sequences. After normalization of the data, we implement Algorithm 12.1 to evaluate ϕ_i, α_i, and σ_i, respectively. According to the normalized data of the electricity rate, the home load demand, and the renewable energy, and we implement the developed error-tolerant iterative ADP algorithm in Algorithm 12.2 for $i = 20$ iterations to guarantee the computation precision $\varepsilon = 10^{-2}$. In each iteration, the parameter $\tilde{\gamma}_i$ is obtained by (12.35). According to $\tilde{\gamma}_i$, the discount factor γ_i can be obtained, where the trajectory of the discount factor γ_i, $i = 0, 1, \ldots$, is shown in Fig. 12.5. We can see that in each iteration, it requires a different discount factor γ_i to guarantee the convergence of the iterative value function.

Under the renewable energy, the optimal control of the battery is shown in Fig. 12.6. From Fig. 12.6, we can see that the battery charges when the load demand and rate are low and the renewable energy is high. The battery discharges when the load demand and rate are high and the renewable energy is low. The plot of the optimal battery energy is shown in Fig. 12.7. The total electricity cost in a week is reduced to 248.85 cents, which shows the effectiveness of the developed algorithm. From Figs. 12.4–12.7, we can see that if we choose an appropriate discount factor in each iteration, then the iterative value function can be guaranteed to converge.

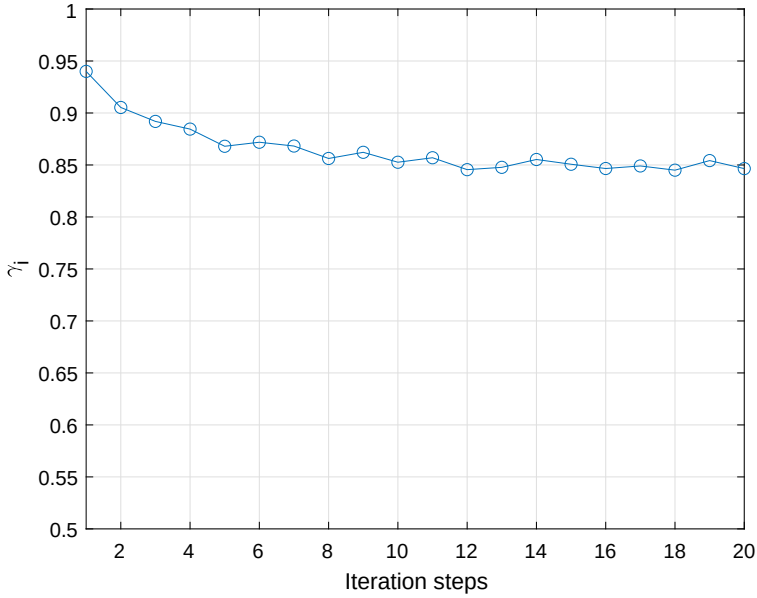

Fig. 12.5. The trajectory of γ_i.

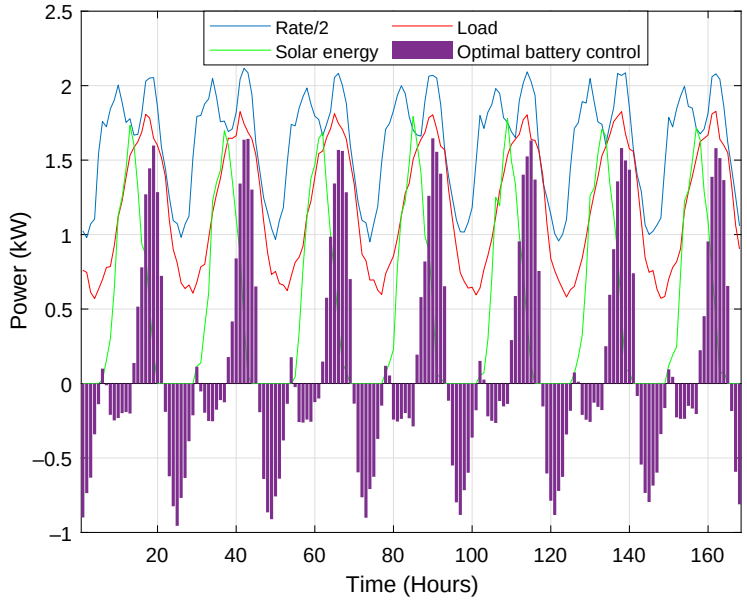

Fig. 12.6. Optimal battery control and management policy with renewable energy.

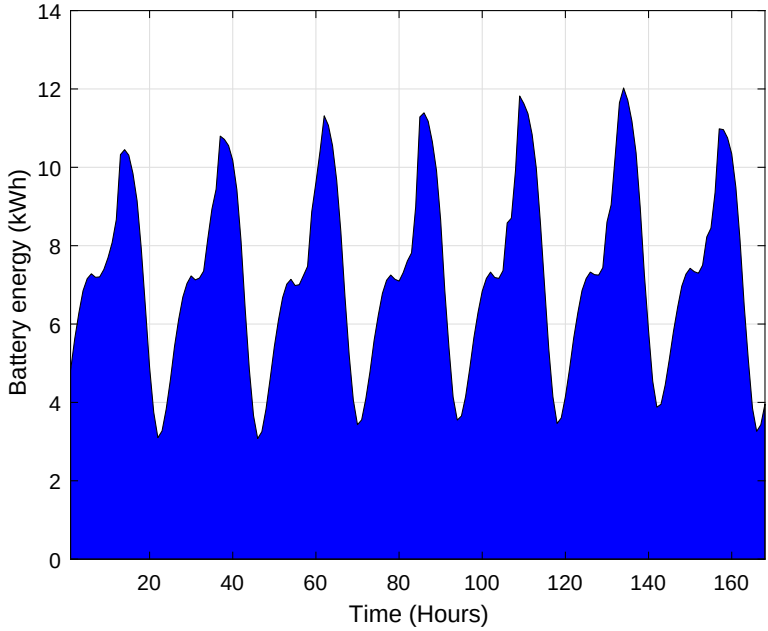

Fig. 12.7. Optimal control of the battery in a week.

12.6 Conclusion

In this chapter, an effective error-tolerant iterative ADP algorithm is developed for optimal battery control and management under the renewable energy. The present iterative ADP algorithm permits the electricity rate, the home load demand, and the renewable energy to be quasi-periodic functions to implement, instead of accurate periodic functions. It is proven that if the discount factor is chosen appropriately in each iteration, then the iterative value function converges to the finite neighborhood of optimum.

References

[1] Boaro, M., Fuselli, D., Angelis, F.D., Liu, D., Wei, Q. and Piazza, F. (2013). Adaptive dynamic programming algorithm for renewable energy scheduling and battery management. *Cognitive Computation* 5(2), 264–277.

[2] National renewable energy laboratory (NREL) of U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, Operated

by the Alliance for Sustainable Energy, LLC. Available: http://www.nrel.gov/rredc/.

[3] Huang, T. and Liu, D. (2013). A self-learning scheme for residential energy system control and management. *Neural Computing & Applications* 22(2), 259–269.

[4] Wei, Q., Liu. D. and Shi, G. (2015). A novel dual iterative Q-learning method for optimal battery management in smart residential environments. *IEEE Transactions on Industrial Electronics* 62(4), 2509–2518.

Chapter 13

Generalized Actor-Critic Learning Optimal Control in Smart Home Energy Management

13.1 Introduction

This chapter is concerned with a new generalized actor-critic learning optimal control method. It aims at the optimal energy control and management for smart home systems, which is expected to minimize the consumption cost for home users. In the present generalized actor-critic learning optimal control method, it is the first time that three iteration processes, which are global iteration, local iteration and interior iteration, respectively, are established to obtain the optimal energy control law. The main contribution of the developed method is to establish a common iteration structure for both value and policy iterations in ADP based on a control law sequence in each iteration for periodic time-varying systems, instead of a single control law. The monotonicity, convergence, and optimality of the iterative value function for the generalized actor-critic learning optimal control method are proven. Finally, numerical results and comparisons are displayed to show the superiority of the developed method.

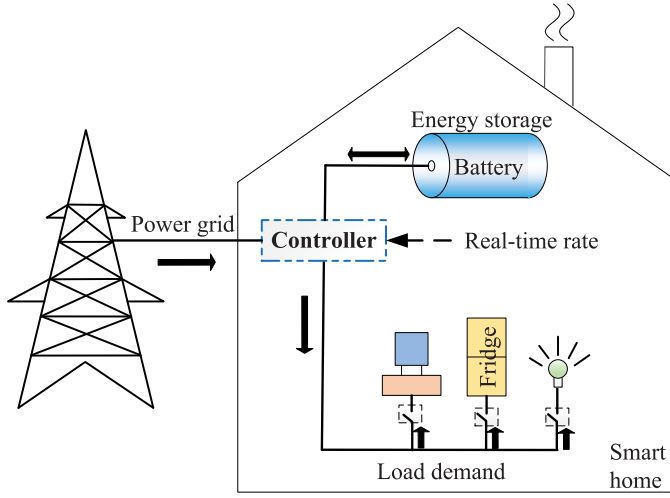

Fig. 13.1. Smart home energy system.

13.2 Preliminaries and Problem Statement

The smart home system is composed of the power grid, the load demand from users, and the energy storage equipment. In this paper, batteries are employed as the energy storage equipment. The construction of the smart home system is shown in Fig. 13.1, where the energy storage equipment can be designed as "Charging Mode", "Idle Mode", and "Discharging Mode", respectively. In this situation, how to optimize the control law of the energy storage equipment is the key to save the cost of the smart home system.

For time $t = 0, 1, \ldots$, letting B_t and $\mathcal{T}_t$ be the energy storage amount and the power output for the energy storage equipment, respectively, the model of the energy storage equipment is expressed as $B_{t+1} = B_t - \mathcal{T}_t \times \eta(\mathcal{T}_t)$, where we let $\mathcal{T}_t > 0$ and $\mathcal{T}_t < 0$ denote discharging and charging for the energy storage equipment, respectively. Let $\mathcal{T}_t = 0$ denote that the energy storage equipment is idle. Let the efficiency of energy storage equipment charging/discharging be expressed as $\eta(\mathcal{T}_t) = 0.898 - 0.173|\mathcal{T}_t|/\mathcal{T}_{\text{rate}}$, where $\mathcal{T}_{\text{rate}} > 0$ is the rated power output of the energy storage equipment.

Inspired by [1], the value function, which is expected to minimize, is expressed as

$$\sum_{t=0}^{\infty} \gamma^t \Big(\alpha_1 (\mathcal{E}_t \mathcal{P}_t)^2 + \alpha_2 (B_t - B^o)^2 + \alpha_3 \mathcal{T}_t^2 \Big), \tag{13.1}$$

where $\mathcal{E}_t$ is the real-time electricity rate and $\mathcal{P}_t$ is the total electricity power from the power grid. Let B^o denote the middle energy storage limit. Let α_1, α_2, and α_3 are positive numbers. Let $0 < \gamma < 1$ be the discount factor.

Let $x_{1,t} = \mathcal{P}_t$ and $x_{2,t} = B_t$ be the system states. Let $u_t = \mathcal{T}_t$ be the control of the system. According to the load balance [1] and the model of the energy storage equipment, the system function can be written as

$$x_{t+1} = \mathcal{F}(x_t, u_t, t) = \begin{pmatrix} \mathcal{Z}_t - u_t \\ x_{2t} - u_t \eta(u_t) \end{pmatrix}, \tag{13.2}$$

where $x_t = [x_{1,t}, x_{2,t}]^{\mathsf{T}}$ and $\mathcal{Z}_t$ denotes the load demand of the users. Letting $\underline{\omega}_t = (u_t, u_{t+1}, \ldots)$, the value function can be expressed as

$$V(x_0, \underline{\omega}_0, 0) = \sum_{t=0}^{\infty} \gamma^t \mathcal{H}(x_t, u_t, t), \tag{13.3}$$

where the utility function is defined as $\mathcal{H}(x_t, u_t, t) = x_t^{\mathsf{T}} \begin{bmatrix} \alpha_1 \mathcal{E}_t^2 & 0 \\ 0 & \alpha_2 \end{bmatrix} x_t + \alpha_3 u_t^2$. The optimal value function can be defined as

$$V^*(x_t, t) = \inf_{\underline{\omega}_t} \left\{ V(x_t, \underline{\omega}_t, t) \right\}. \tag{13.4}$$

13.3 Generalized Actor-Critic Learning Optimal Control Method

In this section, a new generalized actor-critic learning optimal control method is developed to obtain the optimal control law for smart home systems with property analysis.

13.3.1 *Derivation of the generalized actor-critic learning optimal control method*

From (13.2) and (13.3), it is shown that the load demand and the electricity rate are both time-varying parameters, which cause the optimal value function to be time-varying. For general users, it is known that the load demand and the electricity rate are periodic functions.

Assumption 13.1. The load demand $\mathcal{Z}_t$ and the electricity rate $\mathcal{E}_t$ are both periodic functions with the periods $\Theta = 24$ hours.

According to Assumption 13.1, the load demand and the electricity rate satisfy $\mathcal{Z}_t = \mathcal{Z}_{t+\Theta}, \mathcal{E}_t = \mathcal{E}_{t+\Theta}$. Inspired by [2], for $k \in \{0, \Theta, 2\Theta, \ldots\}$, define a new utility function as

$$\Psi\left(x_k, \mathfrak{A}_k\right) = \sum_{\theta=0}^{\Theta-1} \gamma^{\Theta} \mathcal{H}(x_{k+\theta}, u_{k+\theta}, \theta). \tag{13.5}$$

The optimal Q-type value function in the optimal performance index function can be written as

$$\mathcal{J}^*(x_k, \mathfrak{A}_k) = \Psi\left(x_k, \mathfrak{A}_k\right) + \vartheta \min_{\mathfrak{A}_{k+\Theta}} \mathcal{J}^*(x_{k+\Theta}, \mathfrak{A}_{k+\Theta}), \tag{13.6}$$

where it is defined as $\vartheta = \gamma^{\Theta}$. The optimal control law sequence can be expressed by $\mathscr{U}^*(x_k) = \arg\min_{\mathfrak{A}_k} \left\{\mathcal{J}^*(x_k, \mathfrak{A}_k)\right\}$.

Based on the above preparation, the generalized actor-critic learning optimal control method can be introduced. First, we define $\mathcal{N} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots\}$ as a set of positive integers, such that for any $i = 1, 2, \ldots, \mathcal{N}_i > 0$ is a positive integer. Let $i = 0, 1, \ldots$ be the global iteration index. Let $\mathscr{U}_0(x_k)$ be an initial control law, such that

$$\mathcal{J}_0(x_k, \mathfrak{A}_k) = \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{J}_0(x_{k+\Theta}, \mathscr{U}_0(x_{k+\Theta})), \tag{13.7}$$

where $\mathcal{J}_0(x_k, \mathfrak{A}_k)$ is the initial value function. For $i = 1$, the iterative control law sequence $\mathscr{U}_1(x_k)$ can be computed by

$$\mathscr{U}_1(x_k) = \arg\min_{\mathfrak{A}_k} \mathcal{J}_0(x_k, \mathfrak{A}_k). \tag{13.8}$$

Employing an local iteration index $j_1 = 0, 1, \ldots, \mathcal{N}_1 - 1$, the iterative cost function can be updated as

$$\mathcal{J}_{1,j_1+1}(x_k, \mathfrak{A}_k) = \Psi\left(x_k, \mathfrak{A}_k\right) + \vartheta \mathcal{J}_{1,j_1}(x_{k+\Theta}, \mathscr{U}_1(x_{k+\Theta})), \tag{13.9}$$

where $\mathcal{J}_{1,0}(x_k, \mathfrak{A}_k) = \mathcal{J}_0(x_k, \mathfrak{A}_k), \forall x_k, \mathfrak{A}_k$. Define $\mathcal{J}_1(x_k, \mathfrak{A}_k) = \mathcal{J}_{1,\mathcal{N}_1}(x_k, \mathfrak{A}_k)$.

For $i = 2, 3, \ldots$, the iterative control law sequence is computed as

$$\mathscr{U}_i(x_k) = \arg\min_{\mathfrak{A}_k} \mathcal{J}_{i-1}(x_k, \mathfrak{A}_k). \tag{13.10}$$

Letting $\mathcal{J}_{i,0}(x_k, \mathfrak{A}_k) = \mathcal{J}_{i-1}(x_k, \mathfrak{A}_k)$, for $j_i = 0, 1, \ldots, \mathcal{N}_i$, $\mathcal{N}_i \in \mathcal{N}$, the iterative cost function is updated by

$$\mathcal{J}_{i,j_i+1}(x_k, \mathfrak{A}_k) = \Psi\left(x_k, \mathfrak{A}_k\right) + \vartheta \mathcal{J}_{i,j_i}(x_{k+\Theta}, \mathscr{U}_i(x_{k+\Theta})), \tag{13.11}$$

where we define

$$\mathcal{J}_i(x_k, \mathfrak{A}_k) = \mathcal{J}_{i,\mathcal{N}_i}(x_k, \mathfrak{A}_k). \tag{13.12}$$

It is declared that the iterative control law sequence $\mathscr{U}_i(x_k)$, $i = 0, 1, \ldots$, is not obtainable by directly solving (13.8) and (13.10), as $\mathscr{U}_i(x_k)$ is an iterative control law sequence. Thus, a new interior iteration is introduced to construct the sequence.

For $i = 0, 1, \ldots$, let $l_i = 0, 1, \ldots, \Theta - 1$ be the interior iteration index. According to (13.3), for $l_i = 0, 1, \ldots, \Theta - 1$, we define a new system as

$$x_{k+1} = \mathscr{F}(x_k, u_k, l_i) = \begin{pmatrix} \mathcal{Z}_{\Theta-1-l_i} - u_k \\ x_{2,k} - u_k \eta(u_k) \end{pmatrix}. \tag{13.13}$$

For $l_i = 0, 1, \ldots, \Theta - 1$, we let

$$\mathcal{U}(x_k, u_k, l_i) = x_k^{\mathsf{T}} \mathscr{H}_{\Theta-1-l_i} x_k + \alpha_3 u_k^2, \tag{13.14}$$

where $\mathscr{H}_{\Theta-1-l_i} = \begin{bmatrix} \alpha_1 (\mathcal{E}_{\Theta-1-l_i})^2 & 0 \\ 0 & \alpha_2 \end{bmatrix}$.

For $i = 1, 2, \ldots$, we define a new interior iterative cost function as $V_i^{l_i}(x_k)$. For $i = 1, 2, \ldots$ and $l_i = 0$, we define

$$V_i^0(x_k) = \mathcal{J}_{i-1}^o(x_k, \mathscr{U}_{i-1}(x_k)), \tag{13.15}$$

where for $i = 1$, we let $\mathcal{J}^o_{i-1}(x_k, \mathscr{U}_0(x_k)) = \mathcal{J}_0(x_k, \mathscr{U}_0(x_k))$, and for $i = 2, 3, \ldots$, we let $\mathcal{J}^o_{i-1}(x_k, \mathscr{U}_{i-1}(x_k)) = \mathcal{J}_{i-1,\mathcal{N}_{i-1}-1}(x_k, \mathscr{U}_{i-1}(x_k))$.

Then, for $i = 1, 2, \ldots$ and $l_i = 0, 1, \ldots, \Theta - 1$, the interior iterative control law is obtained by

$$v_i^{l_i}(x_k) = \arg\min_{u_k}\{\mathcal{U}(x_k, u_k, l_i) + \gamma V_i^{l_i}(x_{k+1})\}, \tag{13.16}$$

where $x_{k+1} = \mathcal{F}(x_k, u_k, l_i)$ is defined in (13.13) and the utility function $\mathcal{U}(x_k, u_k, l_i)$ is defined in (13.14). Then, the interior iterative cost function is computed as

$$V_i^{l_i+1}(x_k) = \mathcal{U}(x_k, v_i^{l_i}(x_k), l_i) + \gamma V_i^{l_i}(\mathcal{F}(x_k, v_i^{l_i}(x_k), l_i)). \tag{13.17}$$

For $i = 0, 1, \ldots$, the iterative control law sequence $\mathscr{U}_i(x_k)$ can be constructed by

$$\mathscr{U}_i(x_k) = \left\{v_i^{\Theta-1}(x_k), v_i^{\Theta-2}(x_k), \ldots, v_i^0(x_k)\right\}. \tag{13.18}$$

13.3.2 *Properties of the generalized actor-critic learning optimal control method*

In this subsection, the properties of the generalized actor-critic learning optimal control method are analyzed. First, the property of the interior iteration is presented.

Theorem 13.1. *For $i = 0, 1, \ldots$, defining the iterative control law sequence $\mathscr{U}_i(x_k)$ as in* (13.18)*, where $v_i^{l_i}(x_k)$, $l_i = 0, 1, \ldots, \Theta - 1$, is obtained according to* (13.15)–(13.17)*, then $\mathscr{U}_i(x_k)$ satisfies* (13.8) *for $i = 1$ and* (13.10) *for $i = 2, 3, \ldots$.*

Proof. Mathematical induction is employed to prove the statement. First, for $i = 0, 1, \ldots$, we define

$$\Psi(x_k, \mathscr{U}_i(x_k)) = \sum_{l_i=0}^{\Theta-1} \gamma^{l_i} \mathcal{U}(x_k, v_i^{l_i}(x_k), l_i). \tag{13.19}$$

For $i = 1$ and $l_1 = 0$, we know $V_1^0(x_k) = \mathcal{J}_0(x_k, \mathscr{U}_0(x_k))$, where $\mathscr{U}_0(x_k)$ is a given control law sequence. Then, we can solve the control

law $v_1^0(x_k)$ by

$$v_1^0(x_k) = \arg\min_{u_k}\{\mathcal{U}(x_k, u_k, 0) + \gamma V_i^0(x_{k+1})\}, \tag{13.20}$$

where $x_{k+1} = \mathcal{F}(x_k, u_k, 0)$. For any $l_1 = 0, 1, \ldots, \Theta - 1$, we can get

$$\begin{aligned} V_1^{l_1+1}(x_k) &= \mathcal{U}(x_k, v_1^{l_1}(x_k), l_1) + \gamma V_1^{l_1}(x_{k+1}) \\ &= \min_{u_k}\{\mathcal{U}(x_k, u_k, l_1) + \gamma V_1^{l_1}(x_{k+1})\} \\ &= \min_{u_k}\big\{\mathcal{U}(x_k, u_k, l_1) \\ &\quad + \gamma \min_{u_{k+1}}\big\{\mathcal{U}(x_{k+1}, u_{k+1}, l_1 - 1) \\ &\quad + \cdots + \gamma \min_{u_{k+l_1}}\{\mathcal{U}(x_{k+1}, u_{k+l_1}, 0) \\ &\quad + \gamma V_1^0(x_{k+l_1+1})\}\cdots\big\}\big\}. \end{aligned} \tag{13.21}$$

Letting $l_1 = \Theta - 1$, we have

$$\begin{aligned} V_1^{\Theta}(x_k) &= \Psi(x_k, \mathscr{U}_1(x_k)) + \vartheta V_1^0(\mathcal{F}(x_k, \mathscr{U}_1(x_k))) \\ &= \min_{\mathfrak{A}_k}\{\Psi(x_k, \mathfrak{A}_k) + \vartheta V_1^0(x_{k+\Theta})\} \\ &= \min_{\mathfrak{A}_k}\{\Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{J}_0(x_{k+\Theta}, \mathscr{U}_0(x_{k+\Theta}))\} \\ &= \min_{\mathfrak{A}_k} \mathcal{J}_0(x_k, \mathfrak{A}_k). \end{aligned} \tag{13.22}$$

According to (13.22), we can obtain (13.8) for $i = 1$. Assume that the conclusion holds for $i = \tau - 1$, $\tau = 2, 3, \ldots$, i.e.,

$$\mathscr{U}_{\tau-1}(x_k) = \arg\min_{\mathfrak{A}_k} \mathcal{J}_{\tau-2}(x_k, \mathfrak{A}_k). \tag{13.23}$$

Then, for $i = \tau$, according to (13.15), we have

$$V_\tau^0(x_k) = \mathcal{J}_{\tau-1}(x_k, \mathscr{U}_{\tau-1}(x_k)). \tag{13.24}$$

For $l_\tau = 0, 1, \ldots, \Theta - 1$, it can be derived as

$$\begin{aligned} V_\tau^{l_\tau+1}(x_k) &= \mathcal{U}(x_k, v_\tau^{l_\tau}(x_k), l_\tau) + \gamma V_\tau^{l_\tau}(x_{k+1}) \\ &= \min_{u_k}\{\mathcal{U}(x_k, u_k, l_\tau) \end{aligned}$$

$$
\begin{aligned}
&+\gamma \min_{u_{k+1}}\{\mathcal{U}(x_{k+1},u_{k+1},l_\tau-1)\\
&+\cdots+\gamma \min_{u_{k+l_\tau}}\{\mathcal{U}(x_{k+1},u_{k+l_\tau},l_\tau-1)\\
&+\gamma V_\tau^0(x_{k+l_\tau+1})\}\cdots\}\}\\
=&\min_{u_k,u_{k+1},\ldots,u_{k+l_\tau}}\left\{\sum_{\ell=0}^{l_\tau}\gamma^\ell \mathcal{U}(x_{k+\ell},u_{k+\ell},l_\tau-\ell)\right.\\
&\left.+\gamma^{l_\tau}V_\tau^0(x_{k+\ell+1})\right\}. \qquad (13.25)
\end{aligned}
$$

Letting $l_\tau=\Theta-1$, according to (13.24), we can obtain

$$
\begin{aligned}
V_\tau^\Theta(x_k)&=\Psi(x_k,\mathscr{U}_\tau(x_k))+\vartheta V_\tau^0(\mathcal{F}(x_k,\mathscr{U}_\tau(x_k)))\\
&=\min_{\mathfrak{A}_k}\{\Psi(x_k,\mathfrak{A}_k)\\
&\quad+\vartheta\mathcal{J}_{\tau-1,\mathcal{N}_{\tau-1}-1}(x_{k+\Theta},\mathscr{U}_{\tau-1}(x_{k+\Theta}))\}\\
&=\min_{\mathfrak{A}_k}\mathcal{J}_{\tau-1}(x_k,\mathfrak{A}_k). \qquad (13.26)
\end{aligned}
$$

For $i=\tau$, according to (13.26), the iterative control law sequence $\mathscr{U}_\tau(x_k)$ can be written as

$$
\mathscr{U}_\tau(x_k)=\arg\min_{\mathfrak{A}_k}\mathcal{J}_{\tau-1}(x_k,\mathfrak{A}_k). \qquad (13.27)
$$

Thus, the conclusion holds for $i=\tau$. Hence, $\mathscr{U}_i(x_k)$ satisfies (13.8) for $i=1$ and (13.10) for $i=2,3,\ldots$. The mathematical induction is complete. □

In the following, we will show that for any iteration indices i and j_i, the monotonicity and convergence of the iterative cost function in local and global iterations can be guaranteed.

Theorem 13.2. *Let $\mathscr{U}_0(x_k)$ be an arbitrary admissible control law which satisfies* (13.7). *Let $\mathcal{N}=\{N_1,N_2,\ldots\}$ be an sequence, where $N_i>0$, $i=1,2,\ldots$, is a positive integer. For $i=0,1,\ldots$ and $j_i=1,2,\ldots,N_i$, let $\mathcal{J}_{i,j_i}(x_k,\mathfrak{A}_k)$ and $\mathscr{U}_i(x_k)$ be updated according to* (13.7)–(13.12). *Then, the iterative cost function $\mathcal{J}_{i,j_i}(x_k,\mathfrak{A}_k)$ is monotonically non-increasing both for i and j_i.*

Proof. The statements can be proven by mathematical induction. Consider $i = 1$ and $j_1 = 0$. According to (13.8), we have

$$\begin{aligned}\mathcal{J}_{1,1}(x_k, \mathfrak{A}_k) &= \Psi\left(x_k, \mathfrak{A}_k\right) + \vartheta \mathcal{J}_{1,0}(x_{k+\Theta}, \mathscr{U}_1(x_{k+\Theta})) \\ &\leq \Psi\left(x_k, \mathfrak{A}_k\right) + \vartheta \mathcal{J}_0(x_{k+\Theta}, \mathscr{U}_0(x_{k+\Theta})) \\ &= \mathcal{J}_{1,0}(x_k, \mathfrak{A}_k). \end{aligned} \tag{13.28}$$

Assume that the inequality

$$\mathcal{J}_{1,j_1}(x_k, \mathfrak{A}_k) \leq \mathcal{J}_{1,j_1-1}(x_k, \mathfrak{A}_k) \tag{13.29}$$

holds for $j_1 = 1, 2, \ldots, N_1 - 1$. Then, for $j_1 + 1$, we can obtain

$$\begin{aligned}\mathcal{J}_{1,j_1+1}(x_k, \mathfrak{A}_k) &= \Psi\left(x_k, \mathfrak{A}_k\right) + \vartheta \mathcal{J}_{1,j_1}(x_{k+\Theta}, \mathscr{U}_1(x_{k+\Theta})) \\ &\leq \Psi\left(x_k, \mathfrak{A}_k\right) + \vartheta \mathcal{J}_{1,j_1-1}(x_{k+\Theta}, \mathscr{U}_1(x_{k+\Theta})) \\ &= \mathcal{J}_{1,j_1}(x_k, \mathfrak{A}_k), \end{aligned} \tag{13.30}$$

which implies that

$$\mathcal{J}_{1,j_1+1}(x_k, \mathfrak{A}_k) \leq \mathcal{J}_{1,j_1}(x_k, \mathfrak{A}_k) \tag{13.31}$$

holds for all $j_1 = 0, 1, \ldots, N_1 - 1$. Letting

$$\mathscr{U}_2(x_k) = \arg\min_{\mathfrak{A}_k} \mathcal{J}_1(x_k, \mathfrak{A}_k), \tag{13.32}$$

according to (13.12), it can be derived that

$$\begin{aligned}\mathcal{J}_{2,1}(x_k, \mathfrak{A}_k) &= \Psi(x_k, \mathfrak{A}_k) + \mathcal{J}_{2,0}(x_{k+\Theta}, \mathscr{U}_2(x_{k+\Theta})) \\ &\leq \Psi(x_k, \mathfrak{A}_k) + \mathcal{J}_{1,N_1-1}(x_{k+\Theta}, \mathscr{U}_1(x_{k+\Theta})) \\ &= \mathcal{J}_{1,N_1}(x_k, \mathfrak{A}_k) \\ &= \mathcal{J}_{2,0}(x_k, \mathfrak{A}_k). \end{aligned} \tag{13.33}$$

Assume that the inequality $\mathcal{J}_{2,j_2}(x_k, \mathfrak{A}_k) \leq \mathcal{J}_{2,j_2-1}(x_k, \mathfrak{A}_k)$ holds for $j_2 = 1, 2, \ldots, N_2 - 1$. Then for $j_2 + 1$, we have

$$\begin{aligned}\mathcal{J}_{2,j_2+1}(x_k, \mathfrak{A}_k) &= \Psi\left(x_k, \mathfrak{A}_k\right) + \vartheta \mathcal{J}_{2,j_2}(x_{k+\Theta}, \mathscr{U}_2(x_{k+\Theta})) \\ &\leq \Psi\left(x_k, \mathfrak{A}_k\right) + \vartheta \mathcal{J}_{2,j_2-1}(x_{k+\Theta}, \mathscr{U}_2(x_{k+\Theta})) \\ &= \mathcal{J}_{2,j_2}(x_k, \mathfrak{A}_k). \end{aligned} \tag{13.34}$$

Thus, we have $\mathcal{J}_{2,j_2+1}(x_k,\mathfrak{A}_k) \leq \mathcal{J}_{2,j_2}(x_k,\mathfrak{A}_k)$ holds for all $j_2 = 0,1,\ldots,N_2-1$.

For $i=\varsigma$, $\varsigma=0,1,\ldots$, we assume that

$$\mathcal{J}_{\varsigma,j_\varsigma}(x_k,\mathfrak{A}_k) \leq \mathcal{J}_{\varsigma,j_\varsigma-1}(x_k,\mathfrak{A}_k) \tag{13.35}$$

holds, for $j_\varsigma = 1,2,\ldots,j_\varsigma$. Then, we can derive

$$\begin{aligned}\mathcal{J}_{\varsigma,j_\varsigma+1}(x_k,\mathfrak{A}_k) &= \Psi(x_k,\mathfrak{A}_k) + \vartheta\mathcal{J}_{\varsigma,j_\varsigma}(x_{k+\Theta},\mathscr{U}_\varsigma(x_{k+\Theta})) \\ &\leq \Psi(x_k,\mathfrak{A}_k) + \vartheta\mathcal{J}_{\varsigma,j_\varsigma-1}(x_{k+\Theta},\mathscr{U}_\varsigma(x_{k+\Theta})) \\ &= \mathcal{J}_{\varsigma,j_\varsigma}(x_k,\mathfrak{A}_k),\end{aligned} \tag{13.36}$$

which shows

$$\mathcal{J}_{\varsigma,j_\varsigma+1}(x_k,\mathfrak{A}_k) \leq \mathcal{J}_{\varsigma,j_\varsigma}(x_k,\mathfrak{A}_k) \tag{13.37}$$

for all $j_\varsigma = 0,1,\ldots,N_\varsigma-1$. Considering $i=\varsigma+1$, the iterative control law sequence is computed by

$$\mathscr{U}_{\varsigma+1}(x_k) = \arg\min_{\mathfrak{A}_k} \mathcal{J}_\varsigma(x_k,\mathfrak{A}_k). \tag{13.38}$$

It can be derived as

$$\begin{aligned}\mathcal{J}_{\varsigma+1,1}(x_k,\mathfrak{A}_k) &= \Psi(x_k,\mathfrak{A}_k) + \vartheta\mathcal{J}_{\varsigma+1,0}(x_{k+\Theta},\mathscr{U}_{\varsigma+1}(x_{k+\Theta})) \\ &\leq \Psi(x_k,\mathfrak{A}_k) + \vartheta\mathcal{J}_{\varsigma,j_{N_\varsigma}-1}(x_{k+\Theta},\mathscr{U}_\varsigma(x_{k+\Theta})) \\ &= \mathcal{J}_{\varsigma+1,0}(x_k,\mathfrak{A}_k).\end{aligned} \tag{13.39}$$

Assume that the inequality

$$\mathcal{J}_{\varsigma+1,j_{\varsigma+1}}(x_k,\mathfrak{A}_k) \leq \mathcal{J}_{\varsigma+1,j_{\varsigma+1}-1}(x_k,\mathfrak{A}_k) \tag{13.40}$$

holds for $j_{\varsigma+1} = 1,2,\ldots,N_{\varsigma+1}-1$. Then considering $j_{\varsigma+1}+1$, we can derive

$$\begin{aligned}&\mathcal{J}_{\varsigma+1,j_{\varsigma+1}+1}(x_k,\mathfrak{A}_k) \\ &\quad= \Psi(x_k,\mathfrak{A}_k) + \vartheta\mathcal{J}_{\varsigma+1,j_{\varsigma+1}}(x_{k+\Theta},\mathscr{U}_{\varsigma+1}(x_{k+\Theta})) \\ &\quad\leq \Psi(x_k,\mathfrak{A}_k) + \vartheta\mathcal{J}_{\varsigma+1,j_{\varsigma+1}-1}(x_{k+\Theta},\mathscr{U}_{\varsigma+1}(x_{k+\Theta})) \\ &\quad= \mathcal{J}_{\varsigma+1,j_{\varsigma+1}}(x_k,\mathfrak{A}_k).\end{aligned} \tag{13.41}$$

The mathematical induction is complete. From mathematical induction process (13.28)–(13.41), for $i = 1, 2, \ldots, j_i = 0, 1, \ldots, N_i$, $N_i \in \mathcal{N}$ we can derive $\mathcal{J}_{i,j_i+1}(x_k, \mathfrak{A}_k) \leq \mathcal{J}_{i,j_i}(x_k, \mathfrak{A}_k)$ and $\mathcal{J}_{i+1,j_{(i+1)}}(x_k, \mathfrak{A}_k) \leq \mathcal{J}_{i,j_i}(x_k, \mathfrak{A}_k)$. The proof is complete. □

In the next theorem, the optimality of the converged cost function, which is a key property to declare the effectiveness of the generalized actor-critic learning optimal control method, will be analyzed.

Theorem 13.3. *Let $\mathcal{N} = \{N_1, N_2, \ldots\}$ be a sequence, where $N_i > 0$, $i = 1, 2, \ldots,$ is a positive integer. For $i = 0, 1, \ldots$ and $j_i = 1, 2, \ldots, N_i$, let $\mathcal{J}_{i,j_i}(x_k, \mathfrak{A}_k)$ and $\mathscr{U}_i(x_k)$ be updated according to* (13.7)–(13.12). *Initialized by an arbitrary admissible control law $\mathscr{U}_0(x_k)$ which satisfies* (13.7), *the iterative cost function $\mathcal{J}_{i,j_i}(x_k, \mathfrak{A}_k)$ converges to the optimal value function $V^*(x_k, \mathfrak{A}_k)$, as $i \to \infty$, i.e.,*

$$\lim_{i\to\infty} \mathcal{J}_{i,j_i}(x_k, \mathfrak{A}_k) = V^*(x_k, \mathfrak{A}_k), \tag{13.42}$$

which satisfies the HJB equation (13.6).

Proof. The statement can be proven by four steps.

(1) *Show that the limit of the iterative cost function $V_{i,j_i}(x_k, \mathfrak{A}_k)$ exits as $i \to \infty$.*

First, according to (13.7), giving a positive integer $\mathcal{M}$, we can derive that

$$\begin{aligned}\mathcal{J}_0(x_k, \mathfrak{A}_k) = \Psi(x_k, \mathfrak{A}_k) + \sum_{\tau=0}^{\mathcal{M}-1} \vartheta^\tau \Psi(x_{k+\tau\Theta}, \mathscr{U}_0(k+\tau\Theta)) \\ + \vartheta^{\mathcal{M}} \mathcal{J}_0(x_{k+\mathcal{M}\Theta}, \mathscr{U}_0(x_{k+\mathcal{M}\Theta})).\end{aligned} \tag{13.43}$$

As $0 < \vartheta < 1$, letting $\mathcal{M} \to \infty$, we can obtain

$$\mathcal{J}_0(x_k, \mathfrak{A}_k) = \Psi(x_k, \mathfrak{A}_k) + \sum_{\tau=0}^{\infty} \vartheta^\tau \Psi(x_{k+\tau\Theta}, \mathscr{U}_0(k+\tau\Theta)). \tag{13.44}$$

As the utility function $U(x_k, u_k, k)$ is positive definite for x_k and u_k, which implies that $\Psi(x_k, \mathfrak{A}_k)$ is positive for x_k and $\mathfrak{A}_k$, it can be derived that the initial iterative value function $\mathcal{J}_0(x_k, \mathfrak{A}_k) > 0$ for all $x_k \neq 0$ or $\mathfrak{A}_k \neq 0$. According to Theorem 13.2, the iterative cost function $\mathcal{J}_{i,j_i}(x_k, \mathfrak{A}_k)$ is a monotonically non-increasing function

for i and j_i. It implies that the function $\mathcal{J}_i(x_k, \mathfrak{A}_k)$ is monotonically non-increasing and lower bounded which shows that the limit of the iterative cost function $\mathcal{J}_i(x_k, \mathfrak{A}_k)$ exists, i.e.,

$$\mathcal{J}_\infty(x_k, \mathfrak{A}_k) = \lim_{i\to\infty} \mathcal{J}_i(x_k, \mathfrak{A}_k). \tag{13.45}$$

On the other hand, $\{\mathcal{J}_i(x_k, \mathfrak{A}_k)\}$ is the subsequence of $\{\mathcal{J}_{i,j_i}(x_k, \mathfrak{A}_k)\}$. It implies that the two sequences $\{\mathcal{J}_i(x_k, \mathfrak{A}_k)\}$ and $\{\mathcal{J}_{i,j_i}(x_k, \mathfrak{A}_k)\}$ converge to the same limit, i.e.,

$$\mathcal{J}_\infty(x_k, \mathfrak{A}_k) = \lim_{i\to\infty} \mathcal{J}_{i,j_i}(x_k, \mathfrak{A}_k). \tag{13.46}$$

(2) *Prove that as $i \to \infty$ the limit of the iterative cost function $\mathcal{J}_i(x_k, \mathfrak{A}_k)$ satisfies the optimality equation* (13.6).

According to Theorem 13.2, for $i = 0, 1, \ldots$, it can be derived as $\mathcal{J}_{i+1}(x_k, \mathfrak{A}_k) \le \mathcal{J}_i(x_k, \mathfrak{A}_k)$, which implies

$$\begin{aligned}\mathcal{J}_{i+1}(x_k, \mathfrak{A}_k) &= \mathcal{J}_{i,N_i}(x_k, \mathfrak{A}_k)\\ &\le \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{J}_{i-1,0}(x_{k+\Theta}, \mathscr{U}_i(x_{k+\Theta}))\\ &= \Psi(x_k, \mathfrak{A}_k) + \vartheta \min_{\mathfrak{A}_{k+\Theta}} \mathcal{J}_{i-1}(x_{k+\Theta}, \mathfrak{A}_{k+\Theta}).\end{aligned} \tag{13.47}$$

Then, we can get

$$\begin{aligned}\mathcal{J}_\infty(x_k, \mathfrak{A}_k) &= \lim_{i\to\infty} \mathcal{J}_{i+1}(x_k, \mathfrak{A}_k) \le \mathcal{J}_{i+1}(x_k, \mathfrak{A}_k)\\ &\le \Psi(x_k, \mathfrak{A}_k) + \vartheta \min_{\mathfrak{A}_{k+\Theta}} \mathcal{J}_{i-1}(x_{k+\Theta}, \mathfrak{A}_{k+\Theta}).\end{aligned} \tag{13.48}$$

Letting $i \to \infty$, we can obtain

$$\mathcal{J}_\infty(x_k, \mathfrak{A}_k) \le \Psi(x_k, \mathfrak{A}_k) + \min_{\mathfrak{A}_{k+\Theta}} \mathcal{J}_\infty(x_{k+\Theta}, \mathfrak{A}_{k+\Theta}). \tag{13.49}$$

Given an $\varepsilon > 0$, there exists an iteration index $\sigma > 1$, such that

$$\mathcal{J}_\sigma(x_k, \mathfrak{A}_k) - \varepsilon \le \mathcal{J}_\infty(x_k, \mathfrak{A}_k). \tag{13.50}$$

Then, we can obtain

$$\begin{aligned}\mathcal{J}_\infty(x_k,\mathfrak{A}_k) &\geq \Psi(x_k,\mathfrak{A}_k)+\vartheta\mathcal{J}_\sigma(x_{k+\Theta},\mathfrak{A}_{k+\Theta})-\varepsilon\\ &\geq \Psi(x_k,\mathfrak{A}_k)+\vartheta\mathcal{J}_\infty(x_{k+\Theta},\mathfrak{A}_{k+\Theta})-\varepsilon\\ &\geq \Psi(x_k,\mathfrak{A}_k)+\vartheta\min_{\mathfrak{A}_{k+\Theta}}\mathcal{J}_\infty(x_{k+\Theta},\mathfrak{A}_{k+\Theta})-\varepsilon. \end{aligned}\tag{13.51}$$

For arbitrary $\varepsilon>0$, we have

$$\mathcal{J}_\infty(x_k,\mathfrak{A}_k)\geq \Psi(x_k,\mathfrak{A}_k)+\vartheta\min_{\mathfrak{A}_{k+\Theta}}\mathcal{J}_\infty(x_{k+\Theta},\mathfrak{A}_{k+\Theta}).\tag{13.52}$$

Combining (13.49) and (13.52), for all $x_k\in\Omega_x$, we can obtain

$$\mathcal{J}_\infty(x_k,\mathfrak{A}_k)=\Psi(x_k,\mathfrak{A}_k)+\vartheta\min_{\mathfrak{A}_{k+\Theta}}\mathcal{J}_\infty(x_{k+\Theta},\mathfrak{A}_{k+\Theta}).\tag{13.53}$$

(3) *Show that the cost function $\mathcal{J}_\infty(x_k,\mathfrak{A}_k)$ is not larger than the cost function constructed by any admissible control law sequence.*

Define a new iterative cost function $\mathcal{Q}_{i,j_i+1}(x_k,\mathfrak{A}_k)$. For $i=0$, we let

$$\mathcal{Q}_0(x_k,\mathfrak{A}_k)=\mathcal{J}_0(x_k,\mathfrak{A}_k).\tag{13.54}$$

For $i=1,2,\ldots$ and $j_i=0,1,\ldots,N_i-1$, $N_i\in\mathcal{N}$, letting $\mathcal{Q}_{i,0}(x_k,\mathfrak{A}_k)=\mathcal{Q}_{i-1}(x_k,\mathfrak{A}_k)$, the iterative cost function $\mathcal{Q}_{i,j_i}(x_k,\mathfrak{A}_k)$ is designed to update by

$$\mathcal{Q}_{i,j_i+1}(x_k,\mathfrak{A}_k)=\Psi(x_k,\mathfrak{A}_k)+\vartheta\mathcal{Q}_{i,j_i+1}(x_{k+\Theta},\mathscr{U}_0(x_{k+\Theta})),\tag{13.55}$$

where $\mathcal{Q}_i(x_k,\mathfrak{A}_k)=\mathcal{Q}_{i,N_i}(x_k,\mathfrak{A}_k)$. In the following, we will prove

$$\mathcal{J}_{i,j_i}(x_k,\mathfrak{A}_k)\leq\mathcal{Q}_{i,1}(x_k,\mathfrak{A}_k),\tag{13.56}$$

for $i=1,2,\ldots$ and $j_i=0,1,\ldots,N_i-1$, $N_i\in\mathcal{N}$. The statement can be proven by mathematical induction. For $i=1$ and $j_1=0$, we know

$\mathcal{Q}_{1,0}(x_k, \mathfrak{A}_k) = \mathcal{J}_{1,0}(x_k, \mathfrak{A}_k)$. For $j_1 = 1$, we have

$$\begin{aligned}\mathcal{J}_{1,1}(x_k, \mathfrak{A}_k) &= \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{J}_{1,0}(x_{k+\Theta}, \mathscr{U}_1(x_{k+\Theta})) \\ &= \Psi(x_k, \mathfrak{A}_k) + \vartheta \min_{\mathfrak{A}_{k+\Theta}} \mathcal{J}_{1,0}(x_{k+\Theta}, \mathfrak{A}_{k+\Theta}) \\ &\leq \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{Q}_{1,0}(x_{k+\Theta}, \mathscr{U}_0(x_{k+\Theta})) \\ &= \mathcal{Q}_{1,1}(x_{k+\Theta}, \mathfrak{A}_{k+\Theta}). \end{aligned} \tag{13.57}$$

If the inequality $\mathcal{J}_{1,j_1-1}(x_k, \mathfrak{A}_k) \leq \mathcal{Q}_{1,1}(x_k, \mathfrak{A}_k)$ holds, then considering j_1, we have

$$\begin{aligned}\mathcal{J}_{1,j_1}(x_k, \mathfrak{A}_k) &= \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{J}_{1,j_1-1}(x_{k+\Theta}, \mathscr{U}_1(x_{k+\Theta})) \\ &\leq \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{J}_{1,0}(x_{k+\Theta}, \mathscr{U}_1(x_{k+\Theta})) \\ &\leq \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{Q}_{1,0}(x_{k+\Theta}, \mathscr{U}_0(x_{k+\Theta})) \\ &= \mathcal{Q}_{1,1}(x_{k+\Theta}, \mathfrak{A}_{k+\Theta}), \end{aligned} \tag{13.58}$$

which shows that the inequality $\mathcal{J}_{1,j_1}(x_k, \mathfrak{A}_k) \leq \mathcal{Q}_{1,1}(x_k, \mathfrak{A}_k)$ holds for all $j_1 = 0, 1, \ldots, N_1 - 1$. Assume that the inequality (13.55) holds for $i = \varsigma - 1$, $\varsigma = 1, 2, \ldots$. For $i = \varsigma$ and $j_\varsigma = 1$, we know

$$\begin{aligned}\mathcal{J}_{\varsigma,1}(x_k, \mathfrak{A}_k) &= \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{J}_{\varsigma,0}(x_{k+\Theta}, \mathscr{U}_\varsigma(x_{k+\Theta})) \\ &= \Psi(x_k, \mathfrak{A}_k) + \vartheta \min_{\mathfrak{A}_{k+\Theta}} \mathcal{J}_{\varsigma,0}(x_{k+\Theta}, \mathfrak{A}_{k+\Theta}) \\ &\leq \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{Q}_{\varsigma,0}(x_{k+\Theta}, \mathscr{U}_0(x_{k+\Theta})) \\ &= \mathcal{Q}_{\varsigma,1}(x_{k+\Theta}, \mathfrak{A}_k). \end{aligned} \tag{13.59}$$

If the inequality $\mathcal{J}_{\varsigma,j_\varsigma-1}(x_k, \mathfrak{A}_k) \leq \mathcal{Q}_{\varsigma,j_\varsigma-1}(x_k, \mathfrak{A}_k)$, then for j_ς, we can get

$$\begin{aligned}\mathcal{J}_{\varsigma,j_\varsigma}(x_k, \mathfrak{A}_k) &= \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{J}_{\varsigma,j_\varsigma-1}(x_{k+\Theta}, \mathscr{U}_\varsigma(x_{k+\Theta})) \\ &\leq \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{Q}_{\varsigma,0}(x_{k+\Theta}, \mathscr{U}_0(x_{k+\Theta})) \\ &= \mathcal{Q}_{\varsigma,1}(x_k, \mathfrak{A}_k), \end{aligned} \tag{13.60}$$

which shows (13.56) for all $i = 1, 2, \ldots,$ and $j_i = 0, 1, \ldots, N_i$. Mathematical induction is complete.

On the other hand, for $i = 0, 1, \ldots,$ and $\{N_1, N_2, \ldots, N_i\}$, we can obtain

$$\begin{aligned}
&\mathcal{Q}_{i,N_i}(x_k, \mathscr{U}_0(x_k)) \\
&\quad = \sum_{\ell=0}^{i} \left(\prod_{\xi=1}^{\ell} \vartheta^{N_\xi} \left(\sum_{\tau_\ell=0}^{N_\ell} \vartheta^{\tau_\ell} \Psi \left(x_{k+\sum_{\lambda=0}^{\ell-1} N_\lambda \Theta + \tau_\ell \Theta}, \right. \right. \right. \\
&\qquad \left. \left. \left. \mathscr{U}_0(x_{k+\sum_{\lambda=0}^{\ell-1} N_\lambda \Theta + \tau_\ell \Theta}) \right) \right) \right) \\
&\qquad + \prod_{\xi=1}^{i} \vartheta^{N_\xi} J_0 \left(x_{k+\sum_{\lambda=0}^{i} N_\lambda \Theta + 1}, \mathscr{U}_0 \left(x_{k+\sum_{\lambda=0}^{i} N_\lambda \Theta + 1} \right) \right). \quad (13.61)
\end{aligned}$$

From (13.61), letting $i \to \infty$, we can derive that

$$\mathcal{Q}_\infty(x_k, \mathscr{U}_0(x_k)) = \sum_{\ell=0}^{\infty} (\vartheta^\ell \Psi(x_{k+\ell\Theta}, \mathscr{U}_0(x_{k+\ell\Theta}))), \quad (13.62)$$

as $\prod_{\xi=1}^{\infty} \vartheta^{N_\xi} = 0$ and $\mathcal{J}_0(x_k, \mathscr{U}_0(x_k))$ is finite for all x_k. According to (13.56), we can obtain

$$\lim_{i\to\infty} \mathcal{J}_i(x_k, \mathscr{U}_i(x_k)) \le \sum_{\ell=0}^{\infty} (\vartheta^\ell \Psi(x_{k+\ell\Theta}, \mathscr{U}_0(x_{k+\ell\Theta}))). \quad (13.63)$$

(4) *Show that the cost function* $\mathcal{J}_\infty(x_k, \mathfrak{A}_k)$ *equals the optimal value function* $\mathcal{J}^*(x_k, \mathfrak{A}_k)$.

According to the definition of $\mathcal{J}^*(x_k, \mathfrak{A}_k)$ in (13.6), for $i = 0, 1, \ldots$ and $j_i = 1, 2, \ldots, N_i$, we have $\mathcal{J}_{i,j_i}(x_k, \mathfrak{A}_k) \ge \mathcal{J}^*(x_k, \mathfrak{A}_k)$, which shows

$$\mathcal{J}_\infty(x_k, \mathfrak{A}_k) \ge \mathcal{J}^*(x_k, \mathfrak{A}_k). \quad (13.64)$$

for $i \to \infty$. Let $\underline{\mathfrak{A}}_k = (\mathfrak{A}_k, \mathfrak{A}_{k+\Theta}, \ldots)$. According to (13.63), as $\mathscr{U}_0(x_k)$ is an arbitrary control law sequence, we have

$$\begin{aligned}\lim_{i\to\infty} \mathcal{J}_i(x_k, \mathscr{U}_i(x_k)) &\leq \sum_{\ell=0}^{\infty} (\vartheta^\ell \Psi(x_{k+\ell\Theta}, \mathscr{U}_0(x_{k+\ell\Theta})))\\ &\leq \min_{\underline{\mathfrak{A}}_k} \left(\sum_{\ell=0}^{\infty} (\vartheta^\ell \Psi(x_{k+\ell\Theta}, \mathfrak{A}_{k+\ell\Theta})) \right)\\ &= \mathcal{J}^*(x_k, \mathscr{U}^*(x_k)), \end{aligned} \tag{13.65}$$

which shows

$$\begin{aligned}\mathcal{J}_\infty(x_k, \mathfrak{A}_k) &= \Psi(x_k, \mathfrak{A}_k) + \vartheta \mathcal{J}_\infty(x_{k+\Theta}, \mathscr{U}_\infty(x_{k+\Theta}))\\ &\leq \Psi(x_k, \mathfrak{A}_k) + \mathcal{J}^*(x_{k+\Theta}, \mathscr{U}^*(x_{k+\Theta}))\\ &= \mathcal{J}^*(x_k, \mathfrak{A}_k). \end{aligned} \tag{13.66}$$

Combing (13.64) and (13.66), we can obtain the conclusion (13.42). The proof is complete. □

13.4 Simulation Example

This section aims to illustrate the performance of the generalized actor-critic learning optimal control method by a numerical example. The real-time load demand and electricity rate for 168 hours (one week) are shown in Figs. 13.2(a) and (c), respectively. It is shown that the load demand and the electricity rate are both periodic-like functions with the period $\Theta = 24$, where the averages of the load demand and the electricity rate, which are shown in Figs. 13.2(b) and (d), respectively, are used as the periodic functions.

The parameters of the energy storage equipment are given as follows. Set the capacity of the energy storage equipment as 100 kWh, which means $B^o = 50$ kWh. The rated power of the equipment is set as 16.5 kW. The initial energy of the equipment is set as 60 kWh. Define the value function as in (13.1), where we set $\alpha_1 = 1$, $\alpha_2 = 0.5$ and $\alpha_3 = 0.2$, respectively. The initial state is set as $x_0 = [8, 60]^\mathsf{T}$.

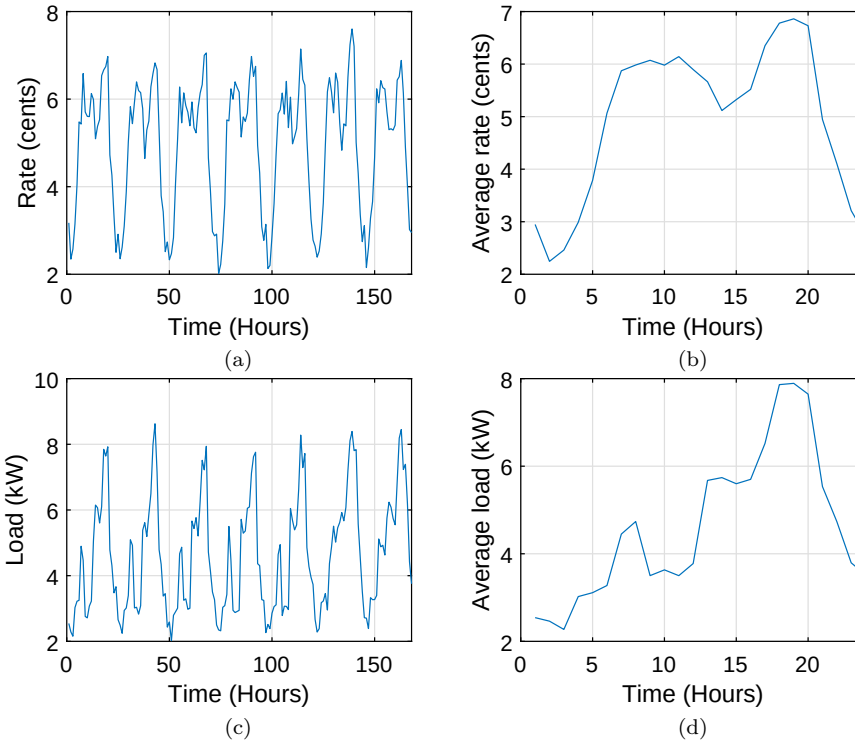

Fig. 13.2. Residential load demand and electricity rate. (a) Residential load demand for 168 hours. (b) Average residential load demand. (c) Real-time electricity rate for 168 hours. (d) Average electricity rate.

An initial control law $\mathscr{U}_0(x_k)$ is necessary to implement the generalized actor-critic learning optimal control method. Generally, the initial control law can be designed by experience. Let the equipment charge when load demand and the electricity rate are both obviously low, such as the midnight time. Let the equipment discharge when load demand and the electricity rate are both obviously high, such as the dinner time. Other time the charging/discharging type of the equipment is chosen arbitrarily.

We implement the developed generalized actor-critic learning optimal control method for $i = 20$ global iterations to show the property of the iterative value function $\mathcal{J}_i(x_k)$, where in each local

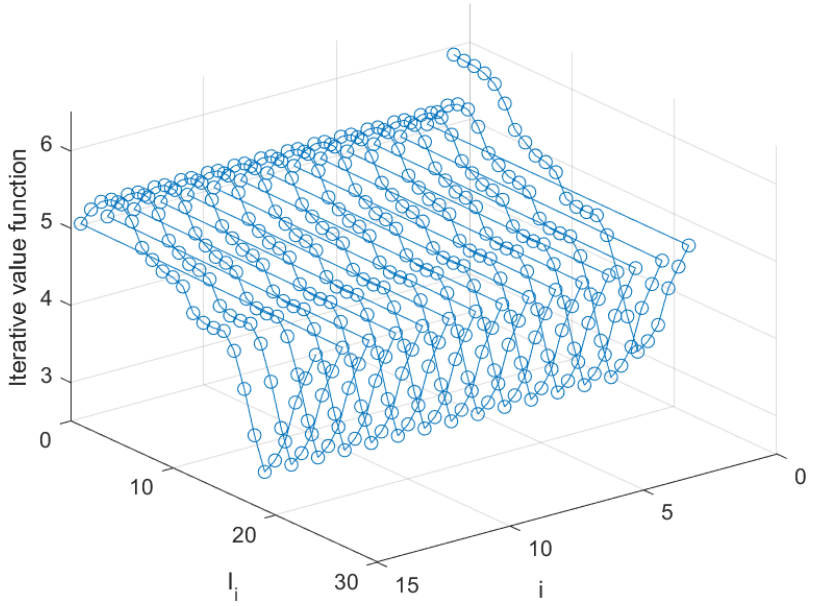

Fig. 13.3. The plot of the iterative value function.

iteration for $i = 1, 2, \ldots,$ we design $\mathcal{N}_i$ as an arbitrary positive integer for local iterations $j = 0, 1, \ldots, N_i$. The plot of the iterative value function $\mathcal{J}_i(x_k, \mathscr{U}_i(x_k))$ is shown in Fig. 13.3.

From Fig. 13.3, it can be seen that at each hour in a day, the iterative value function is monotonically non-increasing and finally converges to the optimum, which verifies the correctness of the theoretical results. The optimal control for the smart home system is displayed in Fig. 13.4. We can see that the equipment is charging when the load demand and the electricity rate are low and the equipment is discharging when load demand and the electricity rate are high.

The optimal energy management for the load demand is shown in Fig. 13.5. It is shown that when load demand and the electricity rate are high, the power grid nearly stops supplying the power to the load and all the power is supplied by the energy storage equipment. When the load demand and the electricity rate are low, the power grid bears all the power from the grid to the load. The corresponding energy in the energy storage equipment is shown in Fig. 13.6.

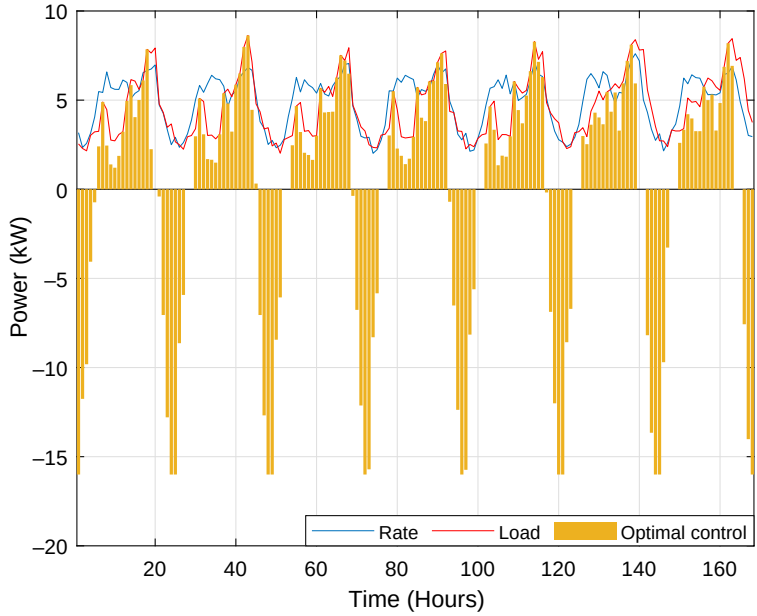

Fig. 13.4. Optimal control of the battery in one week.

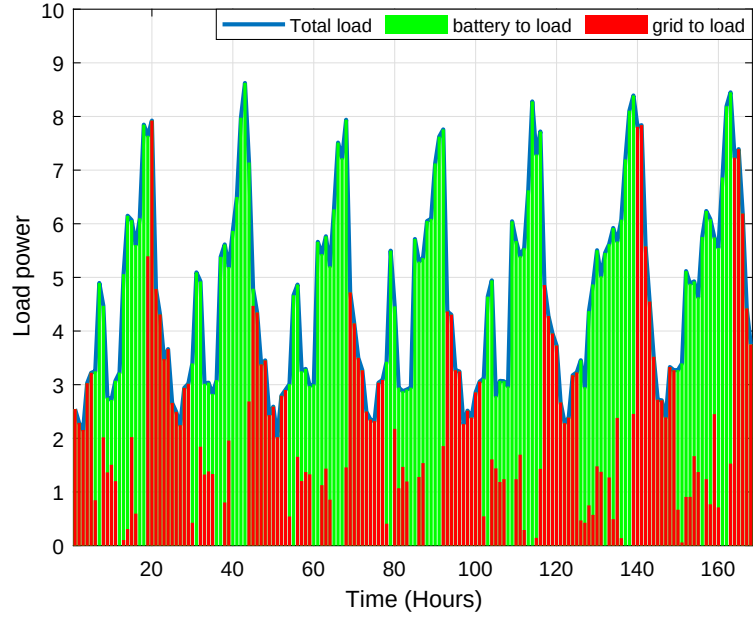

Fig. 13.5. The optimal energy supply for the load demand.

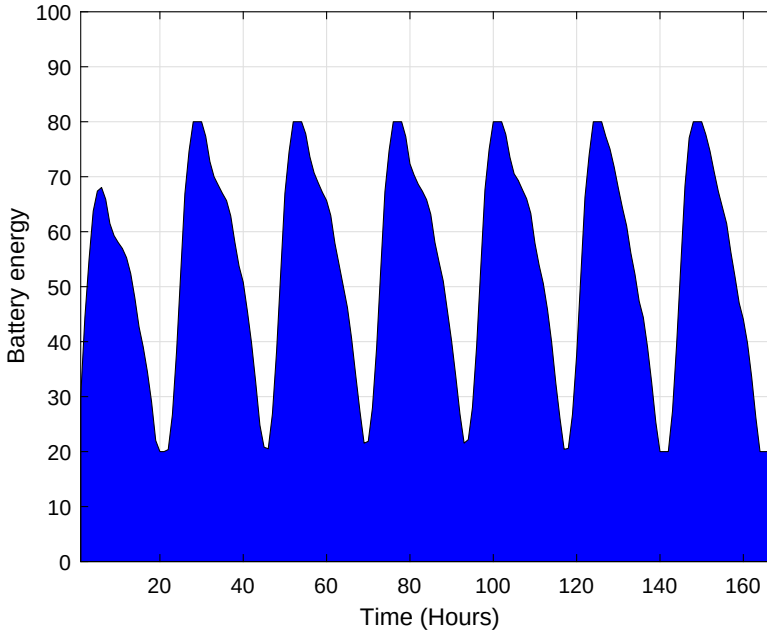

Fig. 13.6. Optimal energy storage in one week.

13.5 Conclusion

In this chapter, a novel generalized actor-critic learning optimal control method is presented. The idea of the developed method is to find the optimal energy control law for smart home systems by ADP. Combining the value and policy iteration ADP techniques, the generalized actor-critic learning optimal control method presents a common iteration structure to for smart home energy management. Three iteration processes in the generalized actor-critic learning optimal control method, which are global iteration, local iteration and interior iteration, have been shown. It has also been proven that the iterative cost function is monotonically non-increasing and converges to the optimal value function. Finally, numerical results are displayed to show the superiority of the developed algorithm.

References

[1] Wei, Q., Liu, D. and Shi, G. (2015). A novel dual iterative Q-learning method for optimal battery management in smart residential environments. *IEEE Transactions on Industrial Electronics* 62(4), 2509–2518.

[2] Wei, Q., Liu, D., Lewis, F.L. and Liu, Y. (2017). Mixed iterative adaptive dynamic programming for optimal battery energy control in smart residential microgrids. *IEEE Transactions on Industrial Electronics* 64(5), 4110–4120.

This page intentionally left blank

Index