

Derong Liu
Qinglai Wei
Ding Wang
Xiong Yang
Hongliang Li

Adaptive Dynamic Programming with Applications in Optimal Control

Advances in Industrial Control

Series editors

Michael J. Grimble, Glasgow, UK

Michael A. Johnson, Kidlington, UK

More information about this series at <http://www.springer.com/series/1412>

Derong Liu · Qinglai Wei · Ding Wang
Xiong Yang · Hongliang Li

Adaptive Dynamic Programming with Applications in Optimal Control



Springer

This PDF document was edited with **Icecream PDF Editor**.
[Upgrade to PRO](#) to remove watermark.

Derong Liu
Institute of Automation
Chinese Academy of Sciences
Beijing
China

Qinglai Wei
Institute of Automation
Chinese Academy of Sciences
Beijing
China

Ding Wang
Institute of Automation
Chinese Academy of Sciences
Beijing
China

Xiong Yang
Tianjin University
Tianjin
China

Hongliang Li
Tencent Inc.
Shenzhen
China

ISSN 1430-9491
Advances in Industrial Control
ISBN 978-3-319-50813-9
DOI 10.1007/978-3-319-50815-3

ISSN 2193-1577 (electronic)
ISBN 978-3-319-50815-3 (eBook)

Library of Congress Control Number: 2016959539

© Springer International Publishing AG 2017

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

Nowadays, nonlinearity is involved in all walks of life. It is a challenge for engineers to design controllers for all kinds of nonlinear systems. To handle this issue, various nonlinear control theories have been developed, such as theories of adaptive control, optimal control, and robust control. Among these theories, the theory of optimal control has drawn considerable attention over the past several decades. This is mainly because optimal control provides an effective way to design controllers with guaranteed robustness properties as well as capabilities of optimization and resource conservation that are important in manufacturing, vehicle emission control, aerospace systems, power systems, chemical engineering processes, and many other applications.

The core challenge in deriving the solutions of nonlinear optimal control problems is that it often boils down to solving certain Hamilton–Jacobi–Bellman (HJB) equations. The HJB equations are nonlinear and difficult to solve for general nonlinear dynamical systems. Indeed, no closed-form solution to such equations exists, except for very special problems. Therefore, numerical solutions to HJB equations have been developed by engineers. To obtain such numerical solutions, a highly effective method known as adaptive/approximate dynamic programming (ADP) can be used. A distinct advantage of ADP is that it can avoid the well-known “curse of dimensionality” of dynamic programming while adaptively solving the HJB equations. Due to this characteristic, many elegant ADP approaches and their applications have been developed in the literature during the past several decades. It is also notable that ADP techniques also provide a link with cognitive decision-making methods that are observed in the human brain, and thus, ADP has become a main channel to achieve truly brain-like intelligence in human-engineered automatic control systems.

Unlike most ADP books, the present book “Adaptive Dynamic Programming with Applications in Optimal Control” focuses on the principles of emerging optimal control techniques for nonlinear systems in both discrete-time and continuous-time domains, and on creating applications of these optimal control techniques. This book contains three themes:

1. Optimal control for discrete-time nonlinear dynamical systems, covering various novel techniques used to derive optimal control in the discrete-time domain, such as general value iteration, θ -ADP, finite approximation error-based value iteration, policy iteration, generalized policy iteration, and error bounds analysis of ADP.
2. Optimal control for continuous-time nonlinear systems, discussing the optimal control for input-affine/input-nonaffine nonlinear systems, robust and optimal guaranteed cost control for input-affine nonlinear systems, decentralized control for interconnected nonlinear systems, and optimal control for differential games.
3. Applications, providing typical applications of optimal control approaches in the areas of energy management in smart homes, coal gasification, and water gas shift reaction.

This book provides timely and informative coverage about ADP, including both rigorous derivations and insightful developments. It will help both specialists and nonspecialists understand the new developments in the field of nonlinear optimal control using online/offline learning techniques. Meanwhile, it will be beneficial for engineers to apply the developed ADP methods to their own problems in practice. I am sure you will enjoy reading this book.

Arlington, TX, USA
September 2016

Frank L. Lewis

Series Editors' Foreword

The series *Advances in Industrial Control* aims to report and encourage technology transfer in control engineering. The rapid development of control technology has an impact on all areas of the control discipline: new theory, new controllers, actuators, sensors, new industrial processes, computer methods, new applications, new design philosophies, and new challenges. Much of this development work resides in industrial reports, feasibility study papers, and the reports of advanced collaborative projects. The series offers an opportunity for researchers to present an extended exposition of such new work in all aspects of industrial control for wider and rapid dissemination.

The method of dynamic programming has a long history in the field of optimal control. It dates back to those days when the subject of control was emerging in a modern form in the 1950s and 1960s. It was devised by Richard Bellman who gave it a modern revision in a publication of 1954 [1]. The name of Bellman became linked to an optimality equation, key to the method, and like the name of Kalman became uniquely associated with the early development of optimal control. One notable extension to the method was that of differential dynamic programming due to David Q. Mayne in 1966 and developed at length in the book by Jacobson and Mayne [2]. Their new technique used locally quadratic models for the system dynamics and cost functions and improved the convergence of the dynamic programming method for optimal trajectory control problems.

Since those early days, the subject of control has taken many different directions, but dynamic programming has always retained a place in the theory of optimal control fundamentals. It is therefore instructive for the *Advances in Industrial Control* monograph series to have a contribution that presents new ways of solving dynamic programming and demonstrating these methods with some up-to-date industrial problems. This monograph, *Adaptive Dynamic Programming with Applications in Optimal Control*, by Derong Liu, Qinglai Wei, Ding Wang, Xiong Yang and Hongliang Li, has precisely that objective.

The authors open the monograph with a very interesting and relevant discussion of another computationally difficult problem, namely devising a computer program to defeat human master players at the Chinese game of Go. Inspiration from the

better programming techniques used in the Go-master problem was used by the authors to defeat the “curse of dimensionality” that arises in dynamic programming methods.

More formally, the objective of the techniques reported in the monograph is to control in an optimal fashion an unknown or uncertain nonlinear multivariable system using recorded and instantaneous output signals. The algorithms’ technical framework is then constructed through different categories of the usual state-space nonlinear ordinary differential system model. The system model can be continuous or discrete, have affine or nonaffine control inputs, be subject to no constraints, or have constraints present. A set of 11 chapters contains the theory for various formulations of the system features.

Since standard dynamic programming schemes suffer from various implementation obstacles, adaptive dynamic programming procedures have been developed to find computable practical suboptimal control solutions. A key technique used by the authors is that of neural networks which are trained using recorded data and updated, or “adapted,” to accommodate uncertain system knowledge. The theory chapters are arranged in two parts: Part 1 *Discrete-Time Systems*—five chapters; and Part 2 *Continuous-Time Systems*—five chapters.

An important feature of the monographs of the *Advances in Industrial Control* series is a demonstration of potential or actual application to industrial problems. After a comprehensive presentation of the theory of adaptive dynamic programming, the authors devote Part 3 of their monograph to three chapter-length application studies. Chapter 12 examines the scheduling of energy supplies in a smart home environment, a topic and problem of considerable contemporary interest. Chapter 13 uses a coal gasification process that is suitably challenging to demonstrate the authors’ techniques. And finally, Chapter 14 concerns the control of the water gas shift reaction. In this example, the data used was taken from a real-world operational system.

This monograph is very comprehensive in its presentation of the adaptive dynamic programming theory and has demonstrations with three challenging processes. It should find a wide readership in both the industrial control engineering and the academic control theory communities. Readers in other fields such as computer science and chemical engineering may also find the monograph of considerable interest.

Michael J. Grimble
Michael A. Johnson
Industrial Control Centre
University of Strathclyde
Glasgow, Scotland, UK

References

1. Bellman R (1954) The theory of dynamic programming. *Bulletin of the American Mathematical Society* 60(6):503–515
2. Jacobson DH, Mayne DQ (1970) Differential dynamic programming, American Elsevier Pub. Co. New York

Preface

With the rapid development in information science and technology, many businesses and industries have undergone great changes, such as chemical industry, electric power engineering, electronics industry, mechanical engineering, transportation, and logistics business. While the scale of industrial enterprises is increasing, production equipment and industrial processes are becoming more and more complex. For these complex systems, decision and control are necessary to ensure that they perform properly and meet prescribed performance objectives. Under this circumstance, how to design safe, reliable, and efficient control for complex systems is essential for our society. As modern systems become more complex and performance requirements become more stringent, advanced control methods are greatly needed to achieve guaranteed performance and satisfactory goals.

In general, optimal control deals with the problem of finding a control law for a given system such that a certain optimality criterion is achieved. The main difference between optimal control of linear and nonlinear systems lies in that the latter often requires solving the nonlinear Bellman equation instead of the Riccati equation. Although dynamic programming is a conventional method in solving optimization and optimal control problems, it often suffers from the “curse of dimensionality.” To overcome this difficulty, based on function approximators such as neural networks, adaptive/approximate dynamic programming (ADP) was proposed by Werbos as a method for solving optimal control problems forward-in-time.

This book presents the recent results of ADP with applications in optimal control. It is composed of 14 chapters which cover most of the hot research areas of ADP and are divided into three parts. Part I concerns discrete-time systems, including five chapters from Chaps. 2 to 6. Part II concerns continuous-time systems, including five chapters from Chaps. 7 to 11. Part III concerns applications, including three chapters from Chaps. 12 to 14.

In Chap. 1, an introduction to the history of ADP is provided, including the basic and iterative forms of ADP. The review begins with the origin of ADP and

describes the basic structures and the algorithm development in detail. Connections between ADP and reinforcement learning are also discussed.

Part I: Discrete-Time Systems (Chaps. 2–6)

In Chap. 2, optimal control problems of discrete-time nonlinear dynamical systems, including optimal regulation, optimal tracking control, and constrained optimal control, are studied using a series of value iteration ADP approaches. First, an ADP scheme based on general value iteration is developed to obtain near-optimal control for discrete-time affine nonlinear systems with continuous state and control spaces. The present scheme is also employed to solve infinite-horizon optimal tracking control problems for a class of discrete-time nonlinear systems. In particular, using the globalized dual heuristic programming technique, a value iteration-based optimal control strategy of unknown discrete-time nonlinear dynamical systems with input constraints is established as a case study. Second, an iterative θ -ADP algorithm is given to solve the optimal control problem of infinite-horizon discrete-time nonlinear systems, which shows that each of the iterative controls can stabilize the nonlinear dynamical systems and the condition of initial admissible control is avoided effectively.

In Chap. 3, a series of iterative ADP algorithms are developed to solve the infinite-horizon optimal control problems for discrete-time nonlinear dynamical systems with finite approximation errors. Iterative control laws are obtained by using the present algorithms such that the iterative value functions reach the optimum. Then, the numerical optimal control problems are solved by a novel numerical adaptive learning control scheme based on ADP algorithm. Moreover, a general value iteration algorithm with finite approximate errors is developed to guarantee the iterative value function to converge to the solution of the Bellman equation. The general value iteration algorithm permits an arbitrary positive semidefinite function to initialize itself, which overcomes the disadvantage of traditional value iteration algorithms.

In Chap. 4, a discrete-time policy iteration ADP method is developed to solve the infinite-horizon optimal control problems for nonlinear dynamical systems. The idea is to use an iterative ADP technique to obtain iterative control laws that optimize the iterative value functions. The convergence, stability, and optimality properties are analyzed for policy iteration method for discrete-time nonlinear dynamical systems, and it is shown that the iterative value functions are nonincreasingly convergent to the optimal solution of the Bellman equation. It is also proven that any of the iterative control laws obtained from the present policy iteration algorithm can stabilize the nonlinear dynamical systems.

In Chap. 5, a generalized policy iteration algorithm is developed to solve the optimal control problems for infinite-horizon discrete-time nonlinear systems. Generalized policy iteration algorithm uses the idea of interacting the policy iteration algorithm and the value iteration algorithm of ADP. It permits an arbitrary positive semidefinite function to initialize the algorithm, where two iteration indices are used for policy evaluation and policy improvement, respectively. The

monotonicity, convergence, admissibility, and optimality properties of the generalized policy iteration algorithm are analyzed.

In Chap. 6, error bounds of ADP algorithms are established for solving undiscounted infinite-horizon optimal control problems of discrete-time deterministic nonlinear systems. The error bounds for approximate value iteration based on a novel error condition are developed. The error bounds for approximate policy iteration and approximate optimistic policy iteration algorithms are also provided. It is shown that the iterative approximate value function can converge to a finite neighborhood of the optimal value function under some conditions. In addition, error bounds are also established for Q-function of approximate policy iteration for optimal control of unknown discounted discrete-time nonlinear systems. Neural networks are used to approximate the Q-function and the control policy.

Part II: Continuous-Time Systems (Chaps. 7–11)

In Chap. 7, optimal control problems of continuous-time affine nonlinear dynamical systems are studied using ADP approaches. First, an identifier–critic architecture based on ADP methods is presented to derive the approximate optimal control for uncertain continuous-time nonlinear dynamical systems. The identifier neural network and the critic neural network are tuned simultaneously, while the restrictive persistence of excitation condition is relaxed. Second, an ADP-based algorithm is developed to solve the optimal control problems for continuous-time nonlinear dynamical systems with control constraints. Only a single critic neural network is utilized to derive the optimal control, and there is no special requirement on the initial control.

In Chap. 8, the optimal control problems are considered for continuous-time nonaffine nonlinear dynamical systems with completely unknown dynamics via ADP methods. First, an ADP-based novel identifier–actor–critic architecture is developed to provide approximate optimal control solutions for continuous-time unknown nonaffine nonlinear dynamical systems, where the identifier is constructed by a dynamic neural network to transform nonaffine nonlinear systems into a class of affine nonlinear systems. Second, an ADP-based observer–critic architecture is presented to obtain the approximate optimal control for nonaffine nonlinear dynamical systems in the presence of unknown dynamics, where the observer is composed of a three-layer feedforward neural network aiming to get the knowledge of system states.

In Chap. 9, robust control and optimal guaranteed cost control of continuous-time uncertain nonlinear systems are studied using the idea of ADP. First, a novel strategy is established to design the robust controller for a class of nonlinear systems with uncertainties based on an online policy iteration algorithm. By properly choosing a cost function that reflects the uncertainties, regulation, and control, the robust control problem is transformed into an optimal control problem, which can be solved effectively under the framework of ADP. Then, the infinite-horizon optimal guaranteed cost control problem of uncertain nonlinear systems is investigated by employing the formulation of ADP-based online optimal

control design, which extends the application scope of ADP methods to nonlinear and uncertain environment.

In Chap. 10, by using neural network-based online learning optimal control approach, a decentralized control strategy is developed to stabilize a class of continuous-time large-scale interconnected nonlinear systems. The decentralized control strategy of the overall system can be established by adding appropriate feedback gains to the optimal control laws of isolated subsystems. Then, an online policy iteration algorithm is presented to solve the Hamilton–Jacobi–Bellman equations related to the optimal control problems. Furthermore, as a generalization, a neural network-based decentralized control law is developed to stabilize the large-scale interconnected nonlinear systems with unknown dynamics by using an online model-free integral policy iteration algorithm.

In Chap. 11, differential game problems of continuous-time systems, including two-player zero-sum games, multiplayer zero-sum games, and multiplayer nonzero-sum games, are studied via a series of ADP approaches. First, an integral policy iteration algorithm is developed to learn online the Nash equilibrium solution of two-player zero-sum differential games with completely unknown continuous-time linear dynamics. Second, multiplayer zero-sum differential games for a class of continuous-time uncertain nonlinear systems are solved by using an iterative ADP algorithm. Finally, an online synchronous approximate optimal learning algorithm based on policy iteration is developed to solve multiplayer nonzero-sum games of continuous-time nonlinear systems without requiring exact knowledge of system dynamics.

Part III: Applications (Chaps. 12–14)

In Chap. 12, intelligent optimization methods based on ADP are applied to the challenges of intelligent price-responsive management of residential energy, with an emphasis on home battery use connected to the power grid. First, an action-dependent heuristic dynamic programming is developed to obtain the optimal control law for residential energy management. Second, a dual iterative Q-learning algorithm is developed to solve the optimal battery management and control problem in smart residential environments where two iterations are introduced, which are respectively internal and external iterations. 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. Finally, a distributed iterative ADP method is developed to solve the multibattery optimal coordination control problem for home energy management systems.

In Chap. 13, a coal gasification optimal tracking control problem is solved through a data-based iterative optimal learning control scheme by using iterative ADP approach. According to system data, neural networks are used to construct the dynamics of coal gasification process, coal quality, and reference control, respectively. Via system transformation, the optimal tracking control problem with approximation errors and disturbances is effectively transformed into a two-person zero-sum optimal control problem. An iterative ADP algorithm is developed to obtain the optimal control laws for the transformed system.

In Chap. 14, a data-driven stable iterative ADP algorithm is developed to solve the optimal temperature control problem of water gas shift reaction system. According to the system data, neural networks are used to construct the dynamics of water gas shift reaction system and solve the reference control. Considering the reconstruction errors of neural networks and the disturbances of the system and control input, a stable iterative ADP algorithm is developed to obtain the optimal control law. Convergence property is developed to guarantee that the iterative value function converges to a finite neighborhood of the optimal cost function. Stability property is developed so that each of the iterative control laws can guarantee the tracking error to be uniformly ultimately bounded.

Beijing, China
Chicago, USA
September 2016

Derong Liu
Qinglai Wei
Ding Wang
Xiong Yang
Hongliang Li

Acknowledgements

The authors would like to acknowledge the help and encouragement they have received from colleagues in Beijing and Chicago during the course of writing this book. Some materials presented in this book are based on the research conducted with several Ph.D. students, including Yuzhu Huang, Dehua Zhang, Pengfei Yan, Yancai Xu, Hongwen Ma, Chao Li, and Guang Shi. The authors also wish to thank Oliver Jackson, Editor (Engineering) from Springer for his patience and encouragements.

The authors are very grateful to the National Natural Science Foundation of China (NSFC) for providing necessary financial support to our research in the past five years. The present book is the result of NSFC Grants 61034002, 61233001, 61273140, 61304086, and 61374105.

Contents

1	Overview of Adaptive Dynamic Programming	1
1.1	Introduction	1
1.2	Reinforcement Learning	3
1.3	Adaptive Dynamic Programming	7
1.3.1	Basic Forms of Adaptive Dynamic Programming	10
1.3.2	Iterative Adaptive Dynamic Programming	15
1.3.3	ADP for Continuous-Time Systems	18
1.3.4	Remarks	21
1.4	Related Books	22
1.5	About This Book	26
	References	27
 Part I Discrete-Time Systems		
2	Value Iteration ADP for Discrete-Time Nonlinear Systems	37
2.1	Introduction	37
2.2	Optimal Control of Nonlinear Systems	
Using General Value Iteration		38
2.2.1	Convergence Analysis	40
2.2.2	Neural Network Implementation	48
2.2.3	Generalization to Optimal Tracking Control	52
2.2.4	Optimal Control of Systems with Constrained Inputs	56
2.2.5	Simulation Studies	59
2.3	Iterative θ -Adaptive Dynamic Programming Algorithm for Nonlinear Systems	67
2.3.1	Convergence Analysis	69
2.3.2	Optimality Analysis	77
2.3.3	Summary of Iterative θ -ADP Algorithm	80
2.3.4	Simulation Studies	83

2.4	Conclusions	87
	References	87
3	Finite Approximation Error-Based Value Iteration ADP	91
3.1	Introduction	91
3.2	Iterative θ -ADP Algorithm with Finite Approximation Errors	92
3.2.1	Properties of the Iterative ADP Algorithm with Finite Approximation Errors	93
3.2.2	Neural Network Implementation	100
3.2.3	Simulation Study	104
3.3	Numerical Iterative θ -Adaptive Dynamic Programming	107
3.3.1	Derivation of the Numerical Iterative θ -ADP Algorithm	107
3.3.2	Properties of the Numerical Iterative θ -ADP Algorithm	111
3.3.3	Summary of the Numerical Iterative θ -ADP Algorithm	120
3.3.4	Simulation Study	121
3.4	General Value Iteration ADP Algorithm with Finite Approximation Errors	125
3.4.1	Derivation and Properties of the GVI Algorithm with Finite Approximation Errors	125
3.4.2	Designs of Convergence Criteria with Finite Approximation Errors	133
3.4.3	Simulation Study	140
3.5	Conclusions	147
	References	147
4	Policy Iteration for Optimal Control of Discrete-Time Nonlinear Systems	151
4.1	Introduction	151
4.2	Policy Iteration Algorithm	152
4.2.1	Derivation of Policy Iteration Algorithm	153
4.2.2	Properties of Policy Iteration Algorithm	154
4.2.3	Initial Admissible Control Law	160
4.2.4	Summary of Policy Iteration ADP Algorithm	162
4.3	Numerical Simulation and Analysis	162
4.4	Conclusions	173
	References	174

5 Generalized Policy Iteration ADP for Discrete-Time Nonlinear Systems	177
5.1 Introduction	177
5.2 Generalized Policy Iteration-Based Adaptive Dynamic Programming Algorithm	177
5.2.1 Derivation and Properties of the GPI Algorithm	179
5.2.2 GPI Algorithm and Relaxation of Initial Conditions	188
5.2.3 Simulation Studies.	192
5.3 Discrete-Time GPI with General Initial Value Functions	199
5.3.1 Derivation and Properties of the GPI Algorithm	199
5.3.2 Relaxations of the Convergence Criterion and Summary of the GPI Algorithm	211
5.3.3 Simulation Studies.	215
5.4 Conclusions	221
References.	221
6 Error Bounds of Adaptive Dynamic Programming Algorithms	223
6.1 Introduction	223
6.2 Error Bounds of ADP Algorithms for Undiscounted Optimal Control Problems	224
6.2.1 Problem Formulation.	224
6.2.2 Approximate Value Iteration	226
6.2.3 Approximate Policy Iteration.	231
6.2.4 Approximate Optimistic Policy Iteration	237
6.2.5 Neural Network Implementation	241
6.2.6 Simulation Study	243
6.3 Error Bounds of Q-Function for Discounted Optimal Control Problems.	247
6.3.1 Problem Formulation	247
6.3.2 Policy Iteration Under Ideal Conditions.	249
6.3.3 Error Bound for Approximate Policy Iteration.	254
6.3.4 Neural Network Implementation	257
6.3.5 Simulation Study	259
6.4 Conclusions	262
References.	263

Part II Continuous-Time Systems

7 Online Optimal Control of Continuous-Time Affine Nonlinear Systems	267
7.1 Introduction	267
7.2 Online Optimal Control of Partially Unknown Affine Nonlinear Systems	267
7.2.1 Identifier–Critic Architecture for Solving HJB Equation	269

7.2.2	Stability Analysis of Closed-Loop System	281
7.2.3	Simulation Study	286
7.3	Online Optimal Control of Affine Nonlinear Systems with Constrained Inputs	291
7.3.1	Solving HJB Equation via Critic Architecture	294
7.3.2	Stability Analysis of Closed-Loop System with Constrained Inputs	298
7.3.3	Simulation Study	302
7.4	Conclusions	305
	References	306
8	Optimal Control of Unknown Continuous-Time Nonaffine Nonlinear Systems	309
8.1	Introduction	309
8.2	Optimal Control of Unknown Nonaffine Nonlinear Systems with Constrained Inputs	310
8.2.1	Identifier Design via Dynamic Neural Networks	311
8.2.2	Actor–Critic Architecture for Solving HJB Equation	316
8.2.3	Stability Analysis of Closed-Loop System	318
8.2.4	Simulation Study	323
8.3	Optimal Output Regulation of Unknown Nonaffine Nonlinear Systems	327
8.3.1	Neural Network Observer	328
8.3.2	Observer-Based Optimal Control Scheme Using Critic Network	333
8.3.3	Stability Analysis of Closed-Loop System	337
8.3.4	Simulation Study	340
8.4	Conclusions	343
	References	343
9	Robust and Optimal Guaranteed Cost Control of Continuous-Time Nonlinear Systems	345
9.1	Introduction	345
9.2	Robust Control of Uncertain Nonlinear Systems	346
9.2.1	Equivalence Analysis and Problem Transformation	348
9.2.2	Online Algorithm and Neural Network Implementation	350
9.2.3	Stability Analysis of Closed-Loop System	353
9.2.4	Simulation Study	356
9.3	Optimal Guaranteed Cost Control of Uncertain Nonlinear Systems	360
9.3.1	Optimal Guaranteed Cost Controller Design	362
9.3.2	Online Solution of Transformed Optimal Control Problem	368

9.3.3	Stability Analysis of Closed-Loop System	373
9.3.4	Simulation Studies	378
9.4	Conclusions	383
	References	384
10	Decentralized Control of Continuous-Time Interconnected Nonlinear Systems	387
10.1	Introduction	387
10.2	Decentralized Control of Interconnected Nonlinear Systems	388
10.2.1	Decentralized Stabilization via Optimal Control Approach	389
10.2.2	Optimal Controller Design of Isolated Subsystems	394
10.2.3	Generalization to Model-Free Decentralized Control	400
10.2.4	Simulation Studies	404
10.3	Conclusions	414
	References	414
11	Learning Algorithms for Differential Games of Continuous-Time Systems	417
11.1	Introduction	417
11.2	Integral Policy Iteration for Two-Player Zero-Sum Games	418
11.2.1	Derivation of Integral Policy Iteration	420
11.2.2	Convergence Analysis	423
11.2.3	Neural Network Implementation	425
11.2.4	Simulation Studies	428
11.3	Iterative Adaptive Dynamic Programming for Multi-player Zero-Sum Games	431
11.3.1	Derivation of the Iterative ADP Algorithm	433
11.3.2	Properties	438
11.3.3	Neural Network Implementation	444
11.3.4	Simulation Studies	451
11.4	Synchronous Approximate Optimal Learning for Multi-player Nonzero-Sum Games	459
11.4.1	Derivation and Convergence Analysis	460
11.4.2	Neural Network Implementation	464
11.4.3	Simulation Study	473
11.5	Conclusions	478
	References	478

Part III Applications

12 Adaptive Dynamic Programming for Optimal Residential Energy Management	483
12.1 Introduction	483
12.2 A Self-learning Scheme for Residential Energy System Control and Management	484
12.2.1 The ADHDP Method	488
12.2.2 A Self-learning Scheme for Residential Energy System	489
12.2.3 Simulation Study	492
12.3 A Novel Dual Iterative Q-Learning Method for Optimal Battery Management	496
12.3.1 Problem Formulation	496
12.3.2 Dual Iterative Q-Learning Algorithm	497
12.3.3 Neural Network Implementation	503
12.3.4 Numerical Analysis	506
12.4 Multi-battery Optimal Coordination Control for Residential Energy Systems	513
12.4.1 Distributed Iterative ADP Algorithm	515
12.4.2 Numerical Analysis	527
12.5 Conclusions	533
References	533
13 Adaptive Dynamic Programming for Optimal Control of Coal Gasification Process	537
13.1 Introduction	537
13.2 Data-Based Modeling and Properties	538
13.2.1 Description of Coal Gasification Process and Control Systems	538
13.2.2 Data-Based Process Modeling and Properties	540
13.3 Design and Implementation of Optimal Tracking Control	546
13.3.1 Optimal Tracking Controller Design by Iterative ADP Algorithm Under System and Iteration Errors	546
13.3.2 Neural Network Implementation	554
13.4 Numerical Analysis	557
13.5 Conclusions	568
References	569
14 Data-Based Neuro-Optimal Temperature Control of Water Gas Shift Reaction	571
14.1 Introduction	571
14.2 System Description and Data-Based Modeling	572
14.2.1 Water Gas Shift Reaction	572
14.2.2 Data-Based Modeling and Properties	573

14.3	Design of Neuro-Optimal Temperature Controller	575
14.3.1	System Transformation	575
14.3.2	Derivation of Stable Iterative ADP Algorithm.	576
14.3.3	Properties of Stable Iterative ADP Algorithm with Approximation Errors and Disturbances	578
14.4	Neural Network Implementation for the Optimal Tracking Control Scheme	582
14.5	Numerical Analysis.	585
14.6	Conclusions	589
	References.	589
Index	591

Abbreviations

ACD	Adaptive critic designs
AD	Action-dependent, e.g., ADHDP and ADDHP
ADP	Adaptive dynamic programming, or approximate dynamic programming
ADPRL	Adaptive dynamic programming and reinforcement learning
BP	Backpropagation
DHP	Dual heuristic programming
DP	Dynamic programming
GDHP	Globalized dual heuristic programming
GPI	Generalized policy iteration
GVI	General value iteration
HDP	Heuristic dynamic programming
HJB	Hamilton–Jacobi–Bellman, e.g., HJB equation
HJI	Hamilton–Jacobi–Isaacs, e.g., HJI equation
NN	Neural network
PE	Persistence of excitation
PI	Policy iteration
UUB	Uniformly ultimately bounded
VI	Value iteration
RL	Reinforcement learning

Symbols

T	The transposition symbol, e.g., A^T is the transposition of matrix A
N	The set of all natural numbers
\mathbb{Z}^+	The set of all positive integers, i.e., $\mathbb{N} = \{0, \mathbb{Z}^+\}$
\mathbb{R}	The set of all real numbers
\mathbb{R}^n	The Euclidean space of all real n -vectors, e.g., a vector $x \in \mathbb{R}^n$ is written as $x = (x_1, x_2, \dots, x_n)^T$
$\mathbb{R}^{m \times n}$	The space of all m by n real matrices, e.g., a matrix $A \in \mathbb{R}^{m \times n}$ is written as $A = (a_{ij}) \in \mathbb{R}^{m \times n}$
$\ \cdot\ $	The vector norm or matrix norm in \mathbb{R}^n or $\mathbb{R}^{n \times m}$
$\ \cdot\ _F$	The Frobenius matrix norm, which is the Euclidean norm of a matrix, is defined as $\ A\ _F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m a_{ij}^2}$ for $A = (a_{ij}) \in \mathbb{R}^{n \times m}$
\in	Belong to
\forall	For all
\Rightarrow	Implies
\Leftrightarrow	Equivalent, or if and only if
\otimes	Kronecker product
\emptyset	The empty set
\triangleq	Equal to by definition
$\mathcal{C}^n(\Omega)$	The class of functions having continuous n th derivative on Ω
$\mathcal{L}_2(\Omega)$	The \mathcal{L}_2 space defined on Ω , i.e., $\sqrt{\int_{\Omega} \ f(x)\ ^2 dx} < \infty$ for $f \in \mathcal{L}_2(\Omega)$
$\mathcal{L}_{\infty}(\Omega)$	The \mathcal{L}_{∞} space defined on Ω , i.e., $\sup_{x \in \Omega} \ f(x)\ < \infty$ for $f \in \mathcal{L}_{\infty}(\Omega)$
$\lambda_{\min}(A)$	The minimum eigenvalue of matrix A
$\lambda_{\max}(A)$	The maximum eigenvalue of matrix A
I_n	The n by n identity matrix
$A > 0$	Matrix A is positive definite
$\det(A)$	Determinant of matrix A
A^{-1}	The inverse of matrix A
$\text{tr}(A)$	The trace of matrix A

$\text{vec}(A)$	The vectorization mapping from matrix A into an mn -dimensional column vector for $A \in \mathbb{R}^{m \times n}$
$\text{diag}\{\zeta_i\}$	Also written as $\text{diag}\{\zeta_1, \zeta_2, \dots, \zeta_n\}$ which is an $n \times n$ diagonal matrix with diagonal elements $\zeta_1, \zeta_2, \dots, \zeta_n$
$\tanh(x)$	The hyperbolic tangent function of x
$\text{sgn}(x)$	The sign function of x , i.e., $\text{sgn}(x) = 1$ for $x > 0$, $\text{sgn}(0) = 0$, and $\text{sgn}(x) = -1$ for $x < 0$
$\mathcal{A}(\mathcal{Q})$	The set of admissible controls on \mathcal{Q}
J	Performance index, or cost-to-go, or cost function
J^μ	Performance index, or cost-to-go, or cost function associated with the policy μ
J^*	Optimal performance index function or optimal cost function
V	Value function, or performance index associated with a specific policy
V^*	Optimal value function
L	Lyapunov function
W_f, Y_f	NN weights for function approximation
W_m, Y_m	Model/identifier NN weights
W_o, Y_o	Observer NN weights
W_c, Y_c	Critic NN weights
W_a, Y_a	Action NN weights

Chapter 1

Overview of Adaptive Dynamic Programming

1.1 Introduction

Big data, artificial intelligence (AI), and deep learning are the three topics talked about the most lately in information technology. The recent emergence of deep learning [10, 17, 38, 68, 88] has pushed neural networks (NNs) to become a hot research topic again. It has also gained huge success in almost every branch of AI, including machine learning, pattern recognition, speech recognition, computer vision, and natural language processing [17, 25, 26, 35, 74]. On the other hand, the study of big data often uses AI technologies such as machine learning [80] and deep learning [17]. One particular subject of study in AI, i.e., the computer game of Go, faced a great challenge of dealing with vast amounts of data. The ancient Chinese board game Go has been studied for years with the hope that one day, computer programs can defeat human professional players. The board of game Go consists of 19×19 grid of squares. At the beginning of the game, each of the two players has roughly 360 options for placing each stone. However, the number of potential legal board positions grows exponentially, and it quickly becomes greater than the total number of atoms in the whole universe [103]. Such a number leads to so many directions any given game can move in that makes it impossible for a computer to play by brute force computation of all possible outcomes.

Previous computer programs focused less on evaluating the state of the board positions and more on speeding up simulations of how the game might play out. The Monte Carlo tree search approach was used often in computer game programs, which samples only some of the possible sequences of plays randomly at each step to choose between different possible moves instead of trying to calculate every possible ones. Google DeepMind, an AI company in London acquired by Google in 2014, developed a program called AlphaGo [92] that has shown performance previously thought to be impossible for at least a decade. Instead of exploring various sequences of moves, AlphaGo learns to make a move by evaluating the strength of its position on the board. Such an evaluation was made possible by NN's deep learning capabilities.

Position evaluation (for approximating the optimal cost-to-go function of the game) is the key to success of AlphaGo. Such ideas have been used previously by many researchers in computer games, such as backgammon (TD-Gammon) [100, 101], checkers [87], othello [13], and chess [16]. A reinforcement learning technique called $TD(\lambda)$ was employed in AlphaGo and TD-Gammon for position evaluation. With TD-Gammon, the program has learned to play backgammon at a grandmaster level [100, 101]. On the other hand, AlphaGo has defeated European Go champion Fan Hui (professional 2 dan) by 5 games to 0 [92] and defeated world Go champion Lee Sedol (professional 9 dan) by 4 games to 1 [71, 111].

The success of reinforcement learning (RL) technique in this case relied on NN's deep learning capabilities [10, 17, 38, 68, 88]. The NNs used in AlphaGo have a deep structure with 13 layers. Even though there were reports on the use of RL and related techniques for the computer game of Go [15, 89, 93, 137, 138], it is only with AlphaGo [92] that value networks were obtained using deep NNs to achieve high evaluation accuracy. On the other hand, position evaluation [8, 20, 24, 89, 93] and deep learning [18, 66, 102] have been considered for building programs to play the game of Go; none of them achieved the level of success by AlphaGo [92]. The match of AlphaGo versus Lee Sedol in March 2016 is a history-making event and a milestone in the quest of AI. The defeat over humanity by a machine has also generated huge public interests in AI technology around the world, especially in China, Korea, US, and UK [111]. It will have a lasting impact on the research in AI, deep learning, and RL [142].

RL is a very useful tool in solving optimization problems by employing the principle of optimality from dynamic programming (DP). In particular, in control systems community, RL is an important approach to handle optimal control problems for unknown nonlinear systems. DP provides an essential foundation for understanding RL. Actually, most of the methods of RL can be viewed as attempts to achieve much the same effect as DP, with less computation and without assuming a perfect model of the environment. One class of RL methods is built upon the actor-critic structure, namely adaptive critic designs, where an actor component applies an action or control policy to the environment, and a critic component assesses the value of that action and the state resulting from it. The combination of DP, NN, and actor-critic structure results in the adaptive dynamic programming (ADP) algorithms.

The present book studies ADP with applications to optimal control. Significant efforts will be devoted to the building of value functions which indicate how well the predicted system performance is, which in turn are used for developing the optimal control strategy. Both RL and ADP provide approximate solutions to dynamic programming, and they are closely related to each other. Therefore, there has been a trend to consider the two together as ADPRL (ADP and RL). Examples include IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (started in 2007), IEEE CIS Technical Committee on Adaptive Dynamic Programming and Reinforcement Learning (started in 2008), and a survey article [42] published in 2009. A brief overview of RL will be given in the next section, followed by a more detailed overview of ADP. We review both the basic forms of

ADP as well as iterative forms. A few related books will be briefly reviewed before the end of this chapter.

1.2 Reinforcement Learning

The main research results in RL can be found in the book by Sutton and Barto [98] and the references cited in the book. Even though both RL and the main topic studied in the present book, i.e., ADP, provide approximate solutions to dynamic programming, research in these two directions has been somewhat independent [7] in the past. The most famous algorithms in RL are the temporal difference algorithm [97] and the Q-learning algorithm [112, 113]. Compared to ADP, the area of RL is more mature and has a vast amount of literature (cf. [27, 34, 47, 98]).

An RL system typically consists of the following four components: $\{S, A, R, F\}$, where S is the set of states, A is the set of actions, R is the set of scalar reinforcement signals or rewards, and F is the function describing the transition from one state to the next under a given action, i.e., $F: S \times A \rightarrow S$. A policy π is defined as a mapping $\pi: S \rightarrow A$. At any given time t , the system can be in a state $s_t \in S$, take an action $a_t \in A$ determined by the policy π , i.e., $a_t = \pi(s_t)$, transition to the next state $s' = s_{t+1}$, which is denoted by $s_{t+1} = F(s_t, a_t)$, and at the same time, receive a reward signal $r_{t+1} = r(s_t, a_t, s_{t+1}) \in R$. The goal of RL is to determine a policy to maximize the accumulated reward starting from initial state s_0 at $t = 0$.

An RL task always involves estimating some kind of value functions. A value function estimates how good it is to be in a given state s and it is defined as,

$$V^\pi(s) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} \Big|_{s_0=s} = \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k, s_{k+1}) \Big|_{s_0=s},$$

where $0 \leq \gamma \leq 1$ is a discount factor and $a_k = \pi(s_k)$ and $s_{k+1} = F(s_k, a_k)$ for $k = 0, 1, \dots$. The definition of $V^\pi(s)$ can also be considered to start from s_t , i.e.,

$$V^\pi(s) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \Big|_{s_t=s} = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}, s_{t+k+1}) \Big|_{s_t=s}, \quad (1.2.1)$$

where $a_{t+k} = \pi(s_{t+k})$ and $s_{t+k+1} = F(s_{t+k}, a_{t+k})$ for $k = 0, 1, \dots$. $V^\pi(s)$ is referred to as the state-value function for policy π . On the other hand, the action-value function for policy π estimates how good it is to perform a given action a in a given state s under the policy π and it is defined as,

$$Q^\pi(s, a) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \Big|_{s_t=s, a_t=a} = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}, s_{t+k+1}) \Big|_{s_t=s, a_t=a}, \quad (1.2.2)$$

where $a_{t+k} = \pi(s_{t+k})$ for $k = 1, 2, \dots$, and $s_{t+k+1} = F(s_{t+k}, a_{t+k})$ for $k = 0, 1, \dots$.

The goal of RL is to find a policy that is optimal, i.e., a policy that is better than or equal to all other policies. The optimal policy may not be unique, and all the optimal policies are denoted by π^* . They have the same optimal state-value function defined by

$$V^*(s) = \sup_{\pi} \{V^{\pi}(s)\}, \quad (1.2.3)$$

or the same optimal action-value function defined by

$$Q^*(s, a) = \sup_{\pi} \{Q^{\pi}(s, a)\}. \quad (1.2.4)$$

They satisfy the Bellman optimality equation (or, Bellman equation)

$$V^*(s) = \max_a \{r(s, a, s') + \gamma V^*(s')\}, \quad (1.2.5)$$

or

$$Q^*(s, a) = r(s, a, s') + \gamma \max_{a'} \{Q^*(s', a')\}. \quad (1.2.6)$$

The optimal policy is determined as follows

$$\pi^*(s) = \arg \max_a \{r(s, a, s') + \gamma V^*(s')\}, \quad (1.2.7)$$

or

$$\pi^*(s) = \arg \max_a Q^*(s, a). \quad (1.2.8)$$

If we were to consider a stochastic process instead of a deterministic process, the transition from state s to the next state s' is described by a probability distribution

$$P(s' | s, a) = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\},$$

and the expected value of the next reward is given by

$$r(s, a, s') = \mathbb{E}\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\},$$

where $\mathbb{E}(\cdot)$ is the expected value. In this case, (1.2.1) is rewritten as

$$V^{\pi}(s) = \mathbb{E}_{\pi} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\}, \quad (1.2.9)$$

where the expected value $\mathbb{E}_{\pi}\{\cdot\}$ is taken under the policy π .

This book will not consider stochastic processes. Therefore, all of our descriptions are for deterministic systems, including the introduction to RL in this section.

Next, we consider how to compute the state-value function V^π for an arbitrary policy π . This procedure is called policy evaluation in the literature which is described by

$$V^\pi(s) = r(s, a, s') + \gamma V^\pi(s') = r(s, \pi(s), F(s, \pi(s))) + \gamma V^\pi(F(s, \pi(s))). \quad (1.2.10)$$

The value function can also be determined using iterative policy evaluation starting from an arbitrary initial approximation $V_0(\cdot)$,

$$V_{i+1}(s) = r(s, a, s') + \gamma V_i(s'), \quad i = 0, 1, 2, \dots \quad (1.2.11)$$

The goal is to obtain the value function $V^\pi(s)$ given policy π .

After $V^\pi(s)$ is obtained or evaluated, we obtain an improved policy by

$$\pi'(s) = \arg \max_a \{r(s, a, s') + \gamma V^\pi(s')\}. \quad (1.2.12)$$

Policy iteration procedure involves computation cycles between policy evaluation (1.2.10) or (1.2.11) and policy improvement (1.2.12) in order to find the optimal policy.

Another way to determine the optimal policy is to use the value iteration procedure. In this case, for each i , $i = 0, 1, \dots$, the state-value functions V_i for a policy π_i can also be computed using value iteration described by value function update

$$V_{i+1}(s) = \max_a \{r(s, a, s') + \gamma V_i(s')\}, \quad (1.2.13)$$

and policy improvement

$$\pi_{i+1}(s) = \arg \max_a \{r(s, a, s') + \gamma V_i(s')\}. \quad (1.2.14)$$

The computation shall continue until convergence, e.g., when $|V_{i+1}(s) - V_i(s)| \leq \varepsilon$, $\forall s$, to obtain $V^*(s) \approx V_{i+1}(s)$, where ε is a small positive number. Then, the optimal policy is approximated as $\pi^*(s) \approx \pi_{i+1}(s)$.

The temporal difference (TD) method [97] is developed to estimate the value function for a given policy. The TD algorithm is described by

$$V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)], \quad (1.2.15)$$

or

$$V_{i+1}(s_t) = V_i(s_t) + \alpha [r_{t+1} + \gamma V_i(s_{t+1}) - V_i(s_t)], \quad i = 0, 1, 2, \dots, \quad (1.2.16)$$

where $\alpha > 0$ is a step size. This algorithm is also called TD(0) (compared to TD(λ) to be introduced later). The update in (1.2.15) is obtained according to the following general formula:

$\text{NewValue} \leftarrow \text{OldValue} + \text{Update} = \text{OldValue} + \text{StepSize} \times (\text{Target} - \text{OldValue})$,

which indicates a step of move toward the ‘Target.’

Q-learning [112, 113] is an off-policy¹ [98] TD algorithm and it is described by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a \{Q(s_{t+1}, a)\} - Q(s_t, a_t) \right], \quad (1.2.17)$$

or

$$Q_{i+1}(s_t, a_t) = Q_i(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a \{Q_i(s_{t+1}, a)\} - Q_i(s_t, a_t) \right], \quad i = 0, 1, 2, \dots \quad (1.2.18)$$

Based on the value function obtained above, an improved policy can be determined as

$$\pi'(s_t) = \arg \max_a Q(s_t, a), \quad (1.2.19)$$

or

$$\pi_{i+1}(s_t) = \arg \max_a Q_{i+1}(s_t, a). \quad (1.2.20)$$

A modification to the above Q-learning algorithm so that it becomes an on-policy² [82] is called Sarsa [96] whose name comes from the fact that the algorithm uses the following five elements of the problem: $\{s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}\}$. It updates the action-value function as follows

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \quad (1.2.21)$$

or

$$Q_{i+1}(s_t, a_t) = Q_i(s_t, a_t) + \alpha [r_{t+1} + \gamma Q_i(s_{t+1}, a_{t+1}) - Q_i(s_t, a_t)], \quad i = 0, 1, 2, \dots \quad (1.2.22)$$

A more general TD algorithm called TD(λ) [97] has been quite popular [8, 13, 15, 16, 20, 24, 87, 89, 92, 93, 100, 101, 137, 138]. Using TD(λ), the update equation (1.2.15) now becomes

$$\begin{aligned} V_{t+1}(s) &= V_t(s) + \Delta V_t(s) \\ &= V_t(s) + \alpha z_t(s) [r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)]. \end{aligned} \quad (1.2.23)$$

In the above, $z_t(s)$ is defined as the eligibility trace of state s given by

$$z_t(s) = \begin{cases} \gamma \lambda z_{t-1}(s) + 1, & \text{if } s = s_t, \\ \gamma \lambda z_{t-1}(s), & \text{if } s \neq s_t, \end{cases}$$

where $z_0(s) = 0, \forall s$, and $0 \leq \lambda \leq 1$ is called the trace-decay parameter.

¹Off-policy learning is defined as evaluating one policy while following another.

²On-policy learning estimates the value of a policy while using it for control.

Similar ideas can be applied to Sarsa [82] to obtain the following Sarsa(λ) update equation

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha z_t(s, a) [r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)], \quad (1.2.24)$$

where

$$z_t(s, a) = \begin{cases} \gamma \lambda z_{t-1}(s, a) + 1, & \text{if } s = s_t \text{ and } a = a_t, \\ \gamma \lambda z_{t-1}(s, a), & \text{otherwise,} \end{cases}$$

with $z_0(s, a) = 0, \forall s, a$. It can also be applied to Q-learning [112, 113] to obtain the following Q(λ) update equation

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \delta_t z_t(s, a), \quad (1.2.25)$$

where

$$\delta_t = r_{t+1} + \gamma \max_{a'} \{Q_t(s_{t+1}, a')\} - Q_t(s_t, a_t),$$

$$z_t(s, a) = I_{s, s_t} I_{a, a_t} + \begin{cases} \gamma \lambda z_{t-1}(s, a), & \text{if } Q_{t-1}(s_t, a_t) = \max_a \{Q_{t-1}(s_t, a)\}, \\ 0, & \text{otherwise} \end{cases}$$

$$I_{x,y} = \begin{cases} 1, & \text{if } x = y, \\ 0, & \text{otherwise,} \end{cases}$$

and $z_0(s, a) = 0, \forall s, a$.

TD, Q-learning, Sarsa, TD(λ), Sarsa(λ), and Q(λ) are developed to obtain the value functions $V(s)$ and $Q(s, a)$. The next step after the value function is obtained is to determine a policy or update an existing policy according to (1.2.12), (1.2.14), (1.2.19), or (1.2.20) (cf. (1.2.7) and (1.2.8)). The process can be repeated until an optimal policy is obtained. Such a procedure solves the Bellman equation (1.2.5) or (1.2.6) iteratively to provide approximate solutions.

1.3 Adaptive Dynamic Programming

There are several schemes of dynamic programming [9, 11, 23, 41]. One can consider discrete-time systems or continuous-time systems, linear systems or nonlinear systems, time-invariant systems or time-varying systems, deterministic systems or stochastic systems, etc. Discrete-time (deterministic) nonlinear (time-invariant) dynamical systems will be discussed first. Time-invariant nonlinear systems cover most of the application areas and discrete-time is the basic consideration for digital implementation.

Consider the following discrete-time nonlinear systems described by

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \dots, \quad (1.3.1)$$

where $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^m$ is the control vector, and $F: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is the nonlinear system function. Starting from an initial state $x_0 \in \mathbb{R}^n$, one needs to choose a control sequence $u_k, k = 0, 1, 2, \dots$, such that certain objectives can be achieved when the control sequence is applied to system (1.3.1).

Suppose that one associates with system (1.3.1) the following performance index (or cost-to-go, or simply, cost)

$$J(x_k, \underline{u}_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, u_i), \quad (1.3.2)$$

where $\underline{u}_k = (u_k, u_{k+1}, \dots)$ denotes the control sequence starting at time k , $U(\cdot, \cdot) \geq 0$ is called the utility function and γ is the discount factor with $0 < \gamma \leq 1$. Note that the function J is dependent on the initial time k and the initial state x_k , and it is referred to as the cost-to-go of state x_k . The cost in this case accumulates indefinitely; this kind of problems is referred to as infinite-horizon problems in dynamic programming. On the other hand, in finite-horizon problems, the cost accumulates over a finite number of steps. Very often, it is desired to determine $\underline{u}_0 = (u_0, u_1, \dots)$ so that $J(x_0, \underline{u}_0)$ is optimized (i.e., maximized or minimized). We will use $\underline{u}_0^* = (u_0^*, u_1^*, \dots)$ and $J^*(x_0)$ to denote the optimal control sequence and the optimal cost function, respectively.

The objective of dynamic programming problem in this book is to determine a control sequence $u_k^*, k = 0, 1, \dots$, so that the function J (i.e., the cost) in (1.3.2) is minimized. The optimal cost function is defined as

$$J^*(x_0) = \inf_{\underline{u}_0} J(x_0, \underline{u}_0) = J(x_0, \underline{u}_0^*),$$

which is dependent upon the initial state x_0 .

The control action may be determined as a function of the state. In this case, we write $u_k = u(x_k), \forall k$. Such a relationship, or mapping $u: \mathbb{R}^n \rightarrow \mathbb{R}^m$, is called feedback control, or control policy, or policy. It is also called control law. For a given control policy μ , the cost function in (1.3.2) is rewritten as

$$J^\mu(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, \mu(x_i)), \quad (1.3.3)$$

which is the cost function for system (1.3.1) starting at x_k when the policy $u_k = \mu(x_k)$ is applied. The optimal cost for system (1.3.1) starting at x_0 is determined as

$$J^*(x_0) = \inf_{\mu} J^\mu(x_0) = J^{\mu^*}(x_0),$$

where μ^* indicates the optimal policy.

This PDF document was edited with **Icecream PDF Editor**.

Upgrade to PRO to remove watermark.

Dynamic programming is based on Bellman's principle of optimality [9, 11, 23, 41]: An optimal (control) policy has the property that no matter what previous decisions have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions.

Suppose that one has computed the optimal cost $J^*(x_{k+1})$ from time $k + 1$ to the terminal time, for all possible states x_{k+1} , and that one has also found the optimal control sequences from time $k + 1$ to the terminal time. The optimal cost results when the optimal control sequence $u_{k+1}^*, u_{k+2}^*, \dots$, is applied to the system with initial state x_{k+1} . Note that the optimal control sequence depends on x_{k+1} . If one applies an arbitrary control action u_k at time k and then uses the known optimal control sequence from $k + 1$ on, the resulting cost will be

$$U(x_k, u_k) + \gamma U(x_{k+1}, u_{k+1}^*) + \gamma^2 U(x_{k+2}, u_{k+2}^*) + \dots = U(x_k, u_k) + \gamma J^*(x_{k+1}),$$

where x_k is the state of the system at time k and x_{k+1} is determined by (1.3.1). According to Bellman, the optimal cost from time k on is equal to

$$J^*(x_k) = \min_{u_k} \{U(x_k, u_k) + \gamma J^*(x_{k+1})\} = \min_{u_k} \{U(x_k, u_k) + \gamma J^*(F(x_k, u_k))\}. \quad (1.3.4)$$

The optimal control u_k^* at time k is the u_k that achieves this minimum, i.e.,

$$u_k^* = \arg \min_{u_k} \{U(x_k, u_k) + \gamma J^*(x_{k+1})\}. \quad (1.3.5)$$

Equation (1.3.4) is the principle of optimality for discrete-time systems. Its importance lies in the fact that it allows one to optimize over only one control vector at a time by working backward in time.

Dynamic programming is a very useful tool in solving optimization and optimal control problems. In particular, it can easily be applied to nonlinear systems with or without constraints on the control and state variables. Equation (1.3.4) is called the functional equation of dynamic programming or Bellman equation and is the basis for computer implementation of dynamic programming. In the above, if the function F in (1.3.1) and the cost function J in (1.3.2) are known, the solution for u^* becomes a simple optimization problem. However, it is often computationally untenable to run true dynamic programming due to the backward numerical process required for its solutions, i.e., as a result of the well-known “curse of dimensionality” [9, 23, 41]. The optimal cost function J^* , which is the theoretical solution to the Bellman equation (1.3.4), is very difficult to obtain, except systems satisfying some very good conditions such as linear time-invariant systems. Over the years, progress has been made to circumvent the curse of dimensionality by building a system, called critic, to approximate the cost function in dynamic programming. The idea is to approximate dynamic programming solutions using a function approximation structure such as NNs to approximate the cost function. Such an approach is called adaptive dynamic programming in this book, though it was previously called adaptive critic designs or approximate dynamic programming.

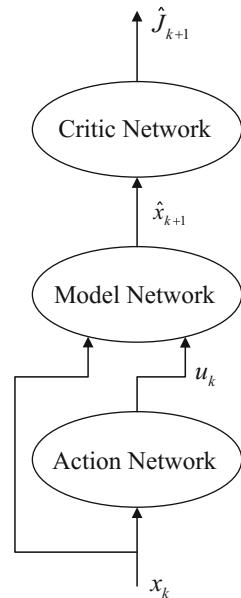
1.3.1 Basic Forms of Adaptive Dynamic Programming

In 1977, Paul Werbos [124] introduced an approach for approximate dynamic programming that was later called adaptive critic designs (ACDs). ACDs have received increasing attention (cf. [2, 6, 7, 15, 21, 22, 36, 39, 50, 55, 65, 72, 73, 78, 79, 83, 90, 91, 104, 125–128, 137]). In the literature, there are several synonyms used for “adaptive critic designs” including “approximate dynamic programming” [40, 76, 90, 128], “asymptotic dynamic programming” [83], “adaptive dynamic programming” [72, 73, 139], “neuro-dynamic programming” [12], “neural dynamic programming” [133], “relaxed dynamic programming” [46, 81], and “reinforcement learning” [14, 40, 98, 106, 127]. No matter what we call it, in all these cases, the goal is to approximate the solutions of dynamic programming. Because of this, the term “approximate dynamic programming” has been quite popular in the past.

In this book, we will use the term ADP to represent “adaptive dynamic programming” or “approximate dynamic programming.” ADP has potential applications in many fields, including controls, management, logistics, economy, military, aerospace, etc. This book contains mostly applications in optimal control of nonlinear systems. A typical design of ADP consists of three modules—critic, model, and action [127, 128], as shown in Fig. 1.1. The critic network will give an estimation of the cost function J , which is often a Lyapunov function, at least for some of the deterministic systems.

The present book considers the case where each module is an NN (refer to, e.g., [5, 141, 145] for ADP implementations using fuzzy systems). In the ADP scheme

Fig. 1.1 The three modules of ADP/ACD



shown in Fig. 1.1, the critic network outputs the function \hat{J} , which is an estimate of the function J in (1.3.2). This is done by minimizing the following square error measure over time

$$\|E_k\| = \frac{1}{2} \sum_k E_k^2 = \frac{1}{2} \sum_k (\hat{J}_k - U_k - \gamma \hat{J}_{k+1})^2, \quad (1.3.6)$$

where $\hat{J}_k = \hat{J}(x_k, W_c)$ and W_c represents the parameters of the critic network. The function U_k is the same utility function as the one in (1.3.2) which indicates the performance of the overall system. The function U_k given in a problem is usually a function of x_k and u_k , i.e., $U_k = U(x_k, u_k)$. When $E_k = 0$ for all k , (1.3.6) implies that

$$\begin{aligned} \hat{J}_k &= U_k + \gamma \hat{J}_{k+1} \\ &= U_k + \gamma (U_{k+1} + \gamma \hat{J}_{k+2}) \\ &= \dots \\ &= \sum_{i=k}^{\infty} \gamma^{i-k} U_i, \end{aligned} \quad (1.3.7)$$

which is exactly the same as the cost function in (1.3.2). It is therefore clear that minimizing the error function in (1.3.6), we will have an NN trained so that its output \hat{J} becomes an estimate of the cost function J defined in (1.3.2).

The model network in Fig. 1.1 learns the nonlinear function F given in equation (1.3.1); it can be trained previously offline [79, 128], or trained in parallel with the critic and action networks [83].

After the model network is obtained, the critic network will be trained. The critic network gives an estimate of the cost function. The training of the critic network in this case is achieved by minimizing the error function defined in (1.3.6), for which many standard NN training algorithms can be utilized [29, 146]. Note that in Fig. 1.1, the output of the critic network $\hat{J}_{k+1} = \hat{J}(\hat{x}_{k+1}, W_c)$ is an approximation to the cost function J at time $k + 1$, where \hat{x}_{k+1} is not a real trajectory but a prediction of the states from the model network.

After the critic network's training is finished, one can start the action network's training with the objective of minimizing $U_k + \gamma \hat{J}_{k+1}$, through the use of the control signal $u_k = u(x_k, W_a)$, where W_a represents the parameters of the action network. Once an action network is trained this way, we will have an NN trained so that it will generate as its output an optimal, or at least, a suboptimal control signal depending on how well the performance of the critic network is. Recall that the goal of dynamic programming is to obtain an optimal control sequence as in (1.3.5), which minimizes the function J in (1.3.2). The key here is to interactively build a link between present actions and future consequences via an estimate of the cost function.

After the action network's training cycle is complete, one may check the system performance, then stop or continue the training procedure by going back to the critic

network's training cycle again, if the performance is not acceptable yet [78, 79]. This process will be repeated until an acceptable system performance is reached. The three networks will be connected as shown in Fig. 1.1. As a part of the process, the control signal u_k will be applied to the external environment and obtain a new state x_{k+1} . Meanwhile, the model network gives an approximation of the next state \hat{x}_{k+1} . By minimizing $\|x_{k+1} - \hat{x}_{k+1}\|$, the model network can be trained.

The training of the action network is done through its parameter updates to minimize the values of $U_k + \gamma \hat{J}_{k+1}$ while keeping the parameters of the critic and model networks fixed. The gradient information is propagated backward through the critic network to the model network and then to the action network, as if the three networks formed one large feedforward network (cf. Fig. 1.1). This implies that the model network in Fig. 1.1 is required for the implementation of ADP in the present case. Even in the case of known function F , one still needs to build a model network so that the action network can be trained by backpropagation algorithm.

Two approaches for the training of critic network are provided in [64]: a forward-in-time approach and a backward-in-time approach. Figure 1.2 shows the diagram of forward-in-time approach. In this approach, we view \hat{J}_k in (1.3.6) as the output of the critic network to be trained and choose $U_k + \gamma \hat{J}_{k+1}$ as the training target. Note that \hat{J}_k and \hat{J}_{k+1} are obtained using state variables at different time instances. Figure 1.3 shows the diagram of backward-in-time approach. In this approach, we view \hat{J}_{k+1} in (1.3.6) as the output of the critic network to be trained and choose $(\hat{J}_k - U_k)/\gamma$ as the training target. Note that both forward-in-time and backward-in-time approaches try to minimize the error measure in (1.3.6) and satisfy the requirement in (1.3.7). In Figs. 1.2 and 1.3, \hat{x}_{k+1} is the output from the model network.

From the TD algorithm in (1.2.15), we can see that the learning objective is to minimize $|r_{t+1} + \gamma V(s_{t+1}) - V(s_t)|$ by using $r_{t+1} + \gamma V(s_{t+1})$ as the learning target. This gives the same idea as in the forward-in-time approach shown in Fig. 1.2, where the target is $U_k + \gamma \hat{J}_{k+1}$. The only difference is the definition of reward function. In (1.2.15), it is defined as $r_{t+1} = r(s_t, a_t, s_{t+1})$, whereas in Fig. 1.2, it is defined as $U_k = U(x_k, u_k)$, where the current times are t and k , respectively. We will make clear

Fig. 1.2 Forward-in-time approach

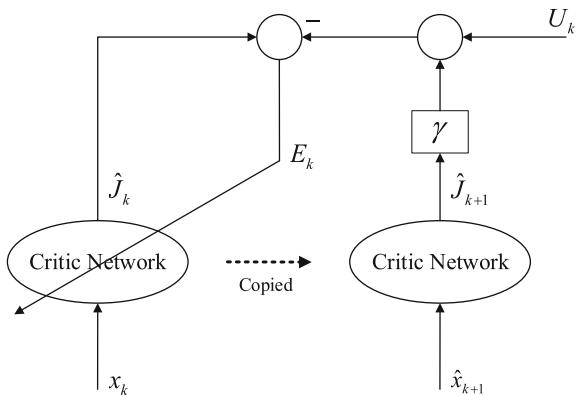
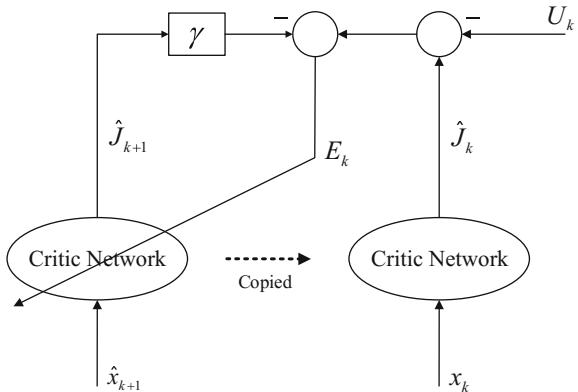


Fig. 1.3 Backward-in-time approach

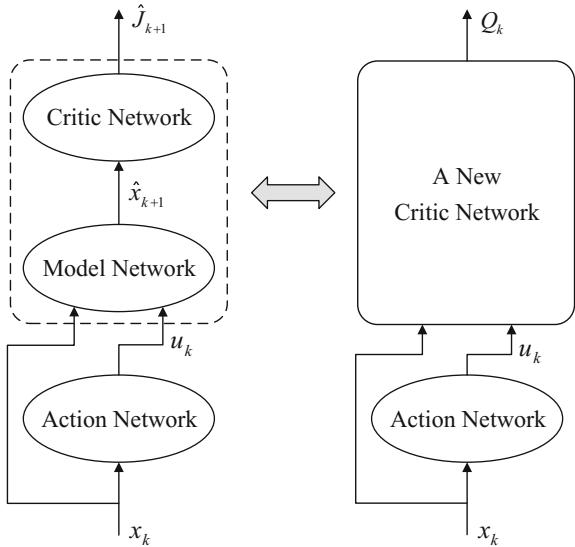


later the reason behind this one-step time difference between r_{t+1} and U_k . Even in $\text{TD}(\lambda)$ given by (1.2.23), the same learning objective is utilized. However, in TD and $\text{TD}(\lambda)$, the update of value functions at each step only makes a move according to the step size toward the target, and presumably, it does not reach the target. On the other hand, in the forward-in-time and backward-in-time approaches, the training will only be performed for certain number of steps, e.g., 3–5 steps [39] or 50 steps [91]. Such a move may or may not reach the target, but for sure will lead to a move in the direction of the target.

Two most important advances of ADP for control start with [79] and [91]. Reference [79] provides a detailed summary of the major developments in ADP up to 1997. Before that, major references are papers by Werbos such as [124–128]. Reference [91] makes significant contributions to model-free ADP. Using the approach of [91], the model network in Fig. 1.1 is not needed anymore. Several practical examples are included in [91] for demonstration which include single inverted pendulum and triple inverted pendulum. The training approach of [91] can be considered as a backward-in-time approach. Reference [64] is also about model-free ADP. It is a model-free, action-dependent adaptive critic design since we can view the model network and the critic network together as another NN, which we call it still a critic network, as illustrated in Fig. 1.4.

The model-free ADP has been called action-dependent ACDs by Werbos. According to [79, 128], ADP approaches were classified into several main schemes: heuristic dynamic programming (HDP), action-dependent HDP (ADHDP; note the prefix “action-dependent” (AD) used hereafter), dual heuristic dynamic programming (DHP), ADDHP, globalized DHP (GDHP), and ADGDHP. HDP is the basic version of ADP, which is described in Fig. 1.1 or the left side of Fig. 1.4. According to Werbos [128], TD algorithms share the same idea as that of HDP to use the same learning objective. On the other hand, the equivalence of ADHDP and Q-learning has been argued by Werbos [128] as well. Both ADHDP and Q-learning (including Sarsa as well) use value functions that are functions of the state and action. For ADHDP, which is described in the right side of Fig. 1.4, the critic network training is done by

Fig. 1.4 Definition of a new critic network for model-free ADP



minimizing the following square error measure over time

$$\|E_q\| = \frac{1}{2} \sum_k E_{qk}^2 = \frac{1}{2} \sum_k (Q_{k-1} - U_k - \gamma Q_k)^2, \quad (1.3.8)$$

where $Q_k = Q(x_k, u_k, W_{qc})$ and W_{qc} represents the parameters of the critic network. When $E_{qk} = 0$ for all k , (1.3.8) implies that

$$\begin{aligned} Q_k &= U_{k+1} + \gamma Q_{k+1} \\ &= U_{k+1} + \gamma (U_{k+2} + \gamma Q_{k+2}) \\ &= \dots \\ &= \sum_{i=k+1}^{\infty} \gamma^{i-k-1} U_i, \end{aligned} \quad (1.3.9)$$

which is exactly the cost function \hat{J}_{k+1} defined in (1.3.7). From Fig. 1.4, we can see that $Q_k = Q(x_k, u_k, W_{qc})$ is equivalent to $\hat{J}_{k+1} = \hat{J}(\hat{x}_{k+1}, W_c) = \hat{J}(\hat{F}(x_k, u_k), W_c)$. The two outputs will be the same given the same inputs x_k and u_k . However, the two relationships are different. In model-free ADP, the output Q_k is explicitly a function of x_k and u_k , while the model network \hat{F} becomes totally hidden and internal. On the other hand, with model network, the output \hat{J}_{k+1} is an explicit function of \hat{x}_{k+1} which in turn is a function of x_k and u_k through the model network \hat{F} .

The one-step time difference here between \hat{J}_k in (1.3.7) and Q_k in (1.3.9) has exactly the same reasoning behind the one-step time difference between the TD

algorithm's r_{t+1} and the HDP algorithm's U_k mentioned earlier. In the HDP structure described above, a model network is used which leads to

$$\hat{J}_k \rightarrow U_k + \gamma U_{k+1} + \gamma^2 U_{k+2} + \dots$$

In the ADHDP structure, there is no model network, and thus the expression becomes

$$Q_k \rightarrow U_{k+1} + \gamma U_{k+2} + \gamma^2 U_{k+3} + \dots$$

As a matter of fact, for the RL system discussed in the TD methods, the value function is defined without model network [97, 98]. Therefore, the value function $V(s_t)$ in (1.2.15) is defined similarly to the function Q_k above and it starts with r_{t+1} instead of r_t .

In addition to the basic structures reviewed above for ADP, there are also other structures proposed in the literature, such as [30, 75].

1.3.2 Iterative Adaptive Dynamic Programming

A popular method to determine the cost function and the optimal cost function is to use value iteration. In this case, a different set of notation has been used in the literature. The cost function J^μ defined above will be called value function V^μ , i.e., $V^\mu(x_k) = J^\mu(x_k)$, $\forall x_k$. Similarly, $V^*(x_0) = \inf_\mu V^\mu(x_0)$ is called the optimal value function.

Note that similar to the case of cost function J defined above, the value function V will also have the following three forms. (i) $V(x_k, \underline{u}_k)$ represents the value of the cost function of system (1.3.1) starting at x_k when the control sequence $\underline{u}_k = (u_k, u_{k+1}, \dots)$ is applied. (ii) $V^\mu(x_k)$ tells the value of the cost function of system (1.3.1) starting at x_k when the control policy $u_k = \mu(x_k)$ is applied. (iii) $V^*(x_k)$ is the optimal cost function of system (1.3.1) starting at x_k . When the context is clear, for convenience in this book, the notation $V(x_k)$ will be used to represent $V(x_k, \underline{u}_k)$ and $V^\mu(x_k)$. Such a use of notation for value functions has been quite standard in the literature. In subsequent chapters, there will also be cases where the value function is a function of x_k and u_k (not explicitly as a function of \underline{u}_k). Thus, $V(x_k, u_k)$ will be appropriate. Also, there are cases where the value function is time-varying, such that $V(x_k, u_k, k)$ will be appropriate. As a convention, we will use $V(x_k)$ to represent $V(x_k, u_k)$ and $V(x_k, u_k, k)$ when the context is clear.

As stated earlier in (1.2.11) and (1.2.13) as well as in (1.2.16), (1.2.18), and (1.2.22), the Bellman equation can be solved using iterative approaches.

We rewrite the Bellman optimality equation (1.3.4) here,

$$J^*(x_k) = \min_{u_k} \{U(x_k, u_k) + \gamma J^*(x_{k+1})\} = \min_{u_k} \{U(x_k, u_k) + \gamma J^*(F(x_k, u_k))\}. \quad (1.3.10)$$

Our goal is to solve for a function J^* which satisfies (1.3.10) and which in turn leads to the optimal control solution u_k^* given by (1.3.5), rewritten below,

$$u_k^* = \arg \min_{u_k} \{U(x_k, u_k) + \gamma J^*(x_{k+1})\}. \quad (1.3.11)$$

One way to solve (1.3.10) is to use the following iterative approach. Replace the function J^* in (1.3.10) using V , i.e., using a value function. Now, we need to solve for function V from

$$V(x_k) = \min_{u_k} \{U(x_k, u_k) + \gamma V(x_{k+1})\}. \quad (1.3.12)$$

This equation can be further rewritten as

$$V_i(x_k) = \min_{u_k} \{U(x_k, u_k) + \gamma V_{i-1}(x_{k+1})\}, \quad i = 1, 2, \dots \quad (1.3.13)$$

This is similar to solving the algebraic equation $x = f(x)$ using iterative method from $x_i = f(x_{i-1})$. Starting from x_0 , we use the iteration to obtain $x_1 = f(x_0)$, $x_2 = f(x_1)$, \dots . We can get the solution as $x_\infty = f(x_\infty)$, if the iterative process is convergent. For (1.3.13), one can start with a function V_0 and the iteration gives V_1 , V_2 , and so on. One would hope that a solution $V_\infty(x_k)$ is obtained when i reaches ∞ . Of course, such a solution can only be obtained if the iterative process is convergent.

Using the procedure above, we would hope to obtain a solution which is also the optimal solution as required by the solution of dynamic programming. During the iterative solution process, corresponding to each iterative value function V_i obtained, there will also be a control signal determined as

$$v_i(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + \gamma V_i(x_{k+1})\}. \quad (1.3.14)$$

This sequence of control signals $\{v_0, v_1, \dots\}$ is called the iterative control law sequence. We would hope to obtain a sequence of stable control laws, or at least a stable control law when it is optimal.

The above gives the rational of iterative ADP based on value iteration. We need theoretical results regarding qualitative analysis of the iterative solution including stability, convergence, and optimality. Stability is the fundamental requirement of any control system. Convergence of the iterative solution process is required in order for the above procedure to be meaningful. Furthermore, only requiring the convergence of $\{V_i\}$ is not enough, since we will also need the iterative solutions to converge to the optimal solution J^* .

The simplest choice of V_0 to start the iteration is $V_0(x_k) \equiv 0, \forall x_k$ [3, 46, 81]. In [3, 46, 81], undiscounted optimal control problems are considered, where $\gamma = 1$. In this case, (1.3.13) becomes

$$V_i(x_k) = \min_{u_k} \{U(x_k, u_k) + V_{i-1}(x_{k+1})\}, \quad i = 1, 2, \dots \quad (1.3.15)$$

Rantzer proved the following proposition.

Proposition 1.3.1 (Convergence of value iteration [46, 81]) *Suppose that, for all x_k and for all u_k , the inequality $J^*(F(x_k, u_k)) \leq \rho U(x_k, u_k)$ holds uniformly for some $\rho < \infty$ and that $\eta J^*(x_k) \leq V_0(x_k) \leq J^*(x_k)$ for some $0 \leq \eta \leq 1$. Then, the sequence $\{V_i\}$ defined iteratively by (1.3.15) approaches J^* according to the inequalities*

$$\left[1 + \frac{\eta - 1}{(1 + \rho^{-1})^i}\right]J^*(x_k) \leq V_i(x_k) \leq J^*(x_k), \quad \forall x_k. \quad (1.3.16)$$

Proposition 1.3.1 clearly shows the convergence of value iteration, i.e., $V_i(x_k) \rightarrow J^*(x_k)$ as $i \rightarrow \infty$.

On the other hand, the affine version of system (1.3.1) has been studied often, which is given by

$$x_{k+1} = f(x_k) + g(x_k)u(x_k), \quad k = 0, 1, 2, \dots, \quad (1.3.17)$$

where $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^m$ is the control vector, and $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $g: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ are nonlinear system functions. The utility function in this case is usually chosen as a quadratic form given by

$$U(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k,$$

where Q and R are positive-definite matrices with appropriate dimensions. In a seminal paper by Al-Tamimi, Lewis, and Abu-Khalaf [3], an iterative ADP algorithm is derived as follows. Starting with the zero initial value function, i.e., $V_0(x_k) \equiv 0, \forall x_k$, solve for v_0 as

$$v_0(x_k) = \arg \min_{u_k} \{x_k^T Q x_k + u_k^T R u_k + V_0(x_{k+1})\},$$

where $V_0(x_{k+1}) = 0$. Once the policy v_0 is determined, the next value function is computed as

$$\begin{aligned} V_1(x_k) &= x_k^T Q x_k + v_0^T(x_k) R v_0(x_k) + V_0(x_{k+1}) \\ &= x_k^T Q x_k + v_0^T(x_k) R v_0(x_k) + V_0(f(x_k) + g(x_k)v_0(x_k)). \end{aligned}$$

The iteration will then be performed between control policies

$$\begin{aligned} v_i(x_k) &= \arg \min_{u_k} \{x_k^T Q x_k + u_k^T R u_k + V_i(x_{k+1})\} \\ &= \arg \min_{u_k} \{x_k^T Q x_k + u_k^T R u_k + V_i(f(x_k) + g(x_k)u_k)\} \\ &= \frac{1}{2} R^{-1} g^T(x_k) \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}}, \end{aligned}$$

and value functions

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

for $i = 1, 2, \dots$. The following results were shown in [3].

- (1) Starting from $V_0(x_k) = 0$, the sequence $\{V_i(x_k)\}$ generated by the above iteration will be monotonically nondecreasing and has an upper bound. Therefore, the limit of $V_i(x_k)$ when $i \rightarrow \infty$, i.e., $V_\infty(x_k)$ exists.
- (2) $V_\infty(x_k) = J^*(x_k)$, i.e., the iterative solution converges to the optimal solution.
- (3) The iterative control law converges to the optimal control law, i.e., $v_\infty(x_k) = u^*(x_k)$.

Along the line of [3], some of the new developments can be found in [58, 59, 61, 63, 84, 108, 109, 115, 143, 144]. Along the line of [46, 81], there are also significant developments [44, 54, 60, 114, 117, 121–123, 132].

The introduction given above is for discrete-time nonlinear systems. We next introduce briefly ADP for continuous-time nonlinear systems before we conclude our review.

1.3.3 ADP for Continuous-Time Systems

For continuous-time systems, the cost function J is also the key to dynamic programming. By minimizing J , one gets the optimal cost function J^* , which is often a Lyapunov function of the system. As a consequence of the Bellman's principle of optimality, J^* satisfies the Hamilton–Jacobi–Bellman (HJB) equation. But usually, one cannot get the analytical solution of the HJB equation. Even to find an accurate numerical solution is very difficult due to the so-called curse of dimensionality.

Continuous-time nonlinear systems can be described by

$$\dot{x}(t) = F(x(t), u(t)), \quad t \geq t_0,$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the state and the control vectors, and $F(x, u)$ is a continuous nonlinear system function. The cost in this case is defined as

$$J(x_0, u) = \int_{t_0}^{\infty} U(x(\tau), u(\tau)) d\tau,$$

with nonnegative utility function $U(x, u) \geq 0$, where $x(t_0) = x_0$. The Bellman's principle of optimality can also be applied to continuous-time systems. In this case, the optimal cost

$$J^*(x(t)) = \min_{u(t)} \{J(x(t), u(t))\}, \quad t \geq t_0,$$

satisfies the HJB equation

$$-\frac{\partial J^*}{\partial t} = \min_{u(t)} \left\{ U(x, u) + \left(\frac{\partial J^*}{\partial x} \right)^T F(x, u) \right\}. \quad (1.3.18)$$

The HJB equation in (1.3.18) can be derived from the Bellman's principle of optimality (1.3.4) [41]. Meanwhile, the optimal control $u^*(t)$ will be the one that minimizes the cost function,

$$u^*(t) = \arg \min_{u(t)} \{J(x(t), u(t))\}, \quad t \geq t_0. \quad (1.3.19)$$

In 1994, Saridis and Wang [86] studied the nonlinear stochastic systems described by

$$dx = f(x, t) dt + g(x, t)u dt + h(x, t)dw, \quad t_0 \leq t \leq T, \quad (1.3.20)$$

with the cost function

$$J(x_0, u) = \mathbb{E} \left\{ \int_{t_0}^T (Q(x, t) + u^T u) dt + \phi(x(T), T) : x(t_0) = x_0 \right\},$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and $w \in \mathbb{R}^k$ are the state vector, the control vector, and a separable Wiener process; f , g and h are measurable system functions; and Q and ϕ are nonnegative functions. A value function V is defined as

$$V(x, t) = \mathbb{E} \left\{ \int_t^T (Q(x, t) + u^T u) dt + \phi(x(T), T) : x(t_0) = x_0 \right\}, \quad t \in I,$$

where $I \triangleq [t_0, T]$. The HJB equation is modified to become the following equation

$$\frac{\partial V}{\partial t} + \mathcal{L}_u V + Q(x, t) + u^T u = \nabla V, \quad (1.3.21)$$

where \mathcal{L}_u is the infinitesimal generator of the stochastic process specified by (1.3.20) and is defined by

$$\mathcal{L}_u V = \frac{1}{2} \text{tr} \left\{ h(x, t) h^T(x, t) \frac{\partial}{\partial x} \left(\frac{\partial V(x, t)}{\partial x} \right)^T \right\} + \left(\frac{\partial V(x, t)}{\partial x} \right)^T (f(x, t) + g(x, t)u).$$

Depending on whether $\nabla V \leq 0$ or $\nabla V \geq 0$, an upper bound \bar{V} or a lower bound \underline{V} of the optimal cost J^* are found by solving equation (1.3.21) such that $\underline{V} \leq J^* \leq \bar{V}$. Using \bar{V} (or \underline{V}) as an approximation to J^* , one can solve for a control law. This leads to the so-called suboptimal control. It was proved that such controls are stable for the infinite-time stochastic regulator optimal control problem, where the cost function is defined as

$$J(x_0, u) = \lim_{T \rightarrow \infty} \mathbb{E} \left\{ \frac{1}{T} \int_{t_0}^T (Q(x, t) + u^T u) dt : x(t_0) = x_0 \right\}.$$

The benefit of the suboptimal control is that the bound V of the optimal cost J^* can be approximated by an iterative process. Beginning from certain chosen functions u_0 and V_0 , let

$$u_i(x, t) = -\frac{1}{2} g^T(x, t) \frac{\partial V_{i-1}(x, t)}{\partial x}, \quad i = 1, 2, \dots \quad (1.3.22)$$

Then, by repeatedly applying (1.3.21) and (1.3.22), one will get a sequence of functions V_i . This sequence $\{V_i\}$ will converge to the bound \bar{V} (or \underline{V}) of the cost function J^* . Consequently, u_i will approximate the optimal control when i tends to ∞ . It is important to note that the sequences $\{V_i\}$ and $\{u_i\}$ are obtainable by computation and they approximate the optimal cost and the optimal control law, respectively.

Some further theoretical results for ADP have been obtained in [72, 73]. These works investigated the stability and optimality for some special cases of ADP. In [72, 73], Murray et al. studied the (deterministic) continuous-time affine nonlinear systems

$$\dot{x} = f(x) + g(x)u, \quad x(t_0) = x_0, \quad (1.3.23)$$

with the cost function

$$J(x, u) = \int_{t_0}^{\infty} U(x, u) dt, \quad (1.3.24)$$

where $U(x, u) = Q(x) + u^T R(x)u$, $Q(x) > 0$ for $x \neq 0$ and $Q(0) = 0$, and $R(x) > 0$ for all x . Similar to [86], an iterative procedure is proposed to find the control law as follows. For the plant (1.3.23) and the cost function (1.3.24), the HJB equation leads to the following optimal control law

$$u^*(x) = -\frac{1}{2} R^{-1}(x) g^T(x) \left[\frac{dJ^*(x)}{dx} \right]. \quad (1.3.25)$$

Applying (1.3.24) and (1.3.25) repeatedly, one will get sequences of estimations of the optimal cost function J^* and the optimal control u^* . Starting from an initial stabilizing control $v_0(x)$, for $i = 0, 1, \dots$, the approximation is given by the following iterations between value functions

$$V_{i+1}(x) = \int_t^{\infty} U(x(\tau), v_i(\tau)) d\tau$$

and control laws

$$v_{i+1}(x) = -\frac{1}{2} R^{-1}(x) g^T(x) \left[\frac{dV_{i+1}(x)}{dx} \right].$$

The following results were shown in [72, 73].

- (1) The sequence of functions $\{V_i\}$ obtained above converges to the optimal cost function J^* .
- (2) Each of the control laws v_{i+1} obtained above stabilizes the plant (1.3.23), for all $i = 0, 1, \dots$.
- (3) Each of the value functions $V_{i+1}(x)$ is a Lyapunov function of the plant, for all $i = 0, 1, \dots$.

Abu-Khalaf and Lewis [1] also studied the system (1.3.23) with the following value function

$$V(x(t)) = \int_t^\infty U(x(\tau), u(\tau)) d\tau = \int_t^\infty (x^\top(\tau) Q x(\tau) + u^\top(\tau) R x(\tau)) d\tau,$$

where Q and R are positive-definite matrices. The successive approximation to the HJB equation starts with an initial stabilizing control law $v_0(x)$. For $i = 0, 1, \dots$, the approximation is given by the following iterations between policy evaluation

$$0 = x^\top Q x + v_i^\top(x) R v_i(x) + \nabla V_i^\top(x) (f(x) + g(x) v_i(x))$$

and policy improvement

$$v_{i+1}(x) = -\frac{1}{2} R^{-1} g^\top(x) \nabla V_i(x),$$

where $\nabla V_i(x) = \partial V_i(x) / \partial x$. In [1], the above iterative approach was applied to systems (1.3.23) with saturating actuators through a modified utility function, with convergence and optimality proofs showing that $V_i \rightarrow J^*$ and $v_i \rightarrow u^*$, as $i \rightarrow \infty$.

For continuous-time optimal control problems, attempts have been going on for a long time in the quest for successive solutions to the HJB equation. Published works can date back to as early as 1967 by Leake and Liu [37]. The brief overview presented here only serves as a beginning of many more recent results [1, 32, 33, 45, 49, 51, 52, 56, 57, 62, 69, 70, 105, 107, 134–136, 140].

1.3.4 Remarks

Most of the early applications of ADP are in the areas of aircraft flight control [6, 21, 78] and missile guidance [22, 28]. Some other applications have also been reported, e.g., for ship steering [55], in power systems [99, 104], in communication networks [65, 67], in engine control [36, 50], and for locomotive planning [77]. The most successful application in industry is perhaps the fleet management and truckload operation problems as reported by Forbes Magazine [19]. Warren Powell from Princeton University has been working with Snyder International [94, 95], one of the largest freight haulers in the USA, in order to find more efficient ways to plan routes for parcels and freight. Applications reported in this book are related to

optimal control approaches in the areas of energy management in smart homes [31, 119, 120], coal gasification process [116], and water gas shift reaction system [118].

New comers to the field of ADP should first take a look at the challenging control problems listed in [4]. Interested readers should also read reference [39], especially the proposed training strategies for the critic network and the action network. There are also several good survey papers to read, e.g., [42, 43, 48, 53, 110].

Paul Werbos, who is the inventor of the backpropagation algorithm and adaptive critic designs, has often talked about brain-like intelligence. He has pointed out that “ADP may be the only approach that can achieve truly brain-like intelligence” [85, 125, 129, 131]. More and more evidence has accumulated, suggesting that optimality is an organizing principle for understanding brain intelligence [129–131]. There has been a great interest in brain research around the world in recent years. We would certainly hope ADP can make contributions to brain research in general and to brain-like intelligence in particular. On the other hand, with more and more advances in the understanding of brain-learning functions, new ADP algorithms can then be developed.

Deep reinforcement learning has been of great interests lately. With the current trends in deep learning, big data, artificial intelligence, as well as cyber-physical systems and Internet of things, we believe that ADP will have a bright future. There are still many pending issues to be solved, and most of them are related to obtaining good approximations to solutions of dynamic programming with less computation. Deep reinforcement learning is able to output control signal directly based on input images, which incorporates both the advantages of the perception of deep learning and the decision-making of reinforcement learning. This mechanism makes artificial intelligence much closer to human thinking. Combining deep learning with reinforcement learning/ADP will benefit us to construct systems with more intelligence and attain higher level of brain-like intelligence.

1.4 Related Books

There have been a few books published on the topics of reinforcement learning and adaptive dynamic programming. A quick overview of these books will be given in this section.

In their book published in 1996 [12], Bertsekas and Tsitsiklis give an overview of neuro-dynamic programming. The book draws on the theory of function approximation, iterative optimization, neural network training, and dynamic programming. It provides the background, gives a detailed introduction to dynamic programming, discusses the neural network architectures and methods for training them, and develops general convergence theorems for stochastic approximation methods as the foundation for the analysis of various neuro-dynamic programming algorithms. It aims at explaining with mathematical analysis, examples, speculative insight, and case studies, a number of computational ideas and phenomena that collectively can provide the foundation for understanding and applying the methodology of neuro-dynamic

programming. It suggests many useful methodologies to apply in neuro-dynamic programming, such as Monte Carlo simulation, online and offline temporal difference methods, Q-learning algorithms, optimistic policy iteration, Bellman error methods, approximate linear programming, approximate dynamic programming with cost-to-go function, etc. First, the theory and computational methods on discounted problems are developed. The computational methods for generalized discounted dynamic programming, including the asynchronous optimistic policy iteration and its application to game and minimax problems, constrained policy iteration, and Q-learning are provided. Then, the stochastic shortest path problems, the undiscounted problems, and the average cost per stage problems are also discussed. The policy iteration methods and the asynchronous optimistic versions for stochastic shortest path problems that involve improper policies are given. At last, the detailed descriptions of ADP on discounted models and nondiscounted models with generalizations are provided. Extensive materials on various simulation-based, approximate value, and policy iteration methods, Monte Carlo linear algebra, and new simulation techniques for multi-step methods, such as geometric and free-form sampling are highlighted.

The classical book by Sutton and Barto [98] provides a clear and simple account of the main ideas and algorithms of reinforcement learning, which is much more focused on goal-directed learning from interaction than other approaches of the machine learning field. It shows how to map situations to actions, so as to maximize a reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. Generally, actions may affect not only the immediate reward but also the next situation and all the subsequent rewards. This book was designed to be used as a text in a one-semester course and can also be used as part of a broader course on machine learning, artificial intelligence, or neural networks. The book consists of four parts: Part I focuses on the simplest aspects of reinforcement learning and the main distinguishing features. A beginning chapter is presented to introduce the reinforcement learning problem whose solution will be explored in the book. Part II presents tabular versions of all the basic solution methods under finite state space environment based on estimating action values. In this part, dynamic programming (including policy iteration and value iteration), Monte Carlo methods, and temporal difference learning are introduced in detail. It also includes the eligibility traces which unifies the latter two methods and contains the unified planning methods (such as dynamic programming and state-space search) and learning methods (such as Monte Carlo and temporal difference learning). Part III is concerned with extending the tabular methods to include various forms of approximation, such as function approximation, policy-gradient methods, and methods designed for solving off-policy learning problems. Part IV presents some of the frontiers of reinforcement learning in biology with practical applications. It shows that trial-and-error search and delayed reward are the two most important distinguishing features of reinforcement learning.

With the growing levels of sophistication in modern operations, it is vital for practitioners to understand how to approach, model, and solve complex industrial problems. The book by Powell [76] on approximate dynamic programming shows the decades of experience working in large industrial settings to develop practical

and high-quality solutions to problems that involve making decisions in the presence of uncertainty. The book integrates the disciplines of Markov design processes, mathematical programming, simulation, and statistics, to demonstrate how to successfully model and solve a wide range of real-life problems using the idea of approximate dynamic programming (ADP). It starts with a simple introduction using a discrete representation of states. The background of dynamic programming and Markov decision processes is given, and meanwhile the phenomenon of the curse of dimensionality is discussed. A detailed description on how to model a dynamic program and some important algorithmic strategies are presented next. The most important dimensions of ADP, i.e., modeling real applications, the interface with stochastic approximation methods, techniques for approximating general value functions, and a more in-depth presentation of ADP algorithms for finite- and infinite-horizon applications are provided, respectively. Several specific problems, including information acquisition and resource allocation, and algorithms that arise in this setting are introduced in the third part. The well-known exploration versus exploitation problem is proposed to discuss how to visit a state. These applications bring out the richness of ADP techniques. In summary, it models complex, high-dimensional problems in a natural and practical way; introduces and emphasizes the power of estimating a value function around the post-decision state; and presents a thorough discussion of recursive estimation. It is shown in this book that ADP is an accessible introduction to dynamic modeling and is also a valuable guide for the development of high-quality solutions to problems that exist in operations research and engineering.

The book by Busoniu, et al. [14] provides an accessible in-depth treatment of dynamic programming (DP) and reinforcement learning (RL) methods using function approximators. Even though DP and RL are methods for solving problems where actions are applied to a dynamical plant to achieve a desired goal, the former requires a model of the systems behavior while the latter does not since it works using only data obtained from the system. However, a core obstacle of them lies in that the solutions cannot be represented exactly for problems with large discrete state-action spaces or continuous spaces. As a result, compact representations relying on function approximators must be constructed. The book adopts a control-theoretic point of view, employing control-theoretical notation and terminology and choosing control systems as examples to illustrate the behavior of DP and RL algorithms. It starts with introducing the basic problems and their solutions, the representative classical algorithms, and the behavior of some algorithms via examples with discrete states and actions. Then, it gives an extensive account of DP and RL methods with function approximation, which are applicable to large- and continuous-space problems. The three major classes of algorithms, including value iteration, policy iteration, and policy search, are presented, respectively. Next, a value iteration algorithm with fuzzy approximation is discussed, and an extensive theoretical analysis of this algorithm is given to illustrate how convergence and consistency guarantees can be developed to perform approximate DP. Moreover, an algorithm for approximate policy iteration is studied and an online version is also developed, in order to emphasize the important issues of online RL. At last, a policy search approach relying on the cross-entropy method for optimization is described, which highlights the possibility to develop

techniques that scales to relatively high-dimensional state spaces, by focusing the computation on important initial states. Some experimental studies via classical control problems are included as performance verification.

The book edited by Frank Lewis and Derong Liu [40] gives an exposition of recently developed reinforcement learning (RL) and adaptive dynamic programming (ADP) techniques for decision and control for human-engineered systems. Included are single-player decision and control, multi-player games, and foundations in Markov decision processes. The first part of the book develops methods for feedback control of nonlinear systems based on RL and ADP. Reviews are given for the foundations, common misconceptions, and challenges ahead of RL and ADP. Novel structures, such as hierarchical adaptive critic, single network adaptive critic, actor-critic-identifier architecture, and robust ADP, are introduced to solve the optimal control problems. Additionally, several RL algorithms, such as value gradient learning and RL without using value and policy iterations, are employed to find the optimal controller. A constrained backpropagation approach to function approximation and a Q-learning-based method for learning unknown information about the timescale properties of dynamical systems are also studied. The second part treats learning and control in multi-agent games. The hybrid learning in stochastic games and its application in network security are studied. Moreover, online learning algorithms for dynamic games are proposed. The third part presents some ideas of fundamental importance in understanding and implementing decision algorithm in Markov processes. Surveys on lambda-policy iteration, event-based optimization, and optimistic planning in Markov decision processes are given. In addition, different strategies for learning policies and the backpropagation on timescales under the framework of ADP are developed. In summary, this book establishes basic methods for feedback control of modern complex systems based on RL and ADP, explains learning algorithms in multi-agent games as well as single-player games, and outlines key approaches for implementing the main algorithms in Markov decision processes.

The book by Vrabie, Vamvoudakis, and Lewis [106] presents new development in optimal adaptive control and reinforcement learning for human-engineered systems including aerospace systems, aircraft autopilots, vehicle engine controllers, ship motion and engine control, and industrial processes. It shows how to use reinforcement learning techniques to design new structures of adaptive feedback control systems that learn the solutions to optimal control problems. It also presents how to design adaptive controllers for multi-player games that converge to optimal differential game theoretic solutions online in real time by measuring data along the system trajectories. Techniques based on reinforcement learning can be used to unify optimal control and adaptive control. Reinforcement learning techniques are used to design adaptive control with novel structures that learn the solutions to optimal control problems in real time by observing data for continuous-time dynamical systems. The methods studied here depend on reinforcement learning techniques such as policy iteration and value iteration, which evaluate the performance of current control policies and provide methods for improving those policies. The adaptive learning systems utilize the actor-critic structure, wherein there are two networks in two control loops—a critic network and an actor network with parameters updated sequentially

or simultaneously. Integral reinforcement learning and synchronous tuning are combined to yield a synchronous adaptive control structure that converges to optimal control solutions without knowing the full system dynamics. Reinforcement learning methods are applied to design adaptive controllers for multi-player games that converge online to optimal game theoretic solutions. This book presents synchronous adaptive controllers that learn in real time the Nash equilibrium solution of two-player zero-sum differential games and multi-player nonzero-sum differential games. Integral reinforcement learning methods are used to learn the solution to the two-player zero-sum games online without knowing the system drift dynamics.

The book by Zhang, Liu, Luo, and Wang [139] studies the control algorithms and stability issues of adaptive dynamic programming (ADP). ADP is a biologically inspired and computational method proposed to solve optimization and optimal control problems. It is an efficient scheme to learn to approximate optimal strategy of action in the general case. With the development of ADP algorithms, more and more people show interest in the convergence of the control algorithms and stability of ADP-based systems. This book includes three parts, i.e., the optimal feedback control, the nonlinear games, and some applications of ADP. In the first part, various recent results on ADP-based infinite-horizon and finite-horizon feedback control, including stabilization and tracking problems, are presented in a systematic fashion with convergence analysis, mainly for nonlinear discrete-time systems. In addition, optimal feedback control and tracking control for nonlinear systems with time delays are studied. The input-/output-data-based optimal robust feedback control strategy for unknown general nonlinear continuous-time systems is also developed with stability proof. In addition, the optimal feedback control schemes for several special systems, such as switched systems, descriptor systems, and singularly perturbed systems, are also constructed. In the second part, both the zero-sum games and nonzero-sum games are studied using the idea of ADP. For the zero-sum games, it is proved for the first time that the iterative policies converge to the mixed optimal ones when the saddle point does not exist. For the nonzero-sum games, a single network ADP approach is established to seek for the Nash equilibrium. In the third part, a self-learning call admission control scheme is established for CDMA cellular networks, while an engine torque and air-fuel ratio control scheme is provided using ADP. In summary, the book establishes the fundamental theory of optimal control based on ADP with convergence proofs and stability analyses, and shows that ADP method can be put into use both in simulation and in real applications.

1.5 About This Book

This book covers some most recent developments in adaptive dynamic programming (ADP). After Derong Liu landed his first academic job in 1995, he was exposed to ADP at the suggestion of Paul Werbos. Within four years, he was lucky enough to receive an NSF CAREER Award, for a project on ADP for network traffic control. He started publishing papers in ADP in the same year [55], with publications ranging

from theoretical developments to applications, some of which were included in his book co-authored with Huaguang Zhang, Yanhong Luo, and Ding Wang in 2013 [139]. The present book reports some of the more recent results of Derong Liu's research group, which were mostly published after 2012.

The development of ADP has been on a fast track since 2006, when many researchers joined the effort. Within Derong Liu's research group alone, since 2012, six PhD students and one post-doctoral fellow have graduated with a few more in the pipeline. They explored new ideas of ADP and they expanded exiting theories in various dimensions. The 14 chapters in the present book cover theoretical developments in discrete-time systems and then continuous-time systems. Some application examples will be given at last. Continuing efforts are still being made in the quest for finding solutions to dynamic programming problems with manageable amount of computation and guarantee for stability, convergence, and optimality.

References

1. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791
2. Al-Tamimi A, Abu-Khalaf M, Lewis FL (2007) Adaptive critic designs for discrete-time zero-sum games with application to H_∞ control. *IEEE Trans Syst Man Cybern Part B Cybern* 37(1):240–247
3. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2008) Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):943–949
4. Anderson CW, Miller WT III (1990) Challenging control problems. In: Miller WT III, Sutton RS, Werbos PJ (eds) *Neural networks for control* (Appendix). MIT Press, Cambridge, MA
5. Bai X, Zhao D, Yi J (2009) Coordinated multiple ramps metering based on neuro-fuzzy adaptive dynamic programming. In: *Proceedings of the international joint conference on neural networks*, pp 241–248
6. Balakrishnan SN, Biega V (1996) Adaptive-critic-based neural networks for aircraft optimal control. *AIAA J Guid Control Dyn* 19:893–898
7. Barto AG (1992) Reinforcement learning and adaptive critic methods. In: White DA, Sofge DA (eds) *Handbook of intelligent control: neural, fuzzy, and adaptive approaches* (chapter 12). Van Nostrand Reinhold, New York
8. Baudis P, Gailly JL (2012) PACHI: state of the art open source Go program. In: *Advances in computer games* (Lecture notes in computer science), vol 7168. pp 24–38
9. Bellman RE (1957) *Dynamic programming*. Princeton University Press, Princeton, NJ
10. Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. *IEEE Trans Pattern Anal Mach Intell* 35(8):1798–1828
11. Bertsekas DP (2005) *Dynamic programming and optimal control*. Athena Scientific, Belmont, MA
12. Bertsekas DP, Tsitsiklis JN (1996) *Neuro-dynamic programming*. Athena Scientific, Belmont, MA
13. Buro M (1998) From simple features to sophisticated evaluation functions. In: *Proceedings of the international conference on computers and games* (Lecture notes in computer science), vol 1558. pp 126–145
14. Busoniu L, Babuska R, De Schutter B, Ernst D (2010) *Reinforcement learning and dynamic programming using function approximators*. CRC Press, Boca Raton, FL

15. Cai X, Wunsch DC (2001) A parallel computer-Go player, using HDP method. In: Proceedings of the international joint conference on neural networks, pp 2373–2375
16. Campbell M, Hoane AJ, Hsu FH (2002) Deep blue. *Artif Intell* 134(1–2):57–83
17. Chen XW, Lin X (2014) Big data deep learning: challenges and perspectives. *IEEE Access* 2:514–525
18. Clark C, Storkey AJ (2015) Training deep convolutional neural networks to play Go. In: Proceedings of the international conference on machine learning, pp 1766–1774
19. Coster H (2011) Schneider National uses data to survive a bumpy economy. *Forbes*, 12 Sept 2011
20. Coulom R (2007) Computing Elo ratings of move patterns in the game of Go. *ICGA J* 30(4):198–208
21. Cox C, Stepniewski S, Jorgensen C, Saeks R, Lewis C (1999) On the design of a neural network autolander. *Int J Robust Nonlinear Control* 9:1071–1096
22. Dalton J, Balakrishnan SN (1996) A neighboring optimal adaptive critic for missile guidance. *Math Comput Model* 23:175–188
23. Dreyfus SE, Law AM (1977) The art and theory of dynamic programming. Academic Press, New York
24. Enzenberger M (2004) Evaluation in Go by a neural network using soft segmentation. In: Advances in computer games - many games, many challenges (Proceedings of the advances in computer games conference), pp 97–108
25. Fu ZP, Zhang YN, Hou HY (2014) Survey of deep learning in face recognition. In: Proceedings of the IEEE international conference on orange technologies, pp 5–8
26. Ghesu FC, Krubasik E, Georgescu B, Singh V, Zheng Y, Hornegger J, Comaniciu D (2016) Marginal space deep learning: efficient architecture for volumetric image parsing. *IEEE Trans Med Imaging* 35(5):1217–1228
27. Gosavi A (2009) Reinforcement learning: a tutorial survey and recent advances. *INFORMS J Comput* 21(2):178–192
28. Han D, Balakrishnan SN (2002) State-constrained agile missile control with adaptive-critic-based neural networks. *IEEE Trans Control Syst Technol* 10(4):481–489
29. Haykin S (2009) Neural networks and learning machines, 3rd edn. Prentice-Hall, Upper Saddle River, NJ
30. He H, Ni Z, Fu J (2012) A three-network architecture for on-line learning and optimization based on adaptive dynamic programming. *Neurocomputing* 78(1):3–13
31. Huang T, Liu D (2013) A self-learning scheme for residential energy system control and management. *Neural Comput Appl* 22(2):259–269
32. Jiang Y, Jiang ZP (2012) Robust adaptive dynamic programming for large-scale systems with an application to multimachine power systems. *IEEE Trans Circuits Syst II: Express Briefs* 59(10):693–697
33. Jiang Y, Jiang ZP (2013) Robust adaptive dynamic programming with an application to power systems. *IEEE Trans Neural Netw Learn Syst* 24(7):1150–1156
34. Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: a survey. *J Artif Intell Res* 4:237–285
35. Konoplich GV, Putin EO, Filchenkov AA (2016) Application of deep learning to the problem of vehicle detection in UAV images. In: Proceedings of the IEEE international conference on soft computing and measurements, pp 4–6
36. Kulkarni NV, KrishnaKumar K (2003) Intelligent engine control using an adaptive critic. *IEEE Trans Control Syst Technol* 11:164–173
37. Leake RJ, Liu RW (1967) Construction of suboptimal control sequences. *SIAM J Control* 5(1):54–63
38. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521:436–444
39. Lendaris GG, Paintz C (1997) Training strategies for critic and action neural networks in dual heuristic programming method. In: Proceedings of the IEEE international conference on neural networks, pp 712–717

40. Lewis FL, Liu D (2012) Reinforcement learning and approximate dynamic programming for feedback control. Wiley, Hoboken, NJ
41. Lewis FL, Syrmos VL (1995) Optimal control. Wiley, New York
42. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Systems Mag* 9(3):32–50
43. Lewis FL, Vrabie D, Vamvoudakis KG (2012) Reinforcement learning and feedback control: Using natural decision methods to design optimla adaptive controllers. *IEEE Control Syst Mag* 32(6):76–105
44. Li H, Liu D (2012) Optimal control for discrete-time affine non-linear systems using general value iteration. *IET Control Theory Appl* 6(18):2725–2736
45. Li H, Liu D, Wang D (2014) Integral reinforcement learning for linear continuous-time zero-sum games with completely unknown dynamics. *IEEE Trans Autom Sci Eng* 11(3):706–714
46. Lincoln B, Rantzer A (2006) Relaxing dynamic programming. *IEEE Trans Autom Control* 51(8):1249–1260
47. Littman ML (2015) Reinforcement learning improves behaviour from evaluative feedback. *Nature* 521:445–451
48. Liu D (2005) Approximate dynamic programming for self-learning control. *Acta Autom Sin* 31(1):13–18
49. Liu D, Huang Y, Wang D, Wei Q (2013) Neural-network-observer-based optimal control for unknown nonlinear systems using adaptive dynamic programming. *Int J Control* 86(9):1554–1566
50. Liu D, Javaherian H, Kovalenko O, Huang T (2008) Adaptive critic learning techniques for engine torque and air-fuel ratio control. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):988–993
51. Liu D, Li C, Li H, Wang D, Ma H (2015) Neural-network-based decentralized control of continuous-time nonlinear interconnected sytems with unknown dynamics. *Neurocomputing* 165:90–98
52. Liu D, Li H, Wang D (2014) Online synchronous approximate optimal learning algorithm for multiplayer nonzero-sum games with unknown dynamics. *IEEE Trans Syst Man Cybern Syst* 44(8):1015–1027
53. Liu D, Li H, Wang D (2013) Data-based self-learning optimal control: research progress and prospects. *Acta Autom Sin* 39(11):1858–1870
54. Liu D, Li H, Wang D (2015) Error bounds of adaptive dynamic programming algorithms for solving undiscounted optimal control problems. *IEEE Trans Neural Netw Learn Syst* 26(6):1323–1334
55. Liu D, Patino HD (1999) A self-learning ship steering controller based on adaptive critic designs. In: Proceedings of the IFAC triennial world congress, pp 367–372
56. Liu D, Wang D, Li H (2014) Decentralized stabilization for a class of continuous-time nonlinear interconnected systems using online learning optimal control approach. *IEEE Trans Neural Netw Learn Syst* 25(2):418–428
57. Liu D, Wang D, Wang FY, Li H, Yang X (2014) Neural-network-based online HJB solution for optimal robust guaranteed cost control of continuous-time uncertain nonlinear systems. *IEEE Trans Cybern* 44(12):2834–2847
58. Liu D, Wang D, Yang X (2013) An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs. *Inf Sci* 220:331–342
59. Liu D, Wang D, Zhao D, Wei Q, Jin N (2012) Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming. *IEEE Trans Autom Sci Eng* 9(3):628–634
60. Liu D, Wei Q (2013) Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems. *IEEE Trans Cybern* 43(2):779–789
61. Liu D, Wei Q (2014) Policy iterative adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Trans Neural Netw Learn Syst* 25(3):621–634

62. Liu D, Wei Q (2014) Multi-person zero-sum differential games for a class of uncertain nonlinear systems. *Int J Adapt Control Signal Process* 28(3–5):205–231
63. Liu D, Wei Q, Yan P (2015) Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems. *IEEE Trans Syst Man Cybern Syst* 45(12):1577–1591
64. Liu D, Xiong X, Zhang Y (2001) Action-dependent adaptive critic designs. In: Proceedings of the international joint conference on neural networks, pp 990–995
65. Liu D, Zhang Y, Zhang H (2005) A self-learning call admission control scheme for CDMA cellular networks. *IEEE Trans Neural Netw* 16(5):1219–1228
66. Maddison CJ, Huang A, Sutskever I, Silver D (2015) Move evaluation in Go using deep convolutional neural networks. In: The 3rd international conference on learning representations. <http://arxiv.org/abs/1412.6564>
67. Marbach P, Mihatsch O, Tsitsiklis JN (2000) Call admission control and routing in integrated service networks using neuro-dynamic programming. *IEEE J Sel Areas Commun* 18(2):197–208
68. Minh V, Kavukcuoglu Silver D et al (2015) Human-level control through deep reinforcement learning. *Nature* 518:529–533
69. Modares H, Lewis FL, Naghibi-Sistani MB (2013) Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks. *IEEE Trans Neural Netw Learn Syst* 24(10):1513–1525
70. Modares H, Lewis FL, Naghibi-Sistani MB (2014) Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. *Automatica* 50:193–202
71. Moyer C (2016) How Google's AlphaGo beat a Go world champion. *The Atlantic*, 28 Mar 2016
72. Murray JJ, Cox CJ, Lendaris GG, Saeks R (2002) Adaptive dynamic programming. *IEEE Trans Syst Man Cybern Part C Appl Rev* 32(2):140–153
73. Murray JJ, Cox CJ, Saeks RE (2003) The adaptive dynamic programming theorem. In: Liu D, Antsaklis PJ (eds) *Stability and control of dynamical systems with applications* (chapter 19). Birkhäuser, Boston
74. Nguyen HD, Le AD, Nakagawa M (2015) Deep neural networks for recognizing online handwritten mathematical symbols. In: Proceedings of the IAPR Asian conference on pattern recognition, pp 121–125
75. Padhi R, Unnikrishnan N, Wang X, Balakrishnan SN (2006) A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems. *Neural Netw* 19(10):1648–1660
76. Powell WB (2007) *Approximate dynamic programming: solving the curses of dimensionality*. Wiley, Hoboken, NJ
77. Powell WB, Bouzaiene-Ayari B, Lawrence C et al (2014) Locomotive planning at Norfolk Southern: an optimizing simulator using approximate dynamic programming. *Interfaces* 44(6):567–578
78. Prokhorov DV, Santiago RA, Wunsch DC (1995) Adaptive critic designs: a case study for neurocontrol. *Neural Netw* 8:1367–1372
79. Prokhorov DV, Wunsch DC (1997) Adaptive critic designs. *IEEE Trans Neural Netw* 8:997–1007
80. Qiu J, Wu Q, Ding G, Xu Y, Feng S (2016) A survey of machine learning for big data processing. *EURASIP J Adv Signal Process*. doi:[10.1186/s13634-016-0355-x](https://doi.org/10.1186/s13634-016-0355-x)
81. Rantzer A (2006) Relaxed dynamic programming in switching systems. *IEE Proc Control Theory Appl* 153(5):567–574
82. Rummery GA, Niranjan M (1994) On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University, UK
83. Saeks RE, Cox CJ, Mathia K, Maren AJ (1997) Asymptotic dynamic programming: preliminary concepts and results. In: Proceedings of the IEEE international conference on neural networks, pp 2273–2278

84. Sahoo A, Xu H, Jagannathan S (2016) Near optimal event-triggered control of nonlinear discrete-time systems using neurodynamic programming. *IEEE Trans Neural Netw Learn Syst* 27(9):1801–1815
85. Santiago RA, Werbos PJ (1994) New progress towards truly brain-like intelligent control. In: *Proceedings of the world congress on neural networks*, vol I. pp 27–33
86. Saridis GN, Wang FY (1994) Suboptimal control of nonlinear stochastic systems. *Control Theory Adv Technol* 10(4):847–871
87. Schaeffer J, Culberson J, Treloar N, Knight B, Lu P, Szafron D (1992) A world championship caliber checkers program. *Artif Intell* 53(2–3):273–289
88. Schmidhuber J (2015) Deep learning in neural networks: an overview. *Neural Netw* 61:85–117
89. Schraudolph NN, Dayan P, Sejnowski TJ (1994) Temporal difference learning of position evaluation in the game of Go. In: *Advances in neural information processing systems 6 (NIPS 1993)*, pp 817–824
90. Si J, Barto AG, Powell WB, Wunsch DC (2004) *Handbook of learning and approximate dynamic programming*. IEEE, Piscataway, NJ
91. Si J, Wang YT (2001) On-line learning control by association and reinforcement. *IEEE Trans Neural Net* 12(3):264–276
92. Silver D, Huang A, Maddison CJ et al (2016) Mastering the game of Go with deep neural networks and tree search. *Nature* 529:484–489
93. Silver D, Sutton R, Müller M (2012) Temporal-difference search in computer Go. *Machine Learning* 87(2):183–219
94. Simao HP, Day J, George AP, et al (2009) An approximate dynamic programming algorithm for large-scale fleet management: a case application. *Transportation Science* 43(2):178–197
95. Simao HP, George Am Powell WB et al (2010) Approximate dynamic programming captures fleet operations for Schneider National. *Interfaces* 40(5):342–352
96. Sutton RS (1996) Generalization in reinforcement learning: successful examples using sparse coarse coding. In: *Advances in neural information processing systems 8 (NIPS 1995)*, pp 1038–1044
97. Sutton RS (1998) Learning to predict by the methods of temporal differences. *Mach Learn* 3:9–44
98. Sutton RS, Barto AG (1998) *Reinforcement learning: an introduction*. MIT Press, Cambridge
99. Tang Y, He H, Wen J, Liu J (2015) Power system stability control for a wind farm based on adaptive dynamic programming. *IEEE Trans Smart Grid* 6(1):166–177
100. Tesauro GJ (1992) Practical issues in temporal difference learning. *Mach Learn* 8:257–277
101. Tesauro G (1994) TD-gammon, self-teaching backgammon program, achieves master-level play. *Neural Comput* 6:215–219
102. Tian YD (2016) A simple analysis of AlphaGo. *Acta Automa Sin* 42(5):671–675
103. Tromp J (2016) Number of legal Go positions. <http://tromp.github.io/> <http://tromp.github.io/go.html>
104. Venayagamoorthy GK, Harley RG, Wunsch DC (2002) Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbo-generator. *IEEE Trans Neural Netw* 13(5):764–773
105. Vrabie D, Pastravanu O, Abu-Khalaf M, Lewis FL (2009) Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica* 45(2):477–484
106. Vrabie D, Vamvoudakis KG, Lewis FL (2013) Optimal adaptive control and differential games by reinforcement learning principles. IET, London
107. Wang D, Liu D, Li H (2014) Policy iteration algorithm for online design of robust control of a class of continuous-time nonlinear systems. *IEEE Trans Automa Sci Eng* 11(2):627–632
108. Wang D, Liu D, Wei Q, Zhao D, Jin N (2012) Optimal control of unknown nonlinear discrete-time systems based on adaptive dynamic programming approach. *Automatica* 48(8):1825–1832
109. Wang FY, Jin N, Liu D, Wei Q (2011) Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε -error bound. *IEEE Trans Neural Netw* 22(1):24–36

110. Wang FY, Zhang H, D. Liu D, (2009) Adaptive dynamic programming: an introduction. *IEEE Comput Intell Mag* 4(2):39–47
111. Wang FY, Zhang JJ, Zheng X et al (2016) Where does AlphaGo go: from Church-Turing thesis to AlphaGo thesis and beyond. *IEEE/CAA J Autom Sin* 3(2):113–120
112. Watkins CJCH (1989) Learning from delayed rewards. Ph.D. Thesis, Cambridge University, UK
113. Watkins CJCH, Dayan P (1992) Q-learning. *Mach Learn* 8:279–292
114. Wei Q, Liu D (2013) Numerical adaptive learning control scheme for discrete-time non-linear systems. *IET Control Theory Appl* 7(11):1472–1486
115. Wei Q, Liu D (2014) A novel iterative θ -adaptive dynamic programming for discrete-time nonlinear systems. *IEEE Trans Autom Sci Eng* 11(4):1176–1190
116. Wei Q, Liu D (2014) Adaptive dynamic programming for optimal tracking control of unknown nonlinear systems with application to coal gasification. *IEEE Trans Autom Sci Eng* 11(4):1020–1036
117. Wei Q, Liu D (2014) Stable iterative adaptive dynamic programming algorithm with approximation errors for discrete-time nonlinear systems. *Neural Comput Appl* 24(6):1355–1367
118. Wei Q, Liu D (2014) Data-driven neuro-optimal temperature control of water-gas shift reaction using stable iterative adaptive dynamic programming. *IEEE Trans Ind Electron* 61(11):6399–6408
119. Wei Q, Liu D, Shi G (2015) A novel dual iterative Q-learning method for optimal battery management in smart residential environments. *IEEE Trans Ind Electron* 62(4):2509–2518
120. Wei Q, Liu D, Shi G, Liu Y (2015) Multibattery optimal coordination control for home energy management systems via distributed iterative adaptive dynamic programming. *IEEE Trans Ind Electron* 62(7):4203–4214
121. Wei Q, Liu D, Xu Y (2014) Neuro-optimal tracking control for a class of discrete-time nonlinear systems via generalized value iteration adaptive dynamic programming. *Soft Comput* 20(2):697–706
122. Wei Q, Liu D, Yang X (2015) Infinite horizon self-learning optimal control of nonaffine discrete-time nonlinear systems. *IEEE Trans Neural Netw Learn Syst* 26(4):866–879
123. Wei Q, Wang FY, Liu D, Yang X (2014) Finite-approximation-error-based discrete-time iterative adaptive dynamic programming. *IEEE Trans Cybern* 44(12):2820–2833
124. Werbos PJ (1977) Advanced forecasting methods for global crisis warning and models of intelligence. *Gen Syst Yearb* 22:25–38
125. Werbos PJ (1987) Building and understanding adaptive systems: a statistical/numerical approach to factory automation and brain research. *IEEE Trans Syst Man Cybern SMC* 17(1):7–20
126. Werbos PJ (1990) Consistency of HDP applied to a simple reinforcement learning problem. *Neural Netw* 3:179–189
127. Werbos PJ (1990) A menu of designs for reinforcement learning over time. In: Miller WT, Sutton RS, Werbos PJ (eds) *Neural networks for control* (chapter 3). MIT Press, Cambridge, MA
128. Werbos PJ (1992) Approximate dynamic programming for real-time control and neural modeling. In: White DA, Sofge DA (eds) *Handbook of intelligent control: neural, fuzzy, and adaptive approaches* (chapter 13). Van Nostrand Reinhold, New York
129. Werbos PJ (2007) Using ADP to understand and replicate brain intelligence: the next level design. In: *Proceedings of the IEEE symposium on approximate dynamic programming and reinforcement learning*, pp 209–216
130. Werbos PJ (2008) ADP: the key direction for future research in intelligent control and understanding brain intelligence. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):898–900
131. Werbos PJ (2009) Intelligence in the brain: a theory of how it works and how to build it. *Neural Netw* 22(3):200–212
132. Yan P, Wang D, Li H, Liu D (2016) Error bound analysis of Q-function for discounted optimal control problems with policy iteration. *IEEE Trans Syst Man Cybern Syst*. doi:[10.1109/TSMC.2016.2563982](https://doi.org/10.1109/TSMC.2016.2563982)

133. Yang L, Enns R, Wang YT, Si J (2003) Direct neural dynamic programming. In: Liu D, Antsaklis PJ (eds) Stability and control of dynamical systems with applications (chapter 10). Birkhauser, Boston
134. Yang X, Liu D, Huang Y (2013) Neural-network-based online optimal control for uncertain non-linear continuous-time systems with control constraints. *IET Control Theory Appl* 7(17):2037–2047
135. Yang X, Liu D, Wang D (2014) Reinforcement learning for adaptive optimal control of unknown continuous-time nonlinear systems with input constraints. *Int J Control* 87(3):553–566
136. Yang X, Liu D, Wei Q (2014) Online approximate optimal control for affine non-linear systems with unknown internal dynamics using adaptive dynamic programming. *IET Control Theory Appl* 8(16):1676–1688
137. Zaman R, Prokhorov D, Wunsch DC (1997) Adaptive critic design in learning to play game of Go. In: Proceedings of the international conference on neural networks, pp 1–4
138. Zaman R, Wunsch DC (1999) TD methods applied to mixture of experts for learning 9×9 Go evaluation function. In: Proceedings of the international joint conference on neural networks, pp 3734–3739
139. Zhang H, Liu D, Luo Y, Wang D (2013) Adaptive dynamic programming for control: algorithms and stability. Springer, London
140. Zhang H, Wei Q, Liu D (2011) An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games. *Automatica* 7(1):207–214
141. Zhang H, Zhang J, Yang GH, Luo Y (2015) Leader-based optimal coordination control for the consensus problem of multiagent differential games via fuzzy adaptive dynamic programming. *IEEE Trans Fuzzy Syst* 23(1):152–163
142. Zhao DB, Shao K, Zhu YH et al (2016) Review of deep reinforcement learning and discussions on the development of computer Go. *Control Theory Appl* 33(6):701–717
143. Zhao Q, Xu H, Jagannathan S (2014) Near optimal output feedback control of nonlinear discrete-time systems based on reinforcement neural network learning. *IEEE/CAA J Automa Sin* 1(4):372–384
144. Zhong X, He H, Zhang H, Wang Z (2014) Optimal control for unknown discrete-time nonlinear markov jump systems using adaptive dynamic programming. *IEEE Trans Neural Netw Learn Syst* 25(12):2141–2155
145. Zhu Y, Zhao D, He H (2012) Integration of fuzzy controller with adaptive dynamic programming. In: Proceedings of the world congress on intelligent control and automation, pp 310–315
146. Zurada JM (1992) Introduction to artificial neural systems. West, St. Paul, MN

Part I

Discrete-Time Systems

Chapter 2

Value Iteration ADP for Discrete-Time Nonlinear Systems

2.1 Introduction

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

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

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

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

2.2 Optimal Control of Nonlinear Systems Using General Value Iteration

Consider the discrete-time nonlinear systems described by

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

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

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

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

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

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

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

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

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

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

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

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

Define the optimal cost function as

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

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

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

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

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

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

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

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

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

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

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

2.2.1 Convergence Analysis

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

and policy improvement

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

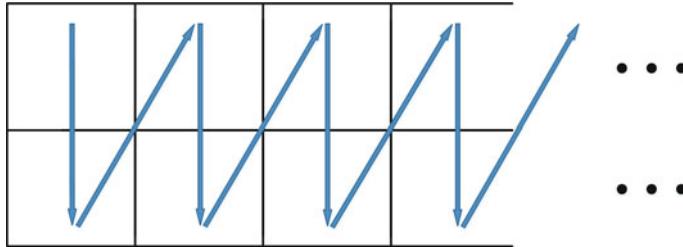
Now, from the i th iteration of the algorithm in (2.2.10)–(2.2.12), we obtain $V_i(x_k)$ and $v_i(x_k)$. Note that it is a simple calculation in (2.2.11) to update the value function using the previous policy and previous value function, while in (2.2.12), it performs the minimization so that an improved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

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

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

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

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

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

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

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

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

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

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

Lemma 2.2.1 Let μ_i be an arbitrary control law and let Λ_i be obtained by

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

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

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

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

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

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

$$\begin{cases} \Phi_1(x_k) = x_k^\top Qx_k + v_0^\top(x_k)Rv_0(x_k) + \Phi_0(f(x_k) + g(x_k)v_0(x_k)), \\ \Phi_{i+1}(x_k) = x_k^\top Qx_k + v_{i-1}^\top(x_k)Rv_{i-1}(x_k) + \Phi_i(f(x_k) + g(x_k)v_{i-1}(x_k)), \quad i \geq 1, \end{cases}$$

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

Now, we use the mathematical induction to demonstrate

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

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

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

and

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

we have

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

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

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

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

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

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

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

Similarly, we use the mathematical induction to demonstrate

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

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

According to

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

and

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

we have

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

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

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

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

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

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

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

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

Theorem 2.2.2 *Suppose the condition*

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

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

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

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

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

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

When $i = 1$, since

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

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

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

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

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

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

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

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

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

and

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

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

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

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

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

Note that larger α will lead to faster convergence speed of the value function.

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

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

Note that smaller β will lead to faster convergence speed of the value function.

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

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

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

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

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

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

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

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

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

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

Second, we have

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

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

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

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

That is,

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

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

$$v_0(x_k) = - (g^T(x_k) P_0 g(x_k) + R)^{-1} g^T(x_k) P_0 f(x_k).$$

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

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

2.2.2 Neural Network Implementation

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

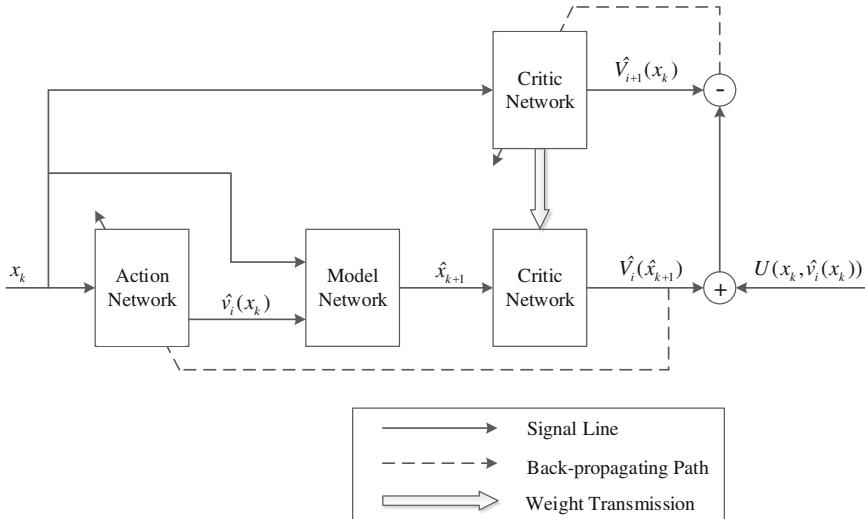


Fig. 2.2 The structural diagram of HDP algorithm

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Algorithm 2.2.1 General value iteration adaptive dynamic programming algorithm

- Step 1. Initialize the weights of critic and action neural networks and the parameters $j_{\max}^m, j_{\max}^a, j_{\max}^c, \varepsilon_m, \varepsilon_a, \varepsilon_c, i_{\max}, \xi, Q, R$.
- Step 2. Construct the model network $\hat{x}_{k+1} = W_m^T \sigma(Y_m^T \chi_k)$. Obtain the training data, and train the model network until the given accuracy ε_m or the maximum number of iterations j_{\max}^m is reached.
- Step 3. Set the iteration index $i = 0$ and $P_0 = \kappa I_n$.
- Step 4. Choose randomly an array of p state vector $\{x_k^1, x_k^2, \dots, x_k^p\}$. Compute the output of the action network $\{\hat{v}_i(x_k^1), \hat{v}_i(x_k^2), \dots, \hat{v}_i(x_k^p)\}$. Compute the output of the model network $\{\hat{x}_{k+1}^1, \hat{x}_{k+1}^2, \dots, \hat{x}_{k+1}^p\}$ and the output of the critic network $\{\hat{V}_i(\hat{x}_{k+1}^1), \hat{V}_i(\hat{x}_{k+1}^2), \dots, \hat{V}_i(\hat{x}_{k+1}^p)\}$.
- Step 5. Set the iteration index $i = i + 1$. Then, compute the target of the critic network training

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

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

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

Step 7. Compute the target of action network training

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

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

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

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

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

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

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

and stop the algorithm.

2.2.3 Generalization to Optimal Tracking Control

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Then, the optimal control law is expressed as

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

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

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

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

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

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

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

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

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

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

and policy improvement

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

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

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

Theorem 2.2.4 For $i = 0, 1, \dots$, let $V_i(z_k)$ and $v_i(z_k)$ be obtained by (2.2.21)–(2.2.24). Let ρ , γ , α , and β be constants such that

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

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

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

and

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

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

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

Proof The theorem can be proved in two steps.

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

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

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

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

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

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

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

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

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

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

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

The proof is complete.

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

Theorem 2.2.5 For $i = 0, 1, \dots$, let $v_i(z_k)$ and $V_i(z_k)$ be obtained by (2.2.21)–(2.2.24). Let ρ , γ , α , and β be constants that satisfy (2.2.25) and

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

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

Corollary 2.2.2 For $i = 0, 1, \dots$, let $v_i(z_k)$ and $V_i(z_k)$ be obtained by (2.2.21)–(2.2.24). Let ρ , γ , α , and β be constants that satisfy (2.2.25) and

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

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

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

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

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

2.2.4 Optimal Control of Systems with Constrained Inputs

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

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

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

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

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

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

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

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

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

and

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

respectively.

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

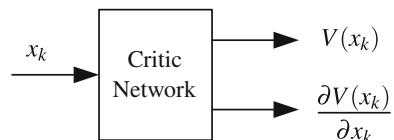
In this section, the GDHP technique is employed to implement the iterative ADP algorithm. In the iterative GDHP algorithm, there are three NNs, which are model network, critic network, and action network. Here, all the NNs are chosen as three-layer feedforward ones. It is important to note that the critic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

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

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

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

Fig. 2.3 The critic network of GDHP technique



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

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

and

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

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

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

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

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

and

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

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

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

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

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

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

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

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

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

The target function of control action is given by

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

The error function of action network can be defined as

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

The weights of action network are updated to minimize

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

Similarly, the weight update algorithm is

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

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

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

2.2.5 *Simulation Studies*

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

Example 2.2.1 Consider the linear system

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

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

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

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

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

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

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

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

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

Example 2.2.2 Consider the nonlinear system

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

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

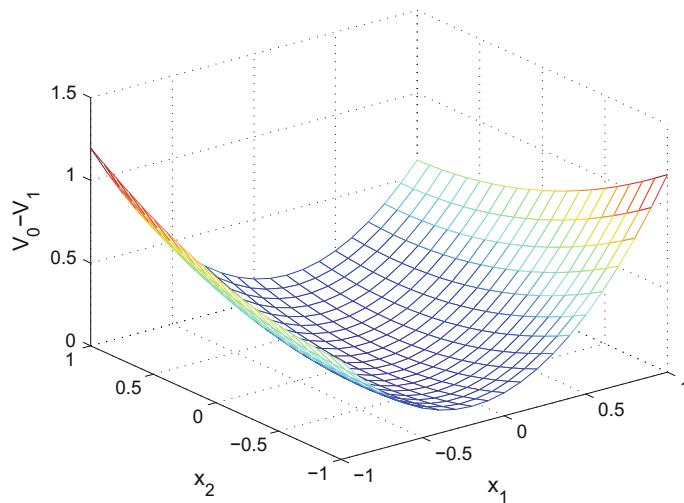


Fig. 2.4 3-D plot of $V_0 - V_1$ in the operation region

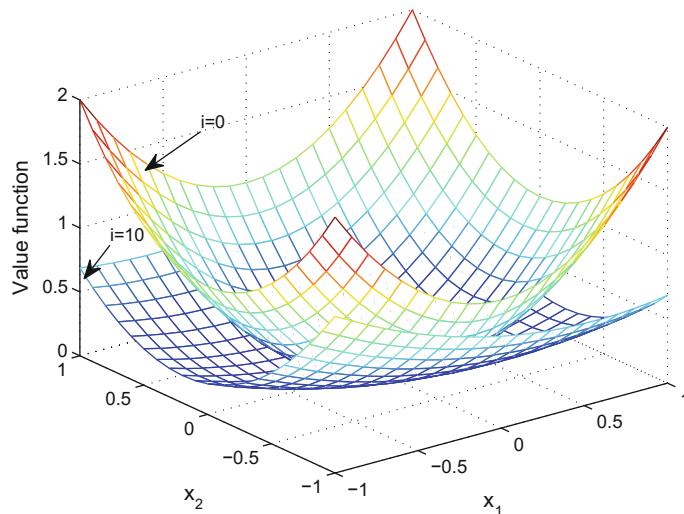


Fig. 2.5 3-D plot of approximate value function at $i = 0, 10$

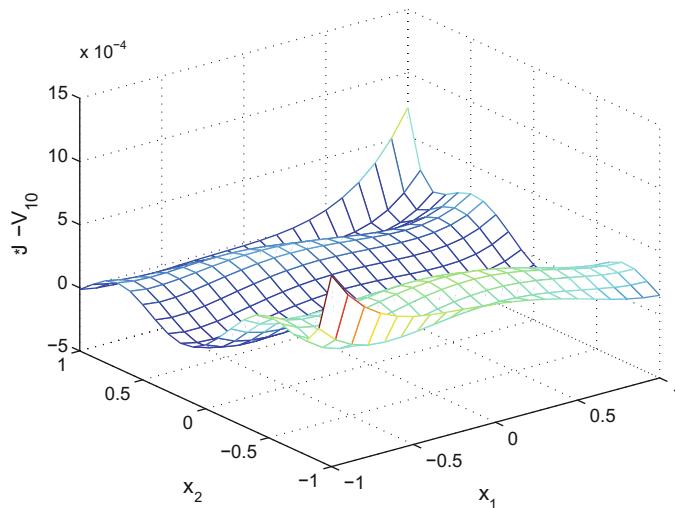


Fig. 2.6 Error between the optimal cost function J^* and the approximate optimal value function V_{10}

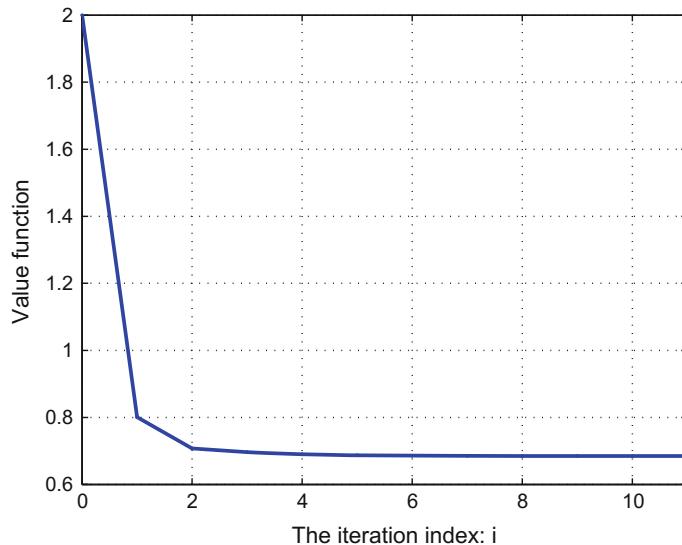


Fig. 2.7 Convergence process of the value function at $x = [1, -1]^T$

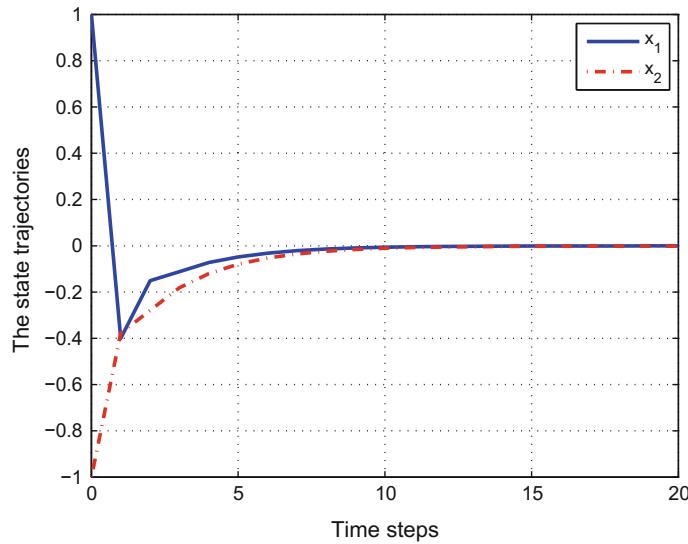


Fig. 2.8 The state trajectories

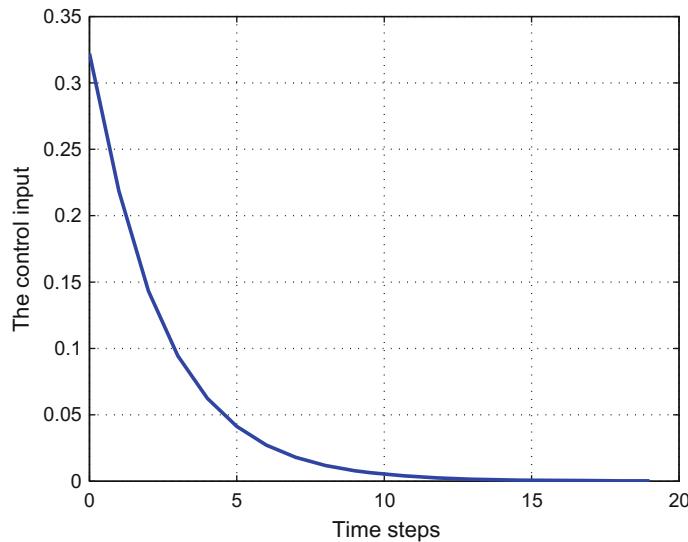


Fig. 2.9 The control input

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

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

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

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

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

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

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

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

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

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

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

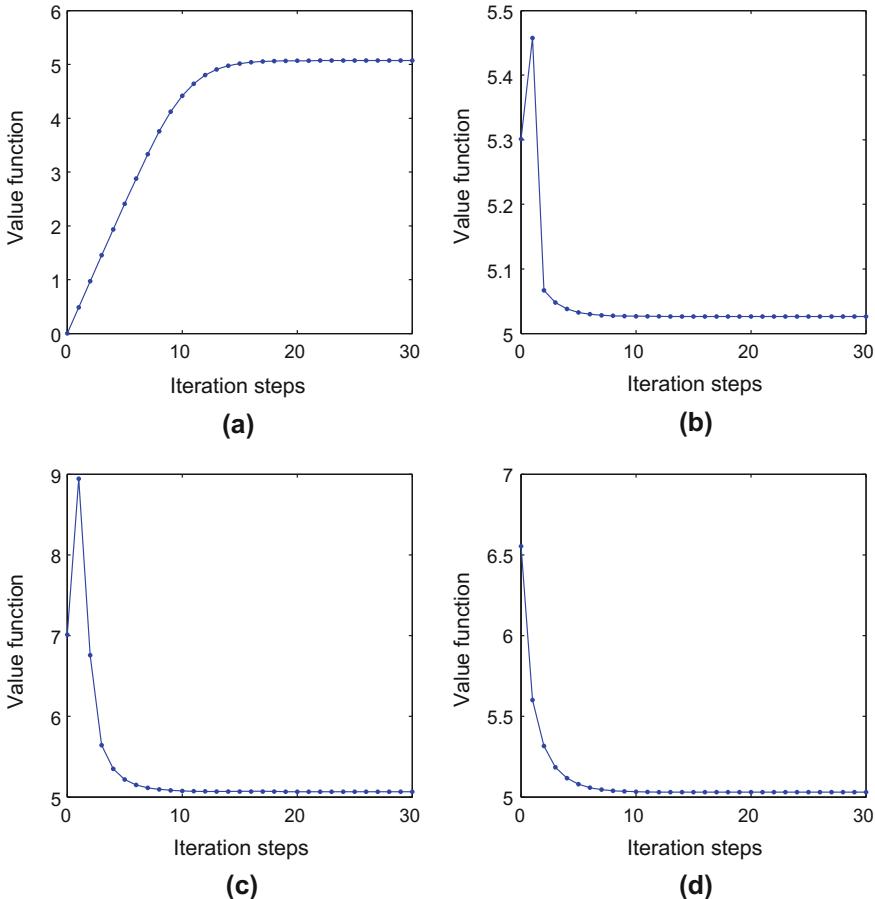


Fig. 2.10 The trajectories of the iterative value functions with initial value function given by $\Psi^j(z_k)$, $j = 1, 2, 3, 4$. **a** $\Psi^1(z_k)$. **b** $\Psi^2(z_k)$. **c** $\Psi^3(z_k)$. **d** $\Psi^4(z_k)$

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

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

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

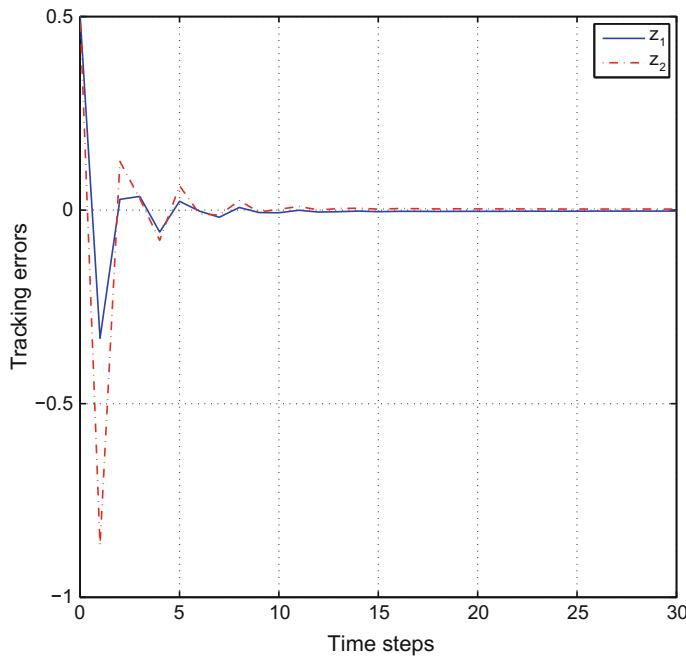


Fig. 2.11 The tracking error

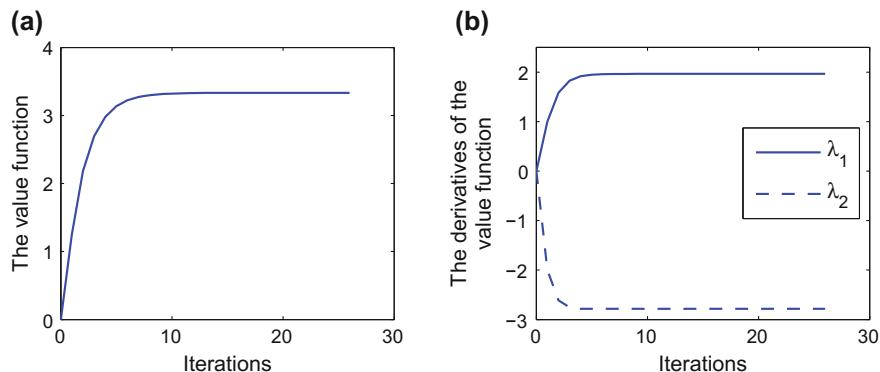


Fig. 2.12 **a** The convergence process of the value function. **b** The convergence process of the derivatives of the value function

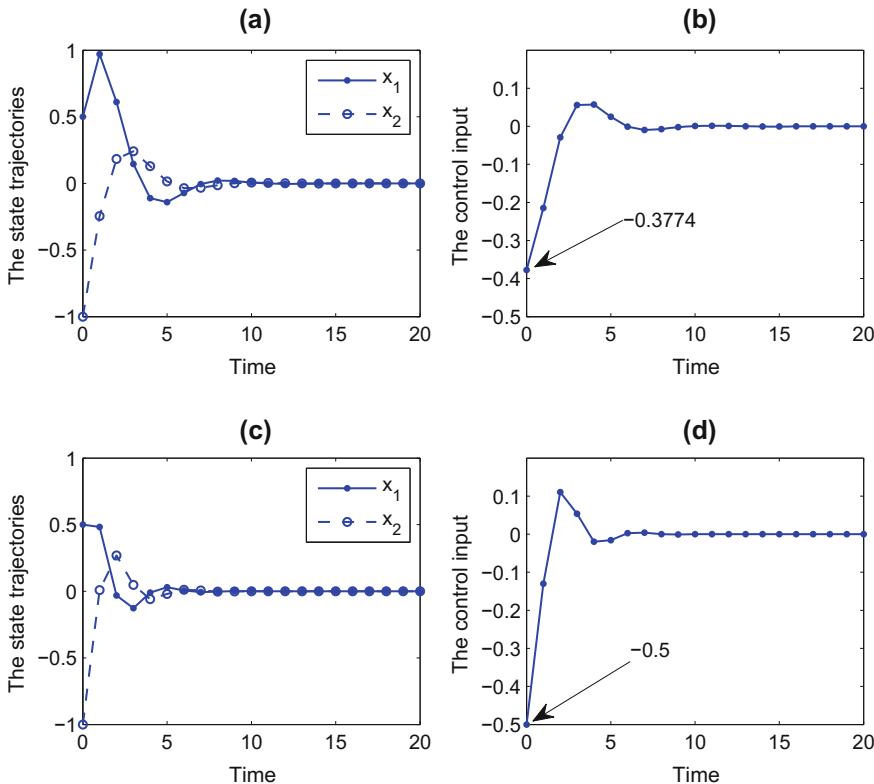


Fig. 2.13 Simulation results of Example 2.2.3. **a** The state trajectory x . **b** The control input u . **c** The state trajectory x without considering the control constraint. **d** The control input u without considering the control constraint

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

2.3 Iterative θ -Adaptive Dynamic Programming Algorithm for Nonlinear Systems

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

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

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

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

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

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

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

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

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

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

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

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

The corresponding optimal control law is given by

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

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

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

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

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

2.3.1 Convergence Analysis

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

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

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

be the set of initial positive-definite functions.

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

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

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

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

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

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

and

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

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

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

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

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

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

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

Proof According to (2.3.8), we have

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

where

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

Define

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

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

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

The proof is complete.

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

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

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

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

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

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

In general, we have

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

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

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

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

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

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

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

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

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

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

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

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

We can get

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

According to Lemma 2.3.2, we have

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

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

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

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

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

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

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

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

Let

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

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

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

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

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

Thus, there exists a finite θ_l satisfying

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

such that (2.3.12) is true.

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

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

The proof is complete.

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

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

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

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

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

Then,

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

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

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

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

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

So,

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

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

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

Then, we let

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

Hence,

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

Since ε is arbitrary, we have

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

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

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

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

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

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

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

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

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

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

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

The proof is complete.

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

2.3.2 Optimality Analysis

The following lemma is needed for the optimality analysis.

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

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

Proof According to Lemma 2.3.1, we have

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

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

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

Next, let

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

We can obtain

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

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

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

we can obtain

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The proof is complete.

We can now derive the following corollary.

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

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

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

Proof The theorem will be proved in two steps.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.3.3 Summary of Iterative θ -ADP Algorithm

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

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

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

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

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

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

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

This completes the proof of the theorem.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Now, we summarize the iterative θ -ADP algorithm as follows.

Algorithm 2.3.1 Iterative θ -adaptive dynamic programming algorithm

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

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

2.3.4 Simulation Studies

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

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

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

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

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

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

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

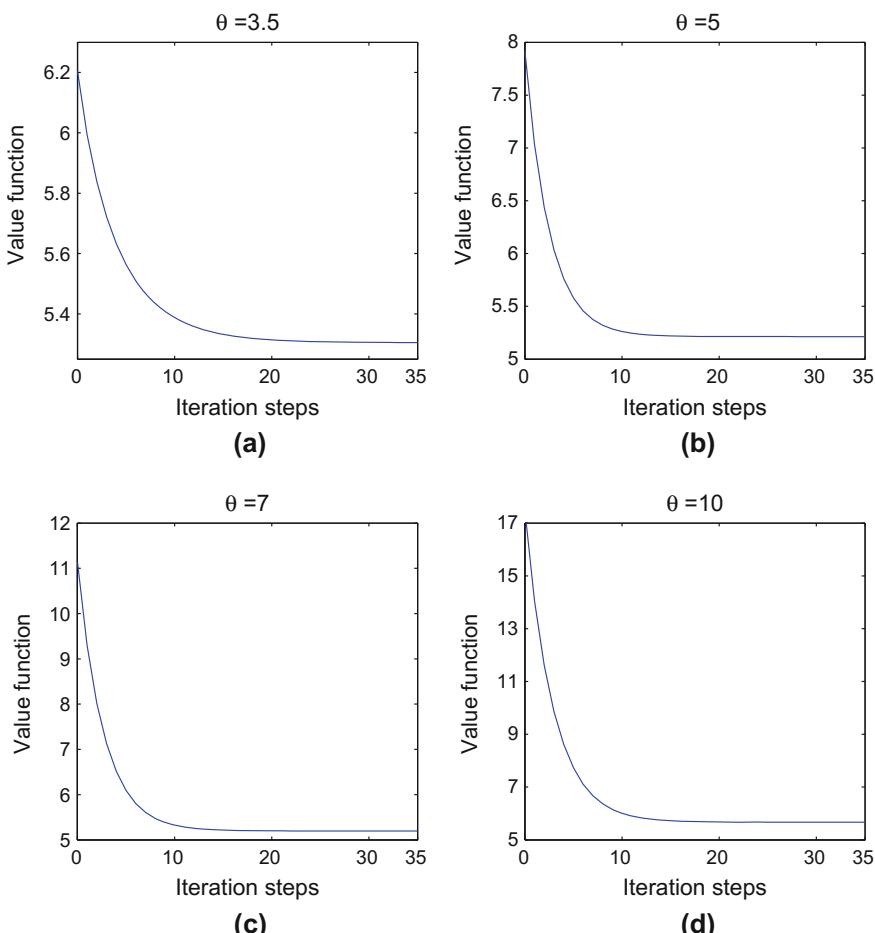


Fig. 2.14 The convergence of value functions for Example 2.3.1. **a** $\theta = 3.5$. **b** $\theta = 5$. **c** $\theta = 7$. **d** $\theta = 10$

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

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

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

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

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

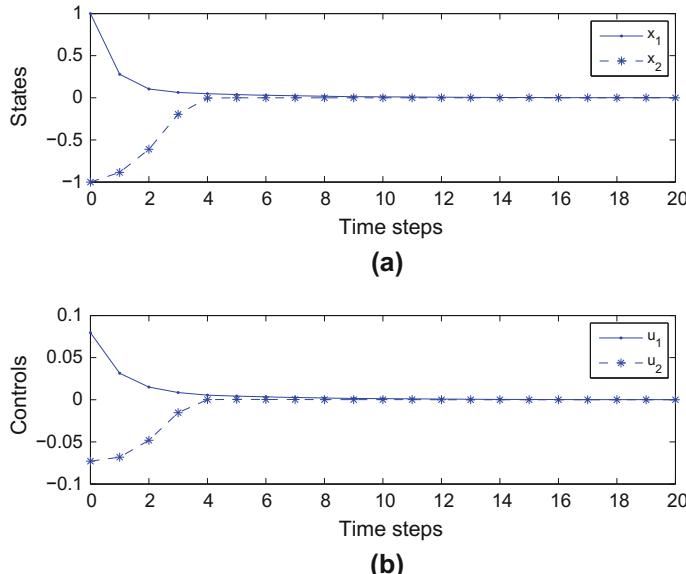


Fig. 2.15 The optimal trajectories. **a** Optimal state trajectories. **b** Optimal control trajectories

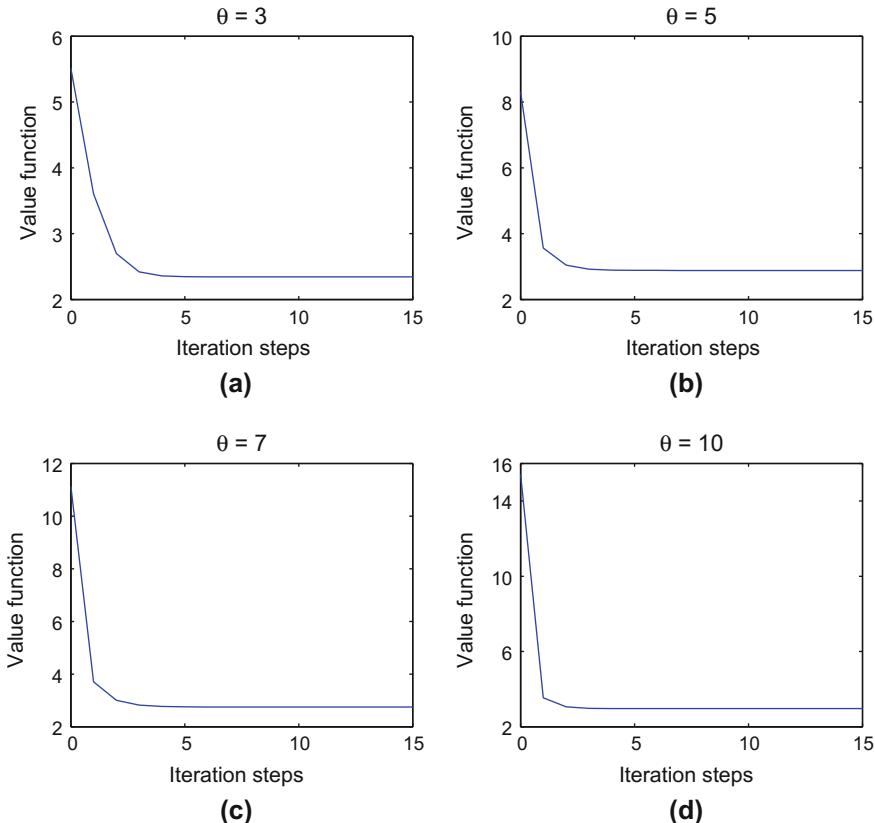


Fig. 2.16 The convergence of value functions for Example 2.3.2. **a** $\theta = 3$. **b** $\theta = 5$. **c** $\theta = 7$. **d** $\theta = 10$

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

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

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

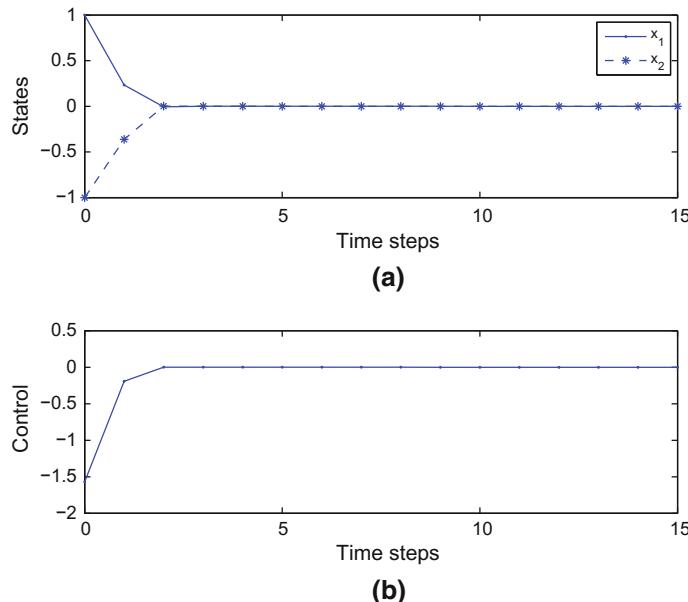


Fig. 2.17 The optimal trajectories. **a** Optimal state trajectories. **b** Optimal control trajectory

2.4 Conclusions

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

References

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

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

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

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

Chapter 3

Finite Approximation Error-Based Value Iteration ADP

3.1 Introduction

The ADP approach, proposed by Werbos [45–47], has played an important role in seeking approximate solutions to dynamic programming problems as a way to solve the optimal control problems [1, 8, 17, 19, 21, 25, 27–29, 34, 37, 40, 42, 52]. In these control strategies, iterative methods are an effective way used in ADP to obtain the solution of Bellman equation indirectly and have received lots of attentions. There are two main iterative ADP algorithms including policy iteration and value iteration algorithms [16, 31].

Although iterative ADP algorithms attract more and more attention [14, 20, 24, 30, 43, 44, 48, 53], for nearly all of the iterative algorithms, the control of each iteration is required to be accurately obtained. These iterative ADP algorithms can be called “accurate iterative ADP algorithms.” For most real-world control systems, however, the accurate control laws in the iterative ADP algorithms cannot be obtained. For example, during the implementation of the iterative ADP algorithm, approximation structures, such as neural networks and fuzzy systems, are used to approximate the iterative value functions and the iterative control laws. While we can see that no matter what kind of neural networks [12] and fuzzy systems are used, and no matter what the approximation precisions are obtained, there must exist approximation errors between the approximated functions and the expected ones. This shows that the accurate value functions and control laws cannot be obtained in the iterative ADP algorithms for real-world control systems [6, 9]. When the accurate iterative control laws cannot be obtained, the convergence properties obtained for the accurate iterative ADP algorithms may not hold anymore. It is therefore necessary to study the convergence and the stability properties of the iterative ADP algorithms when the iterative control cannot be obtained accurately. In this chapter, several new iterative ADP schemes with finite approximation errors will be developed to solve the infinite horizon optimal control problems, with convergence, stability, and optimality proofs [22, 35, 37, 39, 43].

3.2 Iterative θ -ADP Algorithm with Finite Approximation Errors

Consider the following deterministic discrete-time nonlinear systems

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \dots, \quad (3.2.1)$$

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

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

$$J(x_0, \underline{u}_0) = \sum_{k=0}^{\infty} U(x_k, u_k), \quad (3.2.2)$$

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

In this chapter, we will study optimal control problems for (3.2.1). The goal is to find an optimal control scheme which stabilizes the system (3.2.1) and simultaneously minimizes the cost function (3.2.2).

Assumptions 2.2.1–2.2.3 from Chap. 2 will be used in this chapter as well and they are restated here for convenience.

Assumption 3.2.1 $F(0, 0) = 0$, and the state feedback control law $u(\cdot)$ satisfies $u(0) = 0$, i.e., $x_k = 0$ is an equilibrium state of system (3.2.1) under the control $u_k = 0$.

Assumption 3.2.2 $F(x_k, u_k)$ is Lipschitz continuous on a compact set $\Omega \subset \mathbb{R}^n$ containing the origin.

Assumption 3.2.3 System (3.2.1) is controllable in the sense that there exists a continuous control law on Ω that asymptotically stabilizes the system.

For optimal control problems, the designed feedback control must not only stabilize the system (3.2.1) on Ω but also guarantee that the cost function (3.2.2) is finite, i.e., the control must be admissible (cf. Definition 2.2.1 or [4]). Define $\mathcal{A}(\Omega)$ as the set of admissible control sequences associated with the controllable set Ω of states. The optimal cost function is defined as

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

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

$$J^*(x_k) = \min_{u_k} \{U(x_k, u_k) + J^*(F(x_k, u_k))\}. \quad (3.2.3)$$

Define the optimal control sequence as

$$\underline{u}_k^* = \arg \inf_{\underline{u}_k} \{J(x_k, \underline{u}_k) : \underline{u}_k \in \mathcal{A}(\Omega)\}.$$

Then, the optimal control vector at time k can be expressed as

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

Hence, the Bellman equation (3.2.3) can be written as

$$J^*(x_k) = U(x_k, u_k^*) + J^*(F(x_k, u_k^*)). \quad (3.2.4)$$

We can see that if we want to obtain the optimal control u_k^* , we must obtain the optimal cost function $J^*(x_k)$. Generally speaking, $J^*(x_k)$ is unknown before the whole control sequence u_k is considered. If we adopt the traditional dynamic programming method to obtain the optimal cost function at every time step, then we have to face the “curse of dimensionality.” This implies that the optimal control sequence is nearly impossible to obtain analytically by the Bellman equation (3.2.4). Hence, ADP approaches will be investigated. In particular, in this chapter, we study ADP algorithms under a more realistic situation, where neural network approximations are not perfect as in the previous chapter.

3.2.1 Properties of the Iterative ADP Algorithm with Finite Approximation Errors

In Chap. 2, a θ -ADP algorithm is developed to obtain the optimal cost function and optimal control law iteratively. We have shown that in the iterative θ -ADP algorithm (2.3.7)–(2.3.9), the iterative value function $V_i(x_k)$ converges to the optimal cost function $J^*(x_k)$ and $J^*(x_k) = \inf_{\underline{u}_k} \{J(x_k, \underline{u}_k) : \underline{u}_k \in \mathcal{A}(\Omega)\}$, satisfying the Bellman equation (3.2.4) for any controllable state $x_k \in \mathbb{R}^n$. In fact, due to the existence of approximation errors, the accurate iterative control law cannot be obtained in general. In this case, the iterative ADP algorithm with no approximation errors may only exist in ideal situation. Besides, new analysis methods should also be developed [22, 39].

Next, we present the iterative ADP algorithm with finite approximation errors.

A. Derivation of the Iterative ADP Algorithm with Finite Approximation Errors

In the present iterative ADP algorithm, the value function and control law are updated by recurrent iteration, with the iteration index i increasing from 0 to infinity. For all $x_k \in \mathbb{R}^n$, let the initial function $\Psi(x_k)$ be an arbitrary function such that $\Psi(x_k) \in \bar{\Psi}_{x_k}$, where $\bar{\Psi}_{x_k}$ is expressed in Definition 2.3.1, i.e., $\Psi(x_k) \in \bar{\Psi}_{x_k}$ implies that $\Psi(x_k) > 0$ and there exists a stable control law $\bar{v}(x_k)$ such that $\Psi(F(x_k, \bar{v}(x_k))) <$

$\Psi(x_k)$. To avoid duplications, we mention that θ -ADP algorithm (2.3.7)–(2.3.9) will be used here starting from the initial value function $V_0(x) = \theta\Psi(x_k)$ to obtain $V_i(x_k)$ and $v_i(x_k)$ for $i = 0, 1, \dots$, where $\theta > 0$ is a finite constant that is large enough. These value functions $V_i(x_k)$ and control laws $v_i(x_k)$ are obtained assuming no approximation errors, i.e., all results in (2.3.7)–(2.3.9) are obtained accurately.

Next, we consider the same algorithm under approximation errors. For all $x_k \in \mathbb{R}^n$, let the initial value function

$$\hat{V}_0(x_k) = \theta\Psi(x_k). \quad (3.2.5)$$

The control law $\hat{v}_0(x_k)$ can be computed as

$$\begin{aligned} \hat{v}_0(x_k) &= \arg \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_0(x_{k+1}) \right\} + \rho_0(x_k) \\ &= \arg \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_0(F(x_k, u_k)) \right\} + \rho_0(x_k), \end{aligned} \quad (3.2.6)$$

where $\hat{V}_0(x_{k+1}) = \theta\Psi(x_{k+1})$ and $\rho_0(x_k)$ is the approximation error of the control law $\hat{v}_0(x_k)$. For $i = 1, 2, \dots$, the iterative ADP algorithm will iterate between

$$\begin{aligned} \hat{V}_i(x_k) &= \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_{i-1}(x_{k+1}) \right\} + \pi_i(x_k) \\ &= U(x_k, \hat{v}_{i-1}(x_k)) + \hat{V}_{i-1}(F(x_k, \hat{v}_{i-1}(x_k))) + \pi_i(x_k), \end{aligned} \quad (3.2.7)$$

and

$$\begin{aligned} \hat{v}_i(x_k) &= \arg \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_i(x_{k+1}) \right\} + \rho_i(x_k) \\ &= \arg \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_i(F(x_k, u_k)) \right\} + \rho_i(x_k), \end{aligned} \quad (3.2.8)$$

where $\pi_i(x_k)$ and $\rho_i(x_k)$ are approximation errors of the iterative value functions and the iterative control laws, respectively.

We are now in a position to present the following theorem.

Theorem 3.2.1 *Let $x_k \in \mathbb{R}^n$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. For $i = 1, 2, \dots$, define new iterative value functions as*

$$\Gamma_i(x_k) = \min_{u_k} \{U(x_k, u_k) + \hat{V}_{i-1}(x_{k+1})\}, \quad (3.2.9)$$

where $\hat{V}_i(x_k)$ is defined in (3.2.7) and u_k can accurately be obtained in \mathbb{R}^m . Let the initial value function $\Gamma_0(x_k) = \hat{V}_0(x_k) = \theta\Psi(x_k)$. If for $i = 1, 2, \dots$, there exists a finite constant $\bar{\varepsilon} \geq 0$ such that

$$|\hat{V}_i(x_k) - \Gamma_i(x_k)| \leq \bar{\varepsilon} \quad (3.2.10)$$

holds uniformly. Then, we have

$$|\hat{V}_i(x_k) - V_i(x_k)| \leq i\bar{\varepsilon}. \quad (3.2.11)$$

Proof The theorem can be proved by mathematical induction. First, let $i = 1$. We have

$$\begin{aligned} \Gamma_1(x_k) &= \min_{u_k} \{U(x_k, u_k) + \hat{V}_0(x_{k+1})\} \\ &= \min_{u_k} \{U(x_k, u_k) + V_0(F(x_k, u_k))\} = V_1(x_k). \end{aligned}$$

According to (3.2.10), we have

$$-\bar{\varepsilon} \leq \hat{V}_i(x_k) - \Gamma_i(x_k) \leq \bar{\varepsilon} \quad (3.2.12)$$

for $i = 1, 2, \dots$. Then, we can get $-\bar{\varepsilon} \leq \hat{V}_1(x_k) - V_1(x_k) \leq \bar{\varepsilon}$, which proves $|\hat{V}_1(x_k) - V_1(x_k)| \leq \bar{\varepsilon}$. Assume that (3.2.11) holds for $i = l-1, l = 2, 3, \dots$. Then,

$$-(l-1)\bar{\varepsilon} \leq \hat{V}_{l-1}(x_k) - V_{l-1}(x_k) \leq (l-1)\bar{\varepsilon}. \quad (3.2.13)$$

For $i = l$, considering (3.2.13), we can get

$$\begin{aligned} \Gamma_l(x_k) &= \min_{u_k} \{U(x_k, u_k) + \hat{V}_{l-1}(x_{k+1})\} \\ &\geq \min_{u_k} \{U(x_k, u_k) + V_{l-1}(x_{k+1}) - (l-1)\bar{\varepsilon}\} \\ &= V_l(x_k) - (l-1)\bar{\varepsilon}, \end{aligned}$$

and

$$\begin{aligned} \Gamma_l(x_k) &= \min_{u_k} \{U(x_k, u_k) + \hat{V}_{l-1}(x_{k+1})\} \\ &\leq \min_{u_k} \{U(x_k, u_k) + V_{l-1}(x_{k+1}) + (l-1)\bar{\varepsilon}\} \\ &= V_l(x_k) + (l-1)\bar{\varepsilon}. \end{aligned}$$

Then, using (3.2.12) and similarly considering the left side of (3.2.13), we can get (3.2.11). This completes the proof.

From Theorem 3.2.1, we can see that if we let a finite constant $-\bar{\varepsilon} \leq \varepsilon \leq \bar{\varepsilon}$ such that

$$\hat{V}_i(x_k) - \Gamma_i(x_k) \leq \varepsilon \quad (3.2.14)$$

uniformly, then we can immediately obtain

$$\hat{V}_i(x_k) - V_i(x_k) \leq i\varepsilon, \quad (3.2.15)$$

where ε can be defined as uniform finite approximation error (finite approximation error in brief). For the iterative θ -ADP algorithm (3.2.7)–(3.2.9), it has been proved that the iterative value function converges to the optimum as $i \rightarrow \infty$. From (3.2.15), we can see that if we let $\mathcal{T}_i = i\varepsilon$ be the least upper bound of the iterative approximation errors, then for all $\varepsilon \neq 0$, $\mathcal{T}_i \rightarrow \infty$, as $i \rightarrow \infty$. This means that although the approximation error for each single step is finite and may be small, as the iteration index i increases, it is possible that the approximation errors between $\hat{V}_i(x_k)$ and $V_i(x_k)$ increase to infinity. Hence, the convergence and stability analysis results in Chap. 2 [36, 38] are no longer applicable to the analysis of $\hat{V}_i(x_k)$ and $\hat{v}_i(x_k)$ in (3.2.6)–(3.2.8). To overcome these difficulties, new convergence and stability analysis must be established.

B. Properties of the Iterative ADP Algorithm with Finite Approximation Errors

From iterative ADP algorithm (3.2.6)–(3.2.8), we can see that for $i = 0, 1, \dots$, there exists an approximation error between the iterative value functions $\hat{V}_i(x_k)$ and $V_i(x_k)$. Furthermore, the value of the error at each iteration is unknown and nearly impossible to obtain. It is very difficult to analyze the properties of the iterative value function $\hat{V}_i(x_k)$ and iterative control law $\hat{v}_i(x_k)$. So, in this section, a new “error bound” analysis method is developed. The idea of the “error bound” analysis method is that for each iteration index $i = 0, 1, \dots$, the least upper bound of the iterative value functions $\hat{V}_i(x_k)$ is analyzed, which avoids to analyze the value of $\hat{V}_i(x_k)$ directly. Using the “error bound” method, it can be proved that the iterative value functions $\hat{V}_i(x_k)$ can uniformly converge to a bounded neighborhood of optimal cost function. The convergence and stability analysis will also be given in this section.

For convenience of analysis, we transform the expressions of the approximation errors. According to (3.2.14), we have $\hat{V}_i(x_k) \leq \Gamma_i(x_k) + \varepsilon$. Then, for $i = 0, 1, \dots$, there exists a finite constant $\eta > 0$ such that

$$\hat{V}_i(x_k) \leq \eta \Gamma_i(x_k) \quad (3.2.16)$$

holds uniformly. Hence, we can give the following theorem.

Theorem 3.2.2 *Let $x_k \in \mathbb{R}^n$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. For $i = 0, 1, \dots$, let $\Gamma_i(x_k)$ be expressed as in (3.2.9) and $\hat{V}_i(x_k)$ be expressed as in (3.2.7). Let $\gamma < \infty$ and $1 \leq \beta < \infty$ be constants such that*

$$J^*(F(x_k, u_k)) \leq \gamma U(x_k, u_k),$$

and

$$J^*(x_k) \leq V_0(x_k) \leq \beta J^*(x_k),$$

hold uniformly. If there exists $0 < \eta < \infty$ such that (3.2.16) holds uniformly, then

$$\hat{V}_i(x_k) \leq \eta \left(1 + \sum_{j=1}^i \frac{\gamma^j \eta^{j-1}(\eta-1)}{(\gamma+1)^j} + \frac{\gamma^i \eta^i(\beta-1)}{(\gamma+1)^i} \right) J^*(x_k), \quad (3.2.17)$$

where we define $\sum_j^i (\cdot) = 0$, for all $j > i$ and $i, j = 0, 1, \dots$

Proof The theorem can be proved by mathematical induction. First, let $i = 0$. Then, (3.2.17) becomes $\hat{V}_0(x_k) \leq \eta \beta J^*(x_k)$. As $\Gamma_0(x_k) = V_0(x_k) \leq \beta J^*(x_k)$, we can obtain $\hat{V}_0(x_k) \leq \eta \Gamma_0(x_k) \leq \eta \beta J^*(x_k)$. So, the conclusion holds for $i = 0$.

Next, let $i = 1$. By (3.2.9), we have

$$\begin{aligned} \Gamma_1(x_k) &= \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_0(F(x_k, u_k)) \right\} \\ &\leq \min_{u_k} \left\{ U(x_k, u_k) + \eta \beta J^*(F(x_k, u_k)) \right\} \\ &\leq \min_{u_k} \left\{ \left(1 + \gamma \frac{\eta \beta - 1}{\gamma + 1} \right) U(x_k, u_k) + \left(\eta \beta - \frac{\eta \beta - 1}{\gamma + 1} \right) J^*(F(x_k, u_k)) \right\} \\ &= \left(1 + \gamma \frac{\eta \beta - 1}{\gamma + 1} \right) \min_{u_k} \left\{ U(x_k, u_k) + J^*(F(x_k, u_k)) \right\} \\ &= \left(1 + \frac{\gamma(\eta-1)}{\gamma+1} + \frac{\gamma \eta(\beta-1)}{\gamma+1} \right) J^*(x_k). \end{aligned}$$

According to (3.2.16), we can obtain

$$\hat{V}_1(x_k) \leq \eta \left(1 + \frac{\gamma(\eta-1)}{\gamma+1} + \frac{\gamma \eta(\beta-1)}{\gamma+1} \right) J^*(x_k),$$

which shows that (3.2.17) holds for $i = 1$.

Assume that (3.2.17) holds for $i = l - 1$, where $l = 1, 2, \dots$. Then, for $i = l$, we have

$$\begin{aligned} \Gamma_l(x_k) &= \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_{l-1}(F(x_k, u_k)) \right\} \\ &\leq \min_{u_k} \left\{ U(x_k, u_k) + \eta \left(1 + \sum_{j=1}^{l-1} \frac{\gamma^j \eta^{j-1}(\eta-1)}{(\gamma+1)^j} + \frac{\gamma^{l-1} \eta^{l-1}(\beta-1)}{(\gamma+1)^{l-1}} \right) J^*(x_{k+1}) \right\} \end{aligned}$$

$$\begin{aligned}
&\leq \min_{u_k} \left\{ \left(1 + \gamma \sum_{j=1}^{l-1} \frac{\gamma^{j-1} \eta^{j-1} (\eta - 1)}{(\gamma + 1)^j} + \frac{\gamma^{l-1} \eta^{l-1} (\eta \beta - 1)}{(\gamma + 1)^{l-1}} \right) U(x_k, u_k) \right. \\
&\quad + \left(\eta \left[1 + \sum_{j=1}^{l-1} \frac{\gamma^j \eta^{j-1} (\eta - 1)}{(\gamma + 1)^j} + \frac{\gamma^{l-1} \eta^{l-1} (\beta - 1)}{(\gamma + 1)^{l-1}} \right] \right. \\
&\quad \left. \left. - \left[\sum_{j=1}^{l-1} \frac{\gamma^{j-1} \eta^{j-1} (\eta - 1)}{(\gamma + 1)^j} + \frac{\gamma^{l-1} \eta^{l-1} (\eta \beta - 1)}{(\gamma + 1)^{l-1}} \right] \right) J^*(F(x_k, u_k)) \right\} \\
&= \left[1 + \sum_{j=1}^l \frac{\gamma^j \eta^{j-1} (\eta - 1)}{(\gamma + 1)^j} + \frac{\gamma^l \eta^l (\beta - 1)}{(\gamma + 1)^l} \right] J^*(x_k). \tag{3.2.18}
\end{aligned}$$

Then, according to (3.2.16), we can obtain (3.2.17) which proves the conclusion for $i = 0, 1, \dots$. The proof is complete.

From (3.2.17), we can see that for arbitrary finite i , η , and β , there exists a bounded error between the iterative value function $\hat{V}_i(x_k)$ and the optimal cost function $J^*(x_k)$. While as $i \rightarrow \infty$, the bound of the approximation errors may increase to infinity. Thus, in the following theorems, we will prove the convergence properties of the iterative ADP algorithm (3.2.6)–(3.2.8) using the error bound method.

Theorem 3.2.3 *Let $x_k \in \mathbb{R}^n$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. Suppose Theorem 3.2.2 holds for all $x_k \in \mathbb{R}^n$. If for $\gamma < \infty$ and $\eta > 0$, the inequality*

$$0 < \eta < \frac{\gamma + 1}{\gamma} \tag{3.2.19}$$

holds, then as $i \rightarrow \infty$, the value function $\hat{V}_i(x_k)$ in the iterative ADP algorithm (3.2.6)–(3.2.8) is uniformly convergent into a bounded neighborhood of the optimal cost function $J^(x_k)$, i.e.,*

$$\lim_{i \rightarrow \infty} \hat{V}_i(x_k) = \hat{V}_\infty(x_k) \leq \frac{\eta}{1 - \gamma(\eta - 1)} J^*(x_k). \tag{3.2.20}$$

Proof According to (3.2.18) in Theorem 3.2.2, we can see that for $j = 1, 2, \dots$,

$$\left\{ \frac{\gamma^j \eta^{j-1} (\eta - 1)}{(\gamma + 1)^j} \right\}$$

is a geometrical sequence. Then, (3.2.18) can be written as

$$\Gamma_i(x_k) \leq \left(1 + \frac{\gamma(\eta - 1)(1 - (\gamma \eta)^i)}{1 + \gamma - \gamma \eta} + \frac{\gamma^i \eta^i (\beta - 1)}{(\gamma + 1)^i} \right) J^*(x_k). \tag{3.2.21}$$

As $i \rightarrow \infty$, if $0 < \eta < (\gamma + 1)/\gamma$, then (3.2.21) becomes

$$\lim_{i \rightarrow \infty} \Gamma_i(x_k) = \Gamma_\infty(x_k) \leq \left(1 + \frac{\gamma(\eta - 1)}{1 - \gamma(\eta - 1)}\right) J^*(x_k). \quad (3.2.22)$$

Using (3.2.16) and letting $i \rightarrow \infty$, we have

$$\hat{V}_\infty(x_k) \leq \eta \Gamma_\infty(x_k). \quad (3.2.23)$$

Combining (3.2.22) and (3.2.23), we can obtain (3.2.20). The proof is complete.

Corollary 3.2.1 *Let $x_k \in \mathbb{R}^n$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. Suppose Theorem 3.2.2 holds for all $x_k \in \mathbb{R}^n$. If for $\gamma < \infty$ and $\eta > 0$, the inequality (3.2.19) holds, then the iterative control law $\hat{v}_i(x_k)$ of the iterative ADP algorithm (3.2.6)–(3.2.8) is convergent, i.e.,*

$$\hat{v}_\infty(x_k) = \lim_{i \rightarrow \infty} \hat{v}_i(x_k).$$

For the iterative ADP algorithm with finite approximation errors, we can see that for different approximation errors, the limits of the iterative value function $\hat{V}_i(x_k)$ are different. This property can be proved by the following theorem.

Theorem 3.2.4 *Let $x_k \in \mathbb{R}^n$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. Let*

$$\bar{V}_\infty(x_k) = \eta \left(1 + \frac{\gamma(\eta - 1)}{1 - \gamma(\eta - 1)}\right) J^*(x_k) \quad (3.2.24)$$

be the least upper bound of the limit of the iterative value function. If Theorem 3.2.3 holds for all $x_k \in \mathbb{R}^n$, then $\bar{V}_\infty(x_k)$ is a monotonically increasing function of the approximation error η .

Proof From (3.2.24), we can see that the least upper bound of the limit of the iterative value function $\bar{V}_\infty(x_k)$ is a differentiable function of the approximation error η . Hence, we can take the derivative of the approximation error η on both sides of the equation (3.2.24). Then, we can obtain

$$\frac{\partial \bar{V}_\infty(x_k)}{\partial \eta} = \left(\frac{1 + \gamma}{(\gamma(\eta - 1) - 1)^2}\right) J^*(x_k) > 0.$$

The proof is complete.

According to the definitions of iterative value functions $\hat{V}_i(x_k)$ and $\Gamma_i(x_k)$ in (3.2.7) and (3.2.9), if for $i = 0, 1, \dots$, we let

$$\hat{V}_i(x_k) - \Gamma_i(x_k) = \varepsilon_i(x_k), \quad (3.2.25)$$

then, we can choose

$$\varepsilon = \sup\{\varepsilon_0(x_k), \varepsilon_1(x_k), \dots\}. \quad (3.2.26)$$

In this section, the approximation error η which satisfies (3.2.16) is used to analyze the convergence properties of the iterative ADP algorithm. Generally speaking, the approximation error ε is obtained by (3.2.14) instead of obtaining η . So, if we use ε to express the approximation error, then a transformation is needed between the two approximation errors ε and η .

For $i = 0, 1, \dots$, according to (3.2.14) and (3.2.16), for any ε_i expressed in (3.2.25), there exists a η_i such that

$$\hat{V}_i(x_k) - \varepsilon_i(x_k) = \frac{\hat{V}_i(x_k)}{\eta_i(x_k)}.$$

According to (3.2.26), we can obtain $\eta = \sup\{\eta_0(x_k), \eta_1(x_k), \dots\}$.

3.2.2 Neural Network Implementation

In the case of linear systems, the control law is linear and the cost function is quadratic. In the nonlinear case, this is not necessarily true, and therefore we use backpropagation (BP) neural networks to approximate $V_i(x_k)$ and $v_i(x_k)$.

For a given nonlinear function $\xi(x) \in \mathbb{R}^m$, a BP NN with two layers of weights can be used to approximate it. Assume that the number of hidden layer neurons is denoted by L , the weight matrix between the input layer and hidden layer is denoted by $Y_\xi \in \mathbb{R}^{L \times n}$, and the weight matrix between the hidden layer and output layer is denoted by $W_\xi \in \mathbb{R}^{m \times L}$. Then, the NN representation of ξ is given by

$$\xi(x) = W_\xi^* \sigma(Y_\xi^* x) + \varepsilon(x),$$

where $\sigma(\cdot) = \tanh(\cdot)$ is the componentwise activation function, Y_ξ^* and W_ξ^* are the ideal weight parameters of Y_ξ and W_ξ , respectively, and $\varepsilon(x)$ is the reconstruction error. The NN approximation of function ξ is expressed by

$$\hat{\xi}(x, Y_\xi, W_\xi) = W_\xi \sigma(Y_\xi x).$$

Remark 3.2.1 In Chap. 2 and several other chapters, weight matrices in the NN expressions are in transposed forms. It is a matter of convenience, for instance, to use W_ξ or W_ξ^T in NN expressions. In this chapter, to avoid cumbersome notations, we opt to use weight matrices without transposition in NN expressions.

Here, there are two networks, which are critic network and action network, respectively. Both neural networks are chosen as three-layer feedforward network. The whole structural diagram is shown in Fig. 3.1.

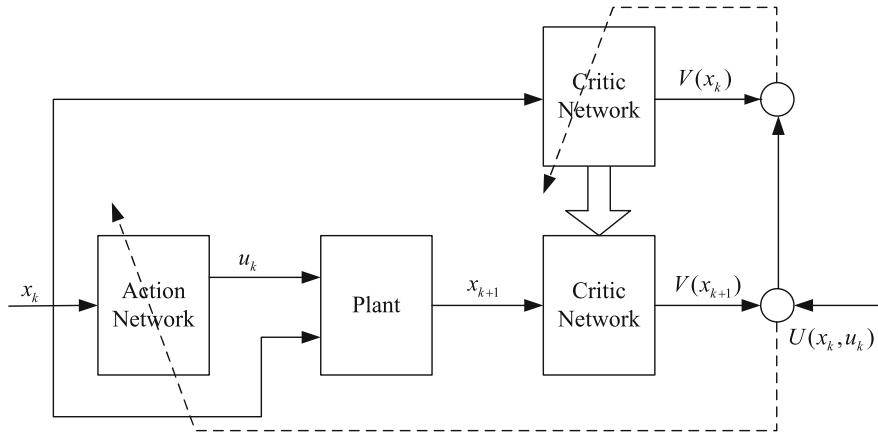


Fig. 3.1 The structural diagram of the algorithm

A. The Critic Network

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

$$\hat{V}_i(x_k) = W_c \sigma(Y_c x_k),$$

where $W_c \in \mathbb{R}^{1 \times L_c}$, $Y_c \in \mathbb{R}^{L_c \times n}$, and L_c is the number of hidden layer nodes in the critic network. The target function can be written as

$$V_i(x_k) = U(x_k, v_{i-1}(x_k)) + \hat{V}_{i-1}(x_{k+1}).$$

Then, we define the error function of the critic network training as

$$e_{ci}(k) = V_i(x_k) - \hat{V}_i(x_k).$$

The objective function to be minimized in the critic network is $E_{ci}(k) = (1/2)e_{ci}^2(k)$. So, the gradient-based weight update rule [13, 28] for the critic network is given by

$$\begin{aligned} X_c(p+1) &= X_c(p) + \Delta X_c(p), \\ \Delta X_c(p) &= \alpha_c \left[-\frac{\partial E_{ci}(k)}{\partial X_c(p)} \right], \\ \frac{\partial E_{ci}(k)}{\partial X_c(p)} &= \frac{\partial E_{ci}(k)}{\partial \hat{V}_i(x_k)} \frac{\partial \hat{V}_i(x_k)}{\partial X_c(p)}, \end{aligned}$$

where p represents the inner-loop iteration step for updating critic NN weight parameters, $\alpha_c > 0$ is the learning rate of critic network, and X_c is the weight matrix of the critic network which can be replaced by W_c and Y_c , respectively.

If we define

$$q_l(p) = \sum_{j=1}^n Y_{c,lj}(p)x_{jk}, \quad l = 1, 2, \dots, L_c,$$

$$r_l(p) = \tanh(q_l(p)), \quad l = 1, 2, \dots, L_c,$$

then

$$\hat{V}_i(x_k) = \sum_{l=1}^{L_c} W_{cl}(p)r_l(p),$$

where $x_k = (x_{1k}, \dots, x_{nk})^\top \in \mathbb{R}^n$, $Y_c = (Y_{c,lj}) \in \mathbb{R}^{L_c \times n}$, and $W_c = (W_{cl}) \in \mathbb{R}^{1 \times L_c}$. By applying the chain rule, the adaptation of the critic network is summarized as follows:

The weights of hidden-to-output layer of the critic network are updated as

$$W_{cl}(p+1) = W_{cl}(p) - \alpha_c \frac{\partial E_{ci}(k)}{\partial \hat{V}_i(x_k)} \frac{\partial \hat{V}_i(x_k)}{\partial W_{cl}(p)}$$

$$= W_{cl}(p) - \alpha_c e_{ci}(k)r_l(p), \quad l = 1, 2, \dots, L_c.$$

The weights of input to hidden layer of the critic network are updated as

$$Y_{c,lj}(p+1) = Y_{c,lj}(p) - \alpha_c \frac{\partial E_{ci}(k)}{\partial \hat{V}_i(x_k)} \frac{\partial \hat{V}_i(x_k)}{\partial Y_{c,lj}(p)}$$

$$= Y_{c,lj}(p) - \alpha_c \frac{\partial E_{ci}(k)}{\partial \hat{V}_i(x_k)} \frac{\partial \hat{V}_i(x_k)}{\partial r_l(p)} \frac{\partial r_l(p)}{\partial q_l(p)} \frac{\partial q_l(p)}{\partial Y_{c,lj}(k)}$$

$$= Y_{c,lj}(p) - \alpha_c e_{ci}(k)W_{cl}(p)(1 - r_l^2(p))x_{jk},$$

$$l = 1, 2, \dots, L_c, \quad j = 1, 2, \dots, n.$$

B. The Action Network

In the action network, the state x_k is used as input to create the optimal control law as the output of the network. The output can be formulated as

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

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

$$e_{ai}(k) = v_i(x_k) - \hat{v}_i(x_k),$$

where the target function of the action network training is given by

$$v_i(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + \hat{V}_{i-1}(x_{k+1})\}.$$

The weights of the action network are updated to minimize the following performance error measure $E_{ai}(k) = (1/2)e_{ai}^\top(k)e_{ai}(k)$. The weights updating algorithm is similar to the one for the critic network. By the gradient descent rule, we can obtain

$$X_a(p+1) = X_a(p) + \Delta X_a(p),$$

$$\Delta X_a(p) = \beta_a \left[-\frac{\partial E_{ai}(k)}{\partial X_a(p)} \right],$$

$$\frac{\partial E_{ai}(k)}{\partial X_a(p)} = \frac{\partial E_{ai}(k)}{\partial e_{ai}(k)} \frac{\partial e_{ai}(k)}{\partial \hat{v}_i(x_k)} \frac{\partial \hat{v}_i(x_k)}{\partial X_a(p)},$$

where $\beta_a > 0$ is the learning rate of action network and X_a is the weight matrix of the action network which can be replaced by W_a and Y_a .

If we define

$$g_l(p) = \sum_{j=1}^n Y_{a,lj}(p)x_{jk}, \quad l = 1, \dots, L_a,$$

$$h_l(p) = \tanh(g_l(p)), \quad l = 1, \dots, L_a,$$

then

$$\hat{v}_{it}(x_k) = \sum_{l=1}^{L_a} W_{a,tl}(p)h_l(p), \quad t = 1, 2, \dots, m,$$

where L_a is the number of hidden nodes in the action network, $\hat{v}_i(x_k) = (\hat{v}_{i1}(x_k), \dots, \hat{v}_{im}(x_k))^\top \in \mathbb{R}^m$, $Y_a = (Y_{a,lj}) \in \mathbb{R}^{L_a \times n}$, and $W_a = (W_{a,tl}) \in \mathbb{R}^{m \times L_a}$. By applying the chain rule, the adaptation of the action network is summarized as follows.

The weight of hidden-to-output layer of the action network is updated as

$$W_{a,tl}(p+1) = W_{a,tl}(p) - \beta_a \frac{\partial E_{ai}(k)}{\partial \hat{v}_{it}(x_k)} \frac{\partial \hat{v}_{it}(x_k)}{\partial W_{a,tl}(p)} = W_{a,tl}(p) - \beta_a e_{ai}(k)h_l(p).$$

The weight of input to hidden layer of the action network is updated as

$$\begin{aligned}
 Y_{a,lj}(p+1) &= Y_{a,lj}(p) - \beta_a \frac{\partial E_{ai}(k)}{\partial \hat{v}_{it}(x_k)} \frac{\partial \hat{v}_{it}(x_k)}{\partial Y_{a,lj}(p)} \\
 &= Y_{a,lj}(p) - \beta_a \frac{\partial E_{ai}(k)}{\partial \hat{v}_{it}(x_k)} \frac{\partial \hat{v}_{it}(x_k)}{\partial h_l(p)} \frac{\partial h_l(p)}{\partial g_l(p)} \frac{\partial g_l(p)}{\partial Y_{a,lj}(p)} \\
 &= Y_{a,lj}(p) - \beta_a e_{ai}(k) W_{a,lj}(k) (1 - g_l^2(t)) x_{jk}.
 \end{aligned}$$

The detailed NN training algorithm can also be seen in [22, 28].

3.2.3 Simulation Study

Consider the following discrete-time nonaffine nonlinear system given by [14, 33]

$$x_{k+1} = F(x_k, u_k) = x_k + \sin(0.1x_k^2 + u_k). \quad (3.2.27)$$

The initial state is $x_0 = 1$. Since $F(0, 0) = 0$, $x_k = 0$ is an equilibrium state of system (3.2.27). But since

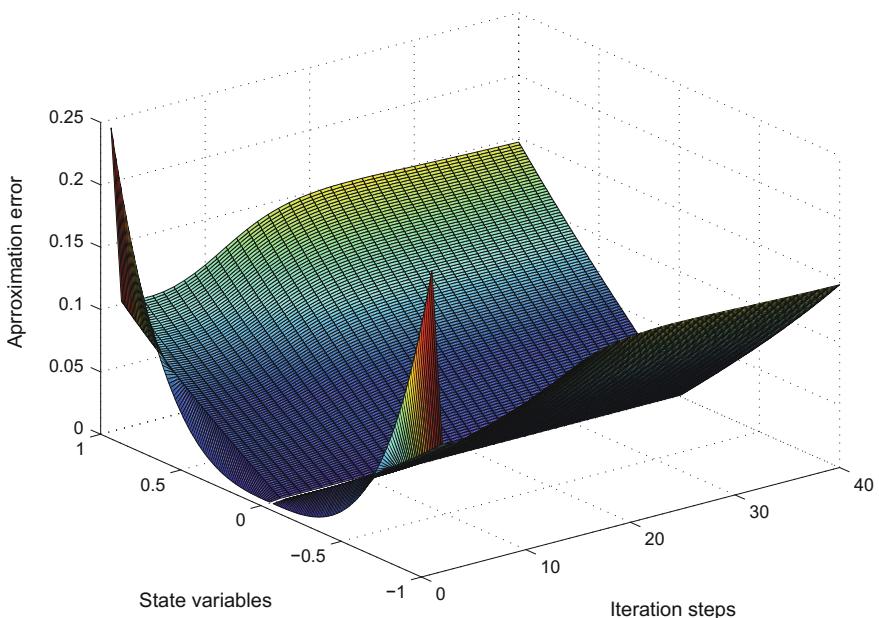


Fig. 3.2 The curve of approximation errors

$$\frac{\partial F(x_k, u_k)}{\partial x_k}(0, 0) = 1,$$

nonlinear system (3.2.27) is unstable at $x_k = 0$. Let the cost function be quadratic form which is expressed by

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

where Q and R are given as identity matrices with suitable dimensions.

Neural networks are used to implement the iterative ADP algorithm. The critic network and the action network are chosen as three-layer BP neural networks with the structures of 1–8–1 and 1–8–1, respectively. Neural networks are used to implement the iterative ADP algorithm and the neural network structural diagram of the algorithm can also be seen in [28, 32, 51]. Use $\theta = 8$ and $\Psi(x_k) = x_k^T Q x_k$ to initialize the algorithm.

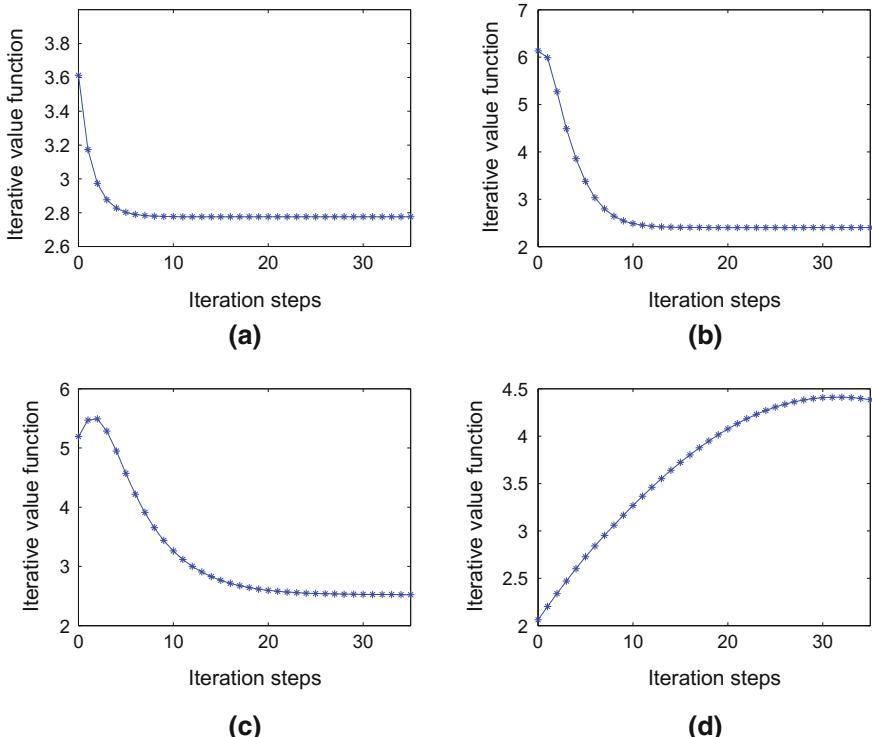


Fig. 3.3 Trajectories of the iterative value functions. **a** Approximation error $\epsilon = 10^{-6}$. **b** Approximation error $\epsilon = 10^{-4}$. **c** Approximation error $\epsilon = 10^{-3}$. **d** Approximation error $\epsilon = 10^{-1}$

The curve of the approximation errors is displayed in Fig. 3.2.

We choose a random array of state variable in $[-1, 1]$ in order to train the neural networks. For each iterative step, the critic network and the action network are trained for 1000 steps under the learning rate $\alpha_c = \beta_a = 0.001$, so that the approximation error is reached. The iterative ADP algorithm runs for 35 iteration steps to guarantee the convergence of the iterative value function. To show the effectiveness of the present iterative ADP algorithm, we choose four different global training precisions of neural networks. First, let the approximation error of the neural networks be $\varepsilon = 10^{-6}$. The trajectory of the iterative value function is shown in Fig. 3.3a. Second, let the approximation error of the neural networks be $\varepsilon = 10^{-4}$. The trajectory of the iterative value function is shown in Fig. 3.3b. Third, let the approximation error of the neural networks be $\varepsilon = 10^{-3}$. The trajectory of the iterative value function is shown in Fig. 3.3c. Forth, let the approximation error of the neural networks be $\varepsilon = 10^{-1}$. The trajectory of the iterative value function is shown in Fig. 3.3d.

For approximation error $\varepsilon = 10^{-6}$, implement the approximate optimal control for system (3.2.27). Let the implementation time $T_f = 20$. The trajectories of the

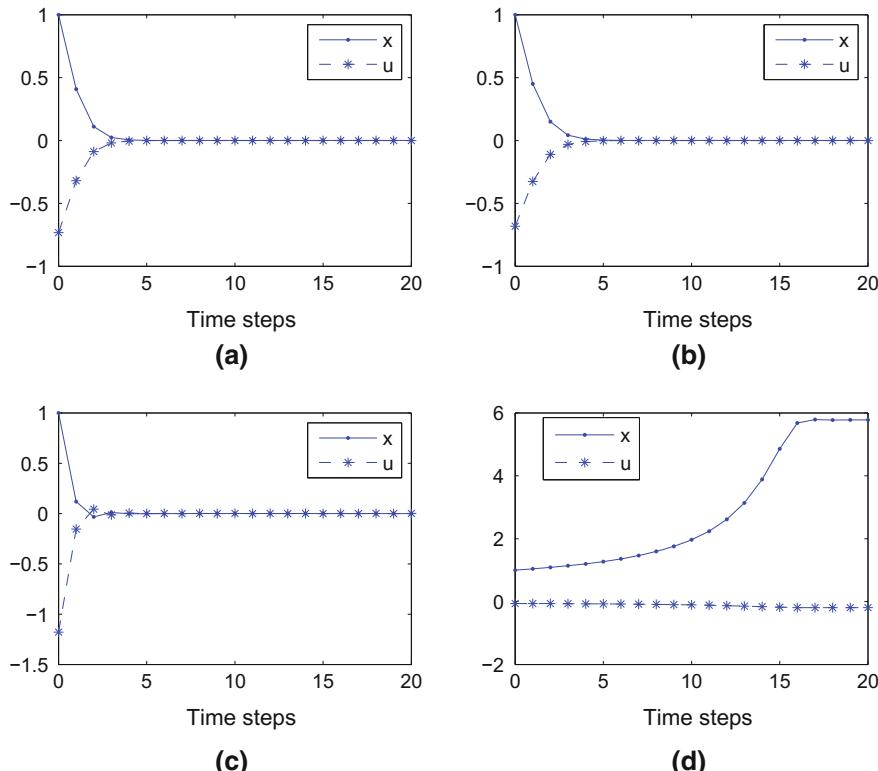


Fig. 3.4 Control and state trajectories. **a** Approximation error $\varepsilon = 10^{-8}$. **b** Approximation error $\varepsilon = 10^{-4}$. **c** Approximation error $\varepsilon = 10^{-3}$. **d** Approximation error $\varepsilon = 10^{-1}$

control and state are displayed in Fig. 3.4a. For approximation error $\varepsilon = 10^{-4}$, the trajectories of the control and state are displayed in Fig. 3.4b. When the approximation error $\varepsilon = 10^{-3}$, we can see that the iterative value functions is not monotonic. The trajectories of the control and state are displayed in Fig. 3.4c. When the approximation error $\varepsilon = 10^{-1}$, we can see that the iterative value functions is not convergent. In this situation, the control system is not stable, and the trajectories of the control and state are displayed in Fig. 3.4d.

3.3 Numerical Iterative θ -Adaptive Dynamic Programming

As the development of digital computers, numerical control attracts more and more attention of researchers [15, 49, 50]. In real-world implementations, especially for numerical control systems, for each i , the accurate iterative value functions $V_i(x_k)$ and the accurate iterative control laws $v_i(x_k)$ are generally impossible to obtain. For the situation that the control $u_k \in \mathfrak{A}$, the iterative θ -ADP algorithm in Sect. 2.3 may be invalid, where \mathfrak{A} denotes the set of numerical controls and we assume $0 \in \mathfrak{A}$. First, for numerical control systems, the set of numerical controls \mathfrak{A} is discrete. This means that there are only finite elements in the set of numerical controls \mathfrak{A} , which implies that the iterative control law and iterative value function can only be obtained with errors. Second, as $u_k \in \mathfrak{A}$, the convergence property of the iterative value function cannot be guaranteed, and the stability of the system under such controls cannot be proved either. Furthermore, even if we solve the iterative control law and the iterative value function at every iteration step, it is not clear whether the errors between the iterative value functions $V_i(x_k)$ and the optimal cost function $J^*(x_k)$ are finite or not for all i . Thus, the optimal cost function and optimal control law are nearly impossible to obtain by the iterative θ -ADP algorithm in Sect. 2.3.

In this section, a numerical iterative θ -ADP algorithm is developed to obtain the numerical optimal controller for nonaffine nonlinear system (3.2.1) [36, 37].

3.3.1 Derivation of the Numerical Iterative θ -ADP Algorithm

In the present numerical iterative θ -ADP algorithm, the value functions and control laws are updated by iterations, with the iteration index i increasing from 0 to infinity. For $x_k \in \mathbb{R}^n$, let the initial value function be

$$\hat{V}_0(x_k) = \theta \Psi(x_k), \quad (3.3.1)$$

where $\theta > 0$ is a large finite positive constant. The numerical iterative control law $\hat{v}_0(x_k)$ can be computed as follows:

$$\begin{aligned}\hat{v}_0(x_k) &= \arg \min_{u_k \in \mathfrak{A}} \left\{ U(x_k, u_k) + \hat{V}_0(x_{k+1}) \right\} \\ &= \arg \min_{u_k \in \mathfrak{A}} \left\{ U(x_k, u_k) + \hat{V}_0(F(x_k, u_k)) \right\},\end{aligned}\quad (3.3.2)$$

where $\hat{V}_0(x_{k+1}) = \theta \Psi(x_{k+1})$. For $i = 1, 2, \dots$, the numerical iterative θ -ADP algorithm will iterate between the iterative value functions

$$\begin{aligned}\hat{V}_i(x_k) &= \min_{u_k \in \mathfrak{A}} \left\{ U(x_k, u_k) + \hat{V}_{i-1}(x_{k+1}) \right\} \\ &= U(x_k, \hat{v}_{i-1}(x_k)) + \hat{V}_{i-1}(F(x_k, \hat{v}_{i-1}(x_k))),\end{aligned}\quad (3.3.3)$$

and the iterative control laws

$$\begin{aligned}\hat{v}_i(x_k) &= \arg \min_{u_k \in \mathfrak{A}} \left\{ U(x_k, u_k) + \hat{V}_i(x_{k+1}) \right\} \\ &= \arg \min_{u_k \in \mathfrak{A}} \left\{ U(x_k, u_k) + \hat{V}_i(F(x_k, u_k)) \right\}.\end{aligned}\quad (3.3.4)$$

Remark 3.3.1 The ADP algorithm in (3.3.2)–(3.3.4) is different from (3.2.6)–(3.2.8) in the choice of control set. Here, the control set \mathfrak{A} is discrete. As the set of numerical controls \mathfrak{A} is discrete, for all $i \geq 0$, $\hat{v}_i(x_k) \neq v_i(x_k)$ in general. Then, for all $i \geq 1$, the iterative value function $\hat{V}_i(x_k) \neq V_i(x_k)$, which means that there exists an error between $\hat{V}_i(x_k)$ and $V_i(x_k)$. It should be pointed out that the iterative approximation error is not a constant. The fact is that as the iteration index $i \rightarrow \infty$, the boundary of iterative approximation errors will also increase to infinity. The following lemma will show this property.

Lemma 3.3.1 *Let $x_k \in \mathbb{R}^n$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. For $i = 1, 2, \dots$, define a new iterative value function as*

$$\Gamma_i(x_k) = \min_{u_k} \{U(x_k, u_k) + \hat{V}_{i-1}(x_{k+1})\}, \quad (3.3.5)$$

where $\hat{V}_i(x_k)$ is defined in (3.3.3). If the initial value function

$$\Gamma_0(x_k) = \hat{V}_0(x_k) = \theta \Psi(x_k)$$

and for $i = 1, 2, \dots$, there exists a finite constant ε such that

$$\hat{V}_i(x_k) - \Gamma_i(x_k) \leq \varepsilon \quad (3.3.6)$$

holds uniformly, then

$$\hat{V}_i(x_k) - V_i(x_k) \leq i\varepsilon, \quad (3.3.7)$$

where ε is called uniform approximation error.

Proof The lemma can be proved by mathematical induction and by following similar steps as in the proof of Theorem 3.2.1. First, let $i = 1$. We have

$$\begin{aligned}\Gamma_1(x_k) &= \min_{u_k}\{U(x_k, u_k) + \hat{V}_0(x_{k+1})\} \\ &= \min_{u_k}\{U(x_k, u_k) + V_0(F(x_k, u_k))\} \\ &= V_1(x_k).\end{aligned}$$

Then, according to (3.3.6), we can get $\hat{V}_1(x_k) - V_1(x_k) \leq \varepsilon$. Assume that (3.3.7) holds for $i - 1$. Then, for i , we have

$$\begin{aligned}\Gamma_i(x_k) &= \min_{u_k}\{U(x_k, u_k) + \hat{V}_{i-1}(x_{k+1})\} \\ &\leq \min_{u_k}\{U(x_k, u_k) + V_{i-1}(x_{k+1}) + (i-1)\varepsilon\} \\ &= V_i(x_k) + (i-1)\varepsilon.\end{aligned}$$

Then, according to (3.3.6), we can get (3.3.7). The proof is complete.

Lemma 3.3.1 shows that although the approximation error for each single step is finite and may be small, as the iteration index $i \rightarrow \infty$ increases, the bounds of approximation errors between $\hat{V}_i(x_k)$ and $V_i(x_k)$ may also increase to infinity. To overcome these difficulties, we must discuss the convergence and stability properties of the iterative ADP algorithm in numerical implementation with finite approximation errors. For convenience of analysis, we perform transformation of approximation errors. According to the definitions of $\hat{V}_i(x_k)$ and $\Gamma_i(x_k)$ in (3.3.3) and (3.3.5), we have $\Gamma_i(x_k) \leq \hat{V}_i(x_k)$. Then, for $i = 0, 1, \dots$, there exists a $\eta \geq 1$ such that

$$\Gamma_i(x_k) \leq \hat{V}_i(x_k) \leq \eta \Gamma_i(x_k) \quad (3.3.8)$$

holds uniformly. Hence, we can give the following theorem.

Theorem 3.3.1 *Let $x_k \in \mathbb{R}^n$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. For $i = 0, 1, \dots$, let $\Gamma_i(x_k)$ be expressed as (3.3.5) and $\hat{V}_i(x_k)$ be expressed as (3.3.3). Let $\gamma < \infty$ and $1 \leq \beta < \infty$ be constants such that*

$$J^*(F(x_k, u_k)) \leq \gamma U(x_k, u_k), \quad (3.3.9)$$

and

$$J^*(x_k) \leq V_0(x_k) \leq \beta J^*(x_k) \quad (3.3.10)$$

hold uniformly. If there exists a η , $1 \leq \eta < \infty$, such that (3.3.8) is satisfied and

$$\eta \leq 1 + \frac{\beta - 1}{\gamma \beta}, \quad (3.3.11)$$

then the iterative value function $\hat{V}_i(x_k)$ converges to a bounded neighborhood of $J^*(x_k)$, as $i \rightarrow \infty$.

This theorem can be proved following the same procedure as Theorem 3.2.3 by noting that condition (3.3.11) implies (3.2.19) of Theorem 3.2.3. We omit the details.

Theorem 3.3.2 *Let $x_k \in \mathbb{R}^n$ be an arbitrary controllable state. If $\forall x_k$, Theorem 3.3.1 holds and η satisfies (3.3.11), then for $i = 0, 1, \dots$, the numerical iterative control law $\hat{v}_i(x_k)$ is an asymptotically stable control law for system (3.2.1).*

Proof As $\hat{V}_0(x_k) = V_0(x_k) = \theta\Psi(x_k)$, $\hat{V}_0(x_k)$ is a positive definite function for $i = 0$. Using the mathematical induction, assume that the iterative value function $\hat{V}_i(x_k)$, $i = 0, 1, \dots$, is positive definite. Then, according to Assumptions 3.2.1–3.2.3, we can get

$$\hat{V}_i(0) = U(0, \hat{v}_{i-1}(0)) + \hat{V}_{i-1}(F(0, \hat{v}_{i-1}(0))) = 0$$

for $x_k = 0$. When $x_k \rightarrow \infty$, as the utility function $U(x_k, u_k)$ is a positive function of x_k , we have $\hat{V}_i(x_k) \rightarrow \infty$. Hence, $\hat{V}_i(x_k)$ is a positive definite function and the mathematical induction is complete. Next, let χ_i be defined as

$$\chi_i = \eta \left(1 + \sum_{j=1}^i \frac{\gamma^j \eta^{j-1} (\eta - 1)}{(\gamma + 1)^j} + \frac{\gamma^i \eta^i (\beta - 1)}{(\gamma + 1)^i} \right),$$

and $\bar{V}_i(x_k) = \chi_i J^*(x_k)$.

As $1 \leq \eta \leq 1 + (\beta - 1)/(\gamma\beta)$, we have $\chi_{i+1} \leq \chi_i$, which means $\bar{V}_{i+1}(x_k) \leq \bar{V}_i(x_k)$. Define a new iterative value function $P_{i+1}(x_k)$ as

$$P_{i+1}(x_k) = U(x_k, \bar{v}_i(x_k)) + \bar{V}_i(x_{k+1}),$$

where $\bar{v}_i(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + \bar{V}_i(x_{k+1})\}$. From Theorem 3.2.2, $\bar{V}_i(x_k) = \chi_i J^*(x_k)$ is the upper bound of any iterative value function. Thus, we have $P_{i+1}(x_k) \leq \bar{V}_{i+1}(x_k) \leq \bar{V}_i(x_k)$, which implies

$$\bar{V}_i(x_{k+1}) - \bar{V}_i(x_k) \leq -U(x_k, \bar{v}_i(x_k)) < 0.$$

Hence, $\bar{V}_i(x_k)$ is a Lyapunov function and $\bar{v}_i(x_k)$ is an asymptotically stable control law for $i = 0, 1, \dots$. As $\bar{v}_i(x_k)$ is an asymptotically stable control law, we have $x_{k+N} \rightarrow 0$ as $N \rightarrow \infty$, that is $\bar{V}_i(x_{k+N}) \rightarrow 0$. Since $0 < \hat{V}_i(x_k) \leq \bar{V}_i(x_k)$ holds for all x_k , then we can get $0 < \hat{V}_i(x_{k+N}) \leq \bar{V}_i(x_{k+N})$ as $N \rightarrow \infty$. So, $\hat{V}_i(x_{k+N}) \rightarrow 0$ as $N \rightarrow \infty$. As $\hat{V}_i(x_k)$ is a positive definite function, we can conclude $x_{k+N} \rightarrow 0$ as $N \rightarrow \infty$ under the control law $\hat{v}_i(x_k)$. The proof is complete.

3.3.2 Properties of the Numerical Iterative θ -ADP Algorithm

Although Theorem 3.3.1 gives the convergence criterion, we can see that the parameters η , γ , and β are very difficult to achieve, and the convergence criterion (3.3.11) is quite difficult to verify. To overcome this difficulty, a new convergence condition must be developed to guarantee the convergence of the numerical iterative θ -ADP algorithm. For the convenience of analysis, we define a new value function as

$$\begin{cases} \hat{\mathcal{V}}_0(x_k, 0) = \hat{V}_0(x_k), \\ \hat{\mathcal{V}}_i(x_k, \hat{v}_i(x_k)) = U(x_k, \hat{v}_i(x_k)) + \hat{V}_{i-1}(F(x_k, \hat{v}_i(x_k))), \quad i = 1, 2, \dots, \end{cases} \quad (3.3.12)$$

where we can see that $\hat{V}_i(x_k) = \hat{\mathcal{V}}_i(x_k, \hat{v}_i(x_k))$. We have the following definition.

Definition 3.3.1 The iterative value function $\hat{\mathcal{V}}_i(x_k, \hat{v}_i(x_k))$ is a Lipschitz continuous function for all $\hat{v}_i(x_k)$, if there exists an $L \in \mathbb{R}$ such that

$$|\hat{\mathcal{V}}_i(x_k, \hat{v}_i(x_k)) - \hat{\mathcal{V}}_i(x_k, \hat{v}'_i(x_k))| \leq L \|\hat{v}_i(x_k) - \hat{v}'_i(x_k)\|, \quad (3.3.13)$$

where $\hat{v}'_i(x_k) \in \mathfrak{A}$ and $\hat{v}'_i(x_k) \neq \hat{v}_i(x_k)$.

As $\hat{\mathcal{V}}_i(x_k, \hat{v}_i(x_k))$ is a function to be approximated by neural networks, the Lipschitz assumption is reasonable. For the numerical control system, the set of numerical controls \mathfrak{A} is discrete. Then, according to the grid principle, let P_j , $j = 1, 2, \dots, m$, be the discrete grids for the j th dimension in \mathfrak{A} . Let Z be the set of positive integers. As $\mathfrak{A} \subset \mathbb{R}^m$, using the grid principle, we can define \mathfrak{A} as

$$\mathfrak{A} = \{u(p_1, \dots, p_m) : p_1, \dots, p_m \in Z, 1 \leq p_1 \leq P_1, \dots, 1 \leq p_m \leq P_m\} \quad (3.3.14)$$

to denote all the control elements in \mathfrak{A} , where P_1, \dots, P_m are all positive integers. Thus, for all $x_k \in \mathbb{R}^n$ and $i = 0, 1, \dots$, there exists a sequence of positive numbers p_1^i, \dots, p_m^i such that

$$u(p_1^i, \dots, p_m^i) = \hat{v}_i(x_k), \quad (3.3.15)$$

where $1 \leq p_1^i \leq P_1, \dots, 1 \leq p_m^i \leq P_m$. Then, according to (3.3.12), we can rewrite $\hat{\mathcal{V}}_i(x_k, \hat{v}_i(x_k))$ as

$$\begin{aligned} \hat{\mathcal{V}}_i(x_k, \hat{v}_i(x_k)) &= \min_{u_k \in \mathfrak{A}} \left\{ U(x_k, u_k) + \hat{V}_{i-1}(x_{k+1}) \right\} \\ &= U(x_k, u(p_1^i, \dots, p_m^i)) + \hat{V}_{i-1}(F(x_k, u(p_1^i, \dots, p_m^i))) \\ &= \hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)). \end{aligned} \quad (3.3.16)$$

Next, $\forall u(p_1^i, \dots, p_m^i) \in \mathfrak{A}, 1 \leq p_j^i \leq P_j, j = 1, \dots, m$, we can define a neighborhood set of $u(p_1^i, \dots, p_m^i)$ as

$$\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i) = \{u(\bar{p}_1^i, \dots, \bar{p}_m^i) : (\bar{p}_1^i, \dots, \bar{p}_m^i) \in \mathfrak{A}, |p_j^i - \bar{p}_j^i| \leq r\}, \quad (3.3.17)$$

where $r \in \mathbb{Z}$ is defined as the radius of $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$.

Define a new value function $\gamma_i(x_k, \tilde{v}_i(x_k))$ as

$$\begin{aligned} \gamma_i(x_k, \tilde{v}_{i-1}(x_k)) &= \min_{u_k} \{U(x_k, u_k) + \hat{V}_{i-1}(x_{k+1})\} \\ &= U(x_k, \tilde{v}_{i-1}(x_k)) + \hat{V}_{i-1}(F(x_k, \tilde{v}_{i-1}(x_k))). \end{aligned} \quad (3.3.18)$$

We can see that $\Gamma_i(x_k) = \gamma_i(x_k, \tilde{v}_i(x_k))$. As $\tilde{v}_i(x_k)$ cannot be obtained, it is very difficult to analyze its properties. While $\forall u(p_1^i, \dots, p_m^i)$, we can get $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$ by (3.3.17) immediately. So, if $\tilde{v}_i(x_k)$ is inside $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$ with the radius $r \geq 1$, then the properties of $\tilde{v}_i(x_k)$ can be obtained. We will analyze the relationship between $\tilde{v}_i(x_k)$ and $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$ next. Before that, some lemmas are necessary.

Lemma 3.3.2 *Let $O = (p_1^i, \dots, p_m^i)$ denote the origin of the m -dimensional coordinate system, and let \overline{OL} be an arbitrary vector in the m -dimensional space. If we let ϑ_j , $j = 1, 2, \dots, m$, be the intersection angle between \overline{OL} and the j th coordinate axis, then we have*

$$\sum_{j=1}^m \cos^2 \vartheta_j = 1.$$

Proof Let $L = (l_1^i, \dots, l_m^i)$ be an arbitrary point in the m -dimensional coordinate system. Then, $\overline{OL} = ((l_1^i - p_1^i), \dots, (l_m^i - p_m^i))$. According to the definition of ϑ_j , we have

$$\cos \vartheta_j = \frac{|l_j^i - p_j^i|}{|\overline{OL}|}.$$

where

$$|\overline{OL}| = \sqrt{(l_1^i - p_1^i)^2 + \dots + (l_m^i - p_m^i)^2}.$$

Then,

$$\sum_{j=1}^m \cos^2 \vartheta_j = \frac{(|l_1^i - p_1^i|)^2 + \dots + (|l_m^i - p_m^i|)^2}{(l_1^i - p_1^i)^2 + \dots + (l_m^i - p_m^i)^2} = 1.$$

The proof is complete.

Lemma 3.3.3 *Let $O = (p_1^i, \dots, p_m^i)$ denote the origin of the m -dimensional coordinate system and let \overline{OL} be an arbitrary vector in the m -dimensional space. Let A_j , $j = 1, 2, \dots, m$, be points on the j th coordinate axis of m -dimensional space and $\forall j = 1, 2, \dots, m$, $\overline{OA_j} = \overline{OL}$. If we let $\vartheta_j = \min\{\vartheta_1, \dots, \vartheta_m\}$, then we have*

$$\overline{A_j L} \leq \sqrt{(m-1)/m} \overline{O L} / \cos\left(\frac{1}{2} \arcsin(\sqrt{(m-1)/m})\right). \quad (3.3.19)$$

Proof Letting $\vartheta_1 = \dots = \vartheta_m = \arccos(1/\sqrt{m})$, then ϑ_j reaches the maximum. Let $\alpha_j = \angle O A_j L$. We can get $\alpha_j = \frac{1}{2}(\pi - \vartheta_j)$. According to sine rule, we have

$$\overline{A_j L} \leq (\sin \vartheta_j / \sin \alpha_j) \overline{O L}. \quad (3.3.20)$$

Since $\cos \vartheta_j = 1/\sqrt{m}$, we have

$$\sin \vartheta_j = \sqrt{(m-1)/m}, \quad \sin \alpha_j = \cos\left(\frac{1}{2} \arcsin(\sqrt{(m-1)/m})\right).$$

Taking $\sin \vartheta_j$ and $\sin \alpha_j$ into (3.3.20), we can obtain the conclusion. The proof is complete.

Lemma 3.3.4 *Let $O = (p_1^i, \dots, p_m^i)$ denote the origin of the m -dimensional coordinate system and let $A_\ell = (\ell p_1^i, \dots, \ell p_m^i)$ be an arbitrary point in $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$, where $1 \leq \ell \leq (2r+1)^m$. Let $L = (\bar{p}_1^i, \dots, \bar{p}_m^i)$ be an arbitrary point such that $\overline{O L} \geq \max_{1 \leq \ell \leq (2r+1)^m} \{\overline{O A_\ell}\}$. If the radius r of $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$ satisfies the following inequality*

$$r \geq \frac{3(m-1) + \sqrt{3(m-1)}}{3m}, \quad (3.3.21)$$

then there exists an ℓ such that

$$\overline{A_\ell L} \leq \overline{O L}. \quad (3.3.22)$$

Proof Without loss of generality, let $L = (\bar{p}_1^i, \dots, \bar{p}_m^i)$ be located in the first quadrant. According to Lemmas 3.3.2 and 3.3.3, we can see that if intersection angles satisfy $\vartheta_1 = \dots = \vartheta_m = \arccos(1/\sqrt{m})$, the value on the left-hand side of (3.3.19) reaches its maximum for each coordinate axis. In this situation, we can let $L = (p_1^i + r, \dots, p_m^i + r)$. Let $A_\ell = (0, \dots, 0, p_m^i + r)$ and $A'_\ell = (p_1^i + r - 1, \dots, p_{m-1}^i + r - 1, p_m^i + r)$. Then, we can see that the points A_ℓ , A'_ℓ , and L are on the same line. Let $\vartheta'_\ell = \angle B'_\ell O L$. Then, we have

$$\cos \vartheta'_\ell = \frac{(\overline{O A'_\ell})^2 + (\overline{O L})^2 - (\overline{L A'_\ell})^2}{2 \overline{O A'_\ell} \overline{O L}}. \quad (3.3.23)$$

Taking the coordinates of A'_ℓ and L into (3.3.23), we can get

$$\cos \vartheta'_\ell = \frac{(r-1)m + 1}{\sqrt{m} \sqrt{(r-1)^2 m - (r-1)^2 + r^2}}. \quad (3.3.24)$$

Next, extend the line $\overline{OA'_\ell}$ to B'_ℓ so that it satisfies $\overline{OB'_\ell} = \overline{OL}$. Then, $\triangle OB'_\ell L$ is an isosceles triangle. It is obvious that when $\vartheta'_\ell \leq \pi/3$, that is $\cos \vartheta'_\ell \geq \frac{1}{2}$, we can get $\overline{B'_\ell L} \leq \overline{OL}$. According to (3.3.24), we can obtain (3.3.21). On the other hand, according to sine rule [2], we can get $\frac{\sin \vartheta'_\ell}{\overline{B'_\ell L}} = \frac{\sin \angle OB'_\ell L}{\overline{OL}}$, and $\frac{\sin \vartheta'_\ell}{\overline{A'_\ell L}} = \frac{\sin \angle OA'_\ell L}{\overline{OL}}$. If $\angle OB'_\ell L \leq \angle OA'_\ell L \leq \frac{\pi}{2}$, we can easily obtain $\overline{A'_\ell L} \leq \overline{B'_\ell L} \leq \overline{OL}$. If $\frac{\pi}{2} \leq \angle OA'_\ell L \leq \pi$, then $\angle OA'_\ell L$ is the maximum angle in $\triangle OA'_\ell L$, which obtain $\overline{A'_\ell L} \leq \overline{OL}$ directly. The proof is complete.

Given the above preparation, we derive the following theorem.

Theorem 3.3.3 *Let $\hat{v}_i(x_k) = u(p_1^i, p_2^i, \dots, p_m^i) \in \mathfrak{A}$ and let $u(\ell p_1^i, \ell p_2^i, \dots, \ell p_m^i) \in \hat{\mathfrak{A}}(p_1^i, p_2^i, \dots, p_m^i)$, $1 \leq \ell \leq (2r+1)^m$, be an arbitrary control vector. If the radius r of $\hat{\mathfrak{A}}(p_1^i, p_2^i, \dots, p_m^i)$ satisfies (3.3.21), then there exists a positive number $L \in \mathbb{R}$ such that*

$$|\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - \mathcal{V}_i(x_k, \tilde{v}_i(x_k))| \leq L \max_{1 \leq \ell \leq (2r+1)^m} \{ \|u(p_1^i, \dots, p_m^i) - u(\ell p_1^i, \dots, \ell p_m^i)\| \}.$$

Proof According to the definitions of the iterative value functions $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ and $\mathcal{V}_i(x_k, \tilde{v}_i(x_k))$ in (3.3.16) and (3.3.18), respectively, we can see that if we put the control $\tilde{v}_i(x_k)$ into the set of numerical controls \mathfrak{A} , then

$$\hat{\mathcal{V}}_i(x_k, \tilde{v}_i(x_k)) = \mathcal{V}_i(x_k, \tilde{v}_i(x_k)).$$

For the control $u(p_1^i, \dots, p_m^i)$, according to Definition 3.3.1, there exists an L such that

$$|\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - \mathcal{V}_i(x_k, \tilde{v}_i(x_k))| \leq L \|u(p_1^i, \dots, p_m^i) - \tilde{v}_i(x_k)\|.$$

Since $\tilde{v}_i(x_k)$ cannot be obtained accurately, the distance between $\tilde{v}_i(x_k)$ and $u(p_1^i, \dots, p_m^i)$ is unknown. Hence, $\tilde{v}_i(x_k)$ must be replaced by other known vector. Next, we will show

$$\|u(p_1^i, \dots, p_m^i) - \tilde{v}_i(x_k)\| \leq \max_{1 \leq \ell \leq (2r+1)^m} \{ \|u(p_1^i, \dots, p_m^i) - u(\ell p_1^i, \dots, \ell p_m^i)\| \}. \quad (3.3.25)$$

As

$$\hat{v}_i(x_k) = u(p_1^i, p_2^i, \dots, p_m^i) \in \mathfrak{A},$$

then $\tilde{v}_i(x_k)$ becomes the neighboring point of $u(p_1^i, \dots, p_m^i)$ and we can put $\tilde{v}_i(x_k)$ into $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$ such that $\tilde{v}_i(x_k) \in \bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$. Next, we will prove the conclusion by contradiction. Assume that the inequality (3.3.25) does not hold. Then,

$$\|u(p_1^i, \dots, p_m^i) - \tilde{v}_i(x_k)\| > \max_{1 \leq \ell \leq (2r+1)^m} \{ \|u(p_1^i, \dots, p_m^i) - u(\ell p_1^i, \dots, \ell p_m^i)\| \}$$

as $\tilde{v}_i(x_k)$ belongs to the set $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$. As there are m dimensions in $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$, we can divide it into 2^m quadrants.

Without loss of generality, let $\tilde{v}_i(x_k)$ be located in the first quadrant where $L = (\bar{p}_1^i, \dots, \bar{p}_m^i)$ is the corresponding coordinate. If we let $O = (p_1^i, \dots, p_m^i)$ be the origin, then

$$\overline{OL} = \|u(p_1^i, \dots, p_m^i) - \tilde{v}_i(x_k)\|.$$

As (3.3.21) holds, according to Theorem 3.3.4, if \overline{OL} is the max vector in $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$, then there exists a vector $OA'_\ell \in \bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$ with the coordinate $A'_\ell = (\ell p_1^i, \dots, \ell p_m^i)$, such that (3.3.22) is true. Let L_1 be the Lipschitz constant. Then, we can get

$$\begin{aligned} |\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - \Upsilon_i(x_k, \tilde{v}_i(x_k))| \\ = L_1 \|u(p_1^i, \dots, p_m^i) - \tilde{v}_i(x_k)\| \\ \geq L_1 \|u(\ell p_1^i, \dots, \ell p_m^i) - \tilde{v}_i(x_k)\| \\ = |\hat{\mathcal{V}}_i(x_k, u_\ell^i(\ell p_1^i, \dots, \ell p_m^i)) - \Upsilon_i(x_k, \tilde{v}_i(x_k))|. \end{aligned} \quad (3.3.26)$$

According to the definition of $\Upsilon_i(x_k, \tilde{v}_i(x_k))$ in (3.3.18), we know $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) \geq \Upsilon_i(x_k, \tilde{v}_i(x_k))$ and $\hat{\mathcal{V}}_i(x_k, u_\ell^i(\ell p_1^i, \dots, \ell p_m^i)) \geq \Upsilon_i(x_k, \tilde{v}_i(x_k))$, for $i = 0, 1, \dots$. Thus, according to (3.3.26), we can obtain

$$\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) > \hat{\mathcal{V}}_i(x_k, u_\ell^i(\ell p_1^i, \dots, \ell p_m^i)).$$

This contradicts the definition of $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ in (3.3.16). Therefore, the assumption is false and the inequality (3.3.25) holds. The proof is complete.

According to the definitions of iterative value functions $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ and $\Upsilon_i(x_k, \tilde{v}_i(x_k))$ in (3.3.16) and (3.3.18), respectively, for $i = 0, 1, \dots$, we can define

$$\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - \Upsilon_i(x_k, \tilde{v}_i(x_k)) = \varepsilon_i(x_k), \quad (3.3.27)$$

where $\varepsilon_0(x_k) = 0$. Then, for any $\varepsilon_i(x_k)$ expressed in (3.3.27), there exists a $\eta_i(x_k)$ such that

$$\begin{aligned}\gamma_i(x_k, \tilde{v}_i(x_k)) &= \hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - \varepsilon_i(x_k) \\ &= \frac{\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))}{\eta_i(x_k)}.\end{aligned}$$

Theorem 3.3.4 Let the iterative value function $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ and the numerical iterative control $u(p_1^i, \dots, p_m^i)$ be obtained by (3.3.15) and (3.3.16), respectively. If for $x_k \in \mathbb{R}^n$, we define the admissible approximation error as

$$\bar{\varepsilon}_i(x_k) = L_i(p_1^i, \dots, p_m^i) \max_{1 \leq \ell \leq (2r+1)^m} \{ \|u(p_1^i, \dots, p_m^i) - u(\ell p_1^i, \dots, \ell p_m^i)\| \}, \quad (3.3.28)$$

where $L_i(p_1^i, \dots, p_m^i)$ are Lipschitz constants and $\bar{\varepsilon}_0(x_k) = 0$, then for $i = 0, 1, \dots$, we have

$$\varepsilon_i(x_k) \leq \bar{\varepsilon}_i(x_k).$$

Proof As $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ are Lipschitz continuous, according to (3.3.13), we have

$$|\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - \gamma_i(x_k, \tilde{v}_i(x_k))| \leq L_i(p_1^i, \dots, p_m^i) (u(p_1^i, \dots, p_m^i) - \tilde{v}_i(x_k)),$$

where $L_i(p_1^i, \dots, p_m^i)$ are the Lipschitz constants. According to (3.3.25), we can draw the conclusion. The proof is complete.

For the set of numerical control \mathfrak{A} expressed in (3.3.14), as \mathfrak{A} is known, then for any control $u(p_1^i, \dots, p_m^i) \in \mathfrak{A}$, we can obtain $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$ immediately. Hence, if the Lipschitz constants $L_i(p_1^i, \dots, p_m^i)$ are known, then the error $\bar{\varepsilon}_i$ can be obtained by (3.3.28). Next, we will give a method to obtain $L_i(p_1^i, \dots, p_m^i)$. Let $u(\ell p_1^i, \dots, \ell p_m^i) \in \bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$ be an arbitrary control vector. Then, we can get

$$\begin{aligned}|\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - \hat{\mathcal{V}}_i(x_k, u(\ell p_1^i, \dots, \ell p_m^i))| \\ = \bar{L}_i^\ell(p_1^i, \dots, p_m^i) \|u(p_1^i, \dots, p_m^i) - u(\ell p_1^i, \dots, \ell p_m^i)\|,\end{aligned} \quad (3.3.29)$$

where $\bar{L}_i^\ell(p_1^i, \dots, p_m^i) > 0$, $\ell = 1, 2, \dots, (2r+1)^m$, $i = 0, 1, \dots$

Let

$$\bar{L}_i(p_1^i, \dots, p_m^i) = \max_{1 \leq \ell \leq (2r+1)^m} \{ \bar{L}_i^\ell(p_1^i, \dots, p_m^i) \}, \quad (3.3.30)$$

be the local Lipschitz constant. Then, (3.3.29) can be written as

$$\begin{aligned}|\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - \hat{\mathcal{V}}_i(x_k, u(\ell p_1^i, \dots, \ell p_m^i))| \\ \leq \bar{L}_i(p_1^i, \dots, p_m^i) \|u(p_1^i, \dots, p_m^i) - u(\ell p_1^i, \dots, \ell p_m^i)\|.\end{aligned}$$

For all $u(p_1^i, \dots, p_m^i) \in \mathfrak{A}$, we can define the global Lipschitz constant \bar{L}_i as

$$\bar{L}_i = \max \left\{ \bar{L}_i(p_1^i, \dots, p_m^i) : 1 \leq p_j^i \leq P_j, j = 1, 2, \dots, m \right\}. \quad (3.3.31)$$

Thus, from (3.3.30) and (3.3.31), we can easily obtain $L_i(p_1^i, \dots, p_m^i) \leq \bar{L}_i$.

In the above, we give an effective method to obtain the approximation error $\bar{\varepsilon}_{i+1}(x_k)$ of the numerical iterative θ -ADP algorithm. We will show how to obtain the admissible approximation error to guarantee the convergence criterion of the present numerical iterative ADP algorithm. According to (3.3.9), we can define $\gamma = \max \{J^*(F(x_k, u_k))/U(x_k, u_k) : x_k \in \mathbb{R}^n, u_k \in \mathfrak{A}\}$. If we let

$$\tilde{\gamma}_i = \left\{ \frac{V_i(F(x_k, u_k))}{U(x_k, u_k)} : x_k \in \mathbb{R}^n, u_k \in \mathfrak{A} \right\}, \quad (3.3.32)$$

then we can get $\tilde{\gamma}_i \geq \gamma$. Before the next theorem, we introduce some notation. Let

$$\bar{\eta}_i(x_k) = \frac{\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))}{\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - \bar{\varepsilon}_i(x_k)}, \quad (3.3.33)$$

and

$$\beta_i(x_k) = \frac{\hat{\mathcal{V}}_0(x_k, 0)}{\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))}. \quad (3.3.34)$$

Theorem 3.3.5 *Let the iterative value function $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ be defined in (3.3.12) and the numerical iterative control $u(p_1^i, \dots, p_m^i)$ be defined in (3.3.15). For $i = 0, 1, \dots$, if the approximation error satisfies*

$$\begin{aligned} \bar{\varepsilon}_i(x_k) \leq & \frac{\mathcal{V}_i(x_k, u(p_1^i, \dots, p_m^i))}{\hat{\mathcal{V}}_0(x_k, 0)(\tilde{\gamma}_i + 1) - \mathcal{V}_i(x_k, u(p_1^i, \dots, p_m^i))} \\ & \times (\hat{\mathcal{V}}_0(x_k, 0) - \mathcal{V}_i(x_k, u(p_1^i, \dots, p_m^i))), \end{aligned} \quad (3.3.35)$$

then the numerical iterative control law $u(p_1^i, \dots, p_m^i)$ stabilizes the nonlinear system (3.2.1) and simultaneously guarantees the iterative value function $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ to converge to a finite neighborhood of $J^(x_k)$, as $i \rightarrow \infty$.*

Proof From (3.3.10), we can define

$$\beta = \max \left\{ \frac{V_0(x_k)}{J^*(x_k)} : x_k \in \mathbb{R}^n \right\}.$$

From (3.3.32)–(3.3.34), for all $x_k \in \mathbb{R}^n$, we can obtain

$$\begin{cases} \eta_i(x_k) \leq \tilde{\eta}_i(x_k), \\ 1 + \frac{\beta_i(x_k) - 1}{\tilde{\gamma}_i \beta_i(x_k)} \leq 1 + \frac{\beta - 1}{\gamma \beta}. \end{cases} \quad (3.3.36)$$

So, if

$$\tilde{\eta}_i(x_k) \leq 1 + \frac{\beta_i(x_k) - 1}{\tilde{\gamma}_i \beta_i(x_k)}, \quad (3.3.37)$$

then (3.3.36) holds. Putting (3.3.33) and (3.3.34) into (3.3.37), we can obtain (3.3.35).

On the other hand, according to (3.3.32) and the definition of η in (3.3.8), we have

$$\begin{aligned} \eta &= \max_{x_k \in \mathbb{R}^n, i=0,1,\dots} \{\eta_i(x_k)\} \leq \max_{x_k \in \mathbb{R}^n, i=0,1,\dots} \{\tilde{\eta}_i(x_k)\} \\ &\leq \max_{x_k \in \mathbb{R}^n, i=0,1,\dots} \left\{ 1 + \frac{\beta_i(x_k) - 1}{\tilde{\gamma}_i \beta_i(x_k)} \right\} \\ &\leq 1 + \frac{\beta - 1}{\gamma \beta}. \end{aligned}$$

By Theorems 3.3.1 and 3.3.2, we can draw the conclusion. The proof is complete.

Theorem 3.3.6 *Let the iterative value function $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ be defined in (3.3.12) and the numerical iterative control $u(p_1^i, \dots, p_m^i)$ be defined in (3.3.15). If for all $x_k \in \mathbb{R}^n$, we have*

$$U(x_k, u_k) \geq U(x_k, 0), \quad (3.3.38)$$

and for $i = 0, 1, \dots$, the approximation error satisfies

$$\bar{e}_i(x_k) \leq \frac{\mathcal{V}_i^2(x_k, u(p_1^i, \dots, p_m^i))(\hat{\mathcal{V}}_0(x_k, 0) - \mathcal{V}_i(x_k, u(p_1^i, \dots, p_m^i)))}{\hat{\mathcal{V}}_0(x_k, 0)\Delta_i^1(x_k, u(p_1^i, \dots, p_m^i)) + \Delta_i^2(x_k, u(p_1^i, \dots, p_m^i))}, \quad (3.3.39)$$

where

$$\Delta_i^1(x_k, u(p_1^i, \dots, p_m^i)) = \mathcal{V}_i(x_k, u(p_1^i, \dots, p_m^i)) - U(x_k, 0)$$

and

$$\Delta_i^2(x_k, u(p_1^i, \dots, p_m^i)) = \mathcal{V}_i(x_k, u(p_1^i, \dots, p_m^i))(\hat{\mathcal{V}}_0(x_k, 0) - \mathcal{V}_i(x_k, u(p_1^i, \dots, p_m^i))),$$

then the numerical iterative control law $u(p_1^i, \dots, p_m^i)$ stabilizes the nonlinear system (3.2.1) and simultaneously guarantees the iterative value functions $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ to converge to a finite neighborhood of $J^*(x_k)$, as $i \rightarrow \infty$.

Proof If we let

$$\hat{\gamma}_i = \max \left\{ \frac{\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))}{U(x_k, 0)} - 1 : x_k \in \mathbb{R}^n, u_k \in \mathfrak{A} \right\}, \quad (3.3.40)$$

then we can obtain

$$\hat{\gamma}_i \geq \frac{J^*(x_k)}{U(x_k, u^*(x_k))} - 1 \geq \gamma.$$

So, if

$$\bar{\eta}_i(x_k) \leq 1 + \frac{\beta_i(x_k) - 1}{\hat{\gamma}_i \beta_i(x_k)},$$

then (3.3.11) holds, which means that iterative value functions $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ converge to a finite neighborhood of $J^*(x_k)$ according to Theorem 3.3.1. According to (3.3.33), (3.3.34), and (3.3.40), and we can get (3.3.39). The proof is complete.

From Theorems 3.3.5 and 3.3.6, we can see that the information of the parameter γ_{i+1} should be used while the value of γ_{i+1} is usually difficult to obtain. So, in the next theorem, we give a more simplified convergence justification for the iterative ADP algorithm.

Theorem 3.3.7 *Let the iterative value function $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ and the numerical iterative control $u(p_1^i, \dots, p_m^i)$ be obtained by (3.3.15) and (3.3.16), respectively. Let $\bar{\varepsilon}_i$ be expressed as in (3.3.28). For $i = 0, 1, \dots$, if the utility function $U(x_k, u_k)$ satisfies (3.3.38) and the iterative approximation error satisfies*

$$\bar{\varepsilon}_i(x_k) \leq \hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - \frac{\hat{\mathcal{V}}_0(x_k, 0)}{\hat{\mathcal{V}}_0(x_k, 0) - U(x_k, 0)} \left(\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i)) - U(x_k, 0) \right), \quad (3.3.41)$$

then the numerical iterative control law $u(p_1^i, \dots, p_m^i)$ stabilizes the nonlinear system (3.2.1) and simultaneously guarantees the iterative value function $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ to converge to a finite neighborhood of $J^(x_k)$, as $i \rightarrow \infty$.*

Proof First, we look at (3.3.9) and (3.3.10). Without loss of generality, we let

$$\begin{aligned} \tilde{\gamma}(x_k) &= \frac{J^*(x_k) - U(x_k, u^*(x_k))}{U(x_k, u^*(x_k))}, \\ \tilde{\beta}(x_k) &= \frac{V_0(x_k)}{J^*(x_k)}. \end{aligned}$$

Then, we can get

$$\tilde{\gamma}(x_k) \tilde{\beta}(x_k) = \frac{V_0(x_k)}{U(x_k, u^*(x_k))} - \tilde{\beta}(x_k).$$

By (3.3.34), we can obtain

$$\max_{x_k \in \mathbb{R}^n, i=0,1,\dots} \{\beta_i(x_k)\} \leq \max_{x_k \in \mathbb{R}^n} \tilde{\beta}(x_k),$$

and

$$\max_{x_k \in \mathbb{R}^n, i=0,1,\dots} \left\{ \frac{\hat{\mathcal{V}}_0(x_k, 0)}{U(x_k, 0)} - \tilde{\beta}_i(x_k) \right\} \geq \max_{x_k \in \mathbb{R}^n} \{\tilde{\gamma}(x_k) \tilde{\beta}(x_k)\} = \gamma \beta.$$

Next, we notice that if $\bar{\eta}_i(x_k)$ satisfies

$$\bar{\eta}_i(x_k) \leq 1 + \frac{(\tilde{\beta}_i(x_k) - 1)U(x_k, 0)}{\hat{\mathcal{V}}_0(x_k, 0) - \tilde{\beta}_i(x_k)U(x_k, 0)}, \quad (3.3.42)$$

then

$$\begin{aligned} \eta &= \max_{x_k \in \mathbb{R}^n, i=0,1,\dots} \bar{\eta}_i(x_k) \\ &\leq \max_{x_k \in \mathbb{R}^n, i=0,1,\dots} \left\{ 1 + \frac{(\tilde{\beta}_i(x_k) - 1)U(x_k, 0)}{\hat{\mathcal{V}}_0(x_k, 0) - \tilde{\beta}_i(x_k)U(x_k, 0)} \right\} \\ &\leq 1 + \frac{\beta - 1}{\gamma \beta}. \end{aligned}$$

Substituting (3.3.33) and (3.3.34) into (3.3.42), we can obtain (3.3.41). The proof is complete.

3.3.3 Summary of the Numerical Iterative θ -ADP Algorithm

Now, we summarize the numerical iterative θ -ADP algorithm.

- Step 1: Choose randomly an array of initial states x_0 and choose the approximation precision ζ . Give the set of numerical controls \mathfrak{A} as in expression (3.3.14). Give the maximum iteration index i_{\max} . Choose a large θ . Let $i = 0$ and $\hat{V}_0(x_k) = \theta \Psi(x_k)$, where $\Psi(x_k) \in \bar{\Psi}_{x_k}$. Obtain $\hat{v}_0(x_k)$ from (3.3.2). Obtain $u(p_1^0, \dots, p_m^0) = \hat{v}_0(x_k)$ by (3.3.15) and $\bar{\mathfrak{A}}(p_1^0, p_2^0, \dots, p_m^0)$ by (3.3.17). Obtain x_{k+1} . Calculate $\hat{\mathcal{V}}_1(x_k, u(p_1^0, \dots, p_m^0))$ by (3.3.16).
- Step 2: Let $i = i + 1$. According to the numerical controls \mathfrak{A} , implement the numerical iterative θ -ADP algorithm (3.3.2)–(3.3.4) to obtain $\hat{V}_i(x_k)$ and $\hat{v}_i(x_k)$. Obtain $u(p_1^i, \dots, p_m^i) = \hat{v}_i(x_k)$ by (3.3.15) and $\bar{\mathfrak{A}}(p_1^i, \dots, p_m^i)$ by (3.3.17). Obtain x_{k+1} . Calculate $\hat{\mathcal{V}}_i(x_k, u(p_1^i, \dots, p_m^i))$ by (3.3.16).
- Step 3: Obtain the Lipschitz constant $L_i(p_1^i, \dots, p_m^i)$ according to (3.3.30). Compute $\bar{\varepsilon}_i(x_k)$ by (3.3.28).
- Step 4: If $\bar{\varepsilon}_i(x_k)$ satisfies (3.3.39), then go to next step. Otherwise, go to Step 6.

Step 5: If $|\hat{V}_i(x_k) - \hat{V}_{i-1}(x_k)| \leq \zeta$, then the iterative value function is converged and go to Step 9; elseif $i > i_{\max}$, then go to Step 6; else, let $i = i + 1$ and go to Step 2.

Step 6: Stop the algorithm.

3.3.4 Simulation Study

Consider the discrete-time nonlinear system given by

$$\begin{aligned} x_1(k+1) &= [x_1^2(k) + x_2^2(k) + u(k)] \cos(x_2(k)), \\ x_2(k+1) &= [x_1^2(k) + x_2^2(k) + u(k)] \sin(x_2(k)). \end{aligned} \quad (3.3.43)$$

Let $x_k = [x_1(k), x_2(k)]^\top$ denote the system state vector and $u_k = u(k)$ denote the control. Let

$$\mathfrak{A} = \{-2, -2 + \varsigma, -2 + 2\varsigma, \dots, 2\},$$

where ς is the grid step. The cost function is defined as (3.2.3) with the utility function

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

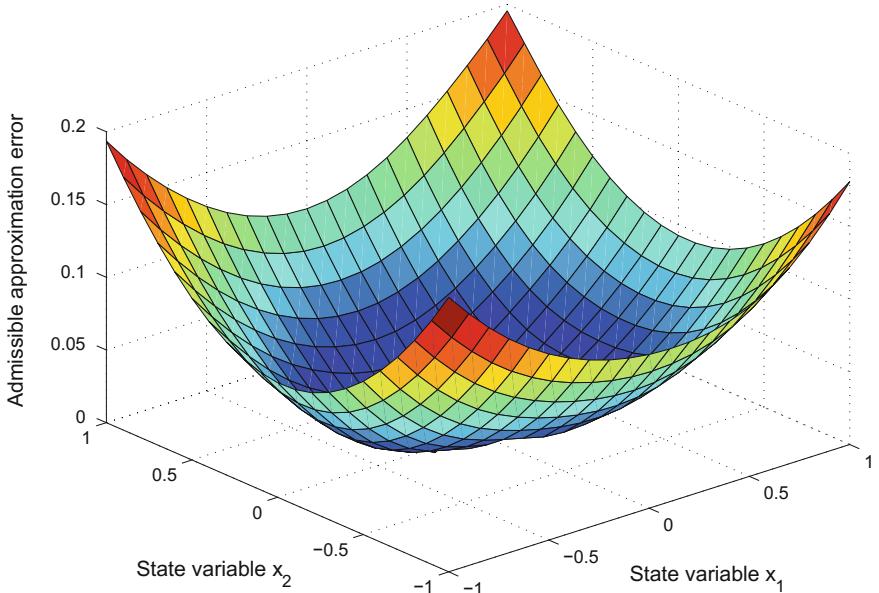


Fig. 3.5 The curve of the admissible approximation error obtained by (3.3.39)

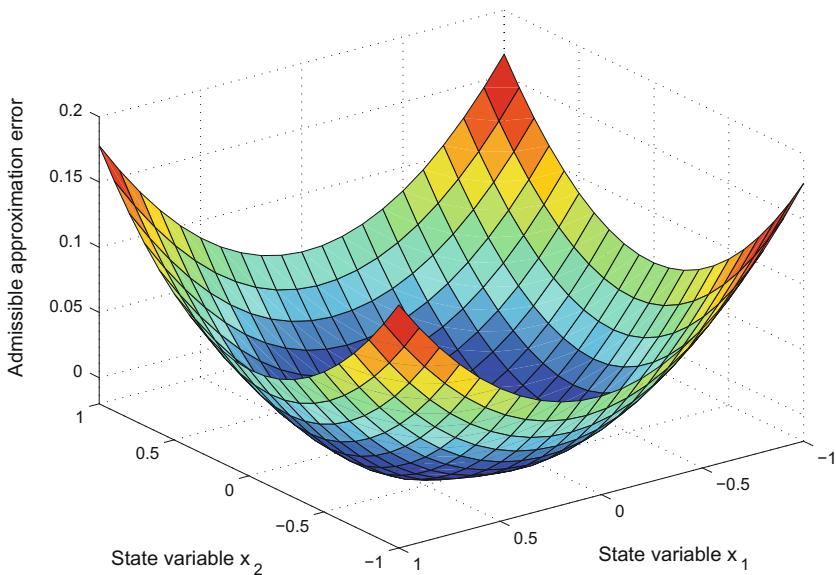


Fig. 3.6 The curve of the admissible approximation error obtained by (3.3.41)

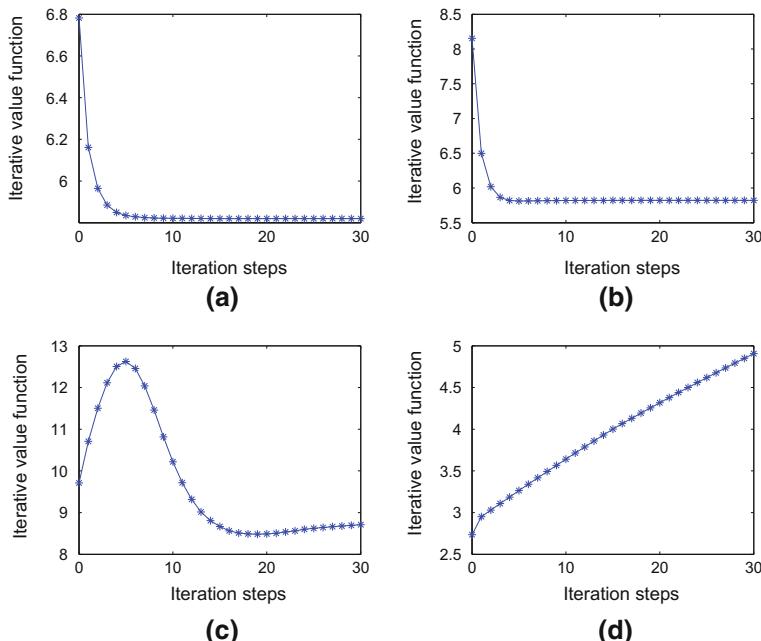


Fig. 3.7 The trajectories of the iterative value functions. **a** $\zeta = 10^{-8}$. **b** $\zeta = 10^{-4}$. **c** $\zeta = 10^{-2}$. **d** $\zeta = 10^{-1}$

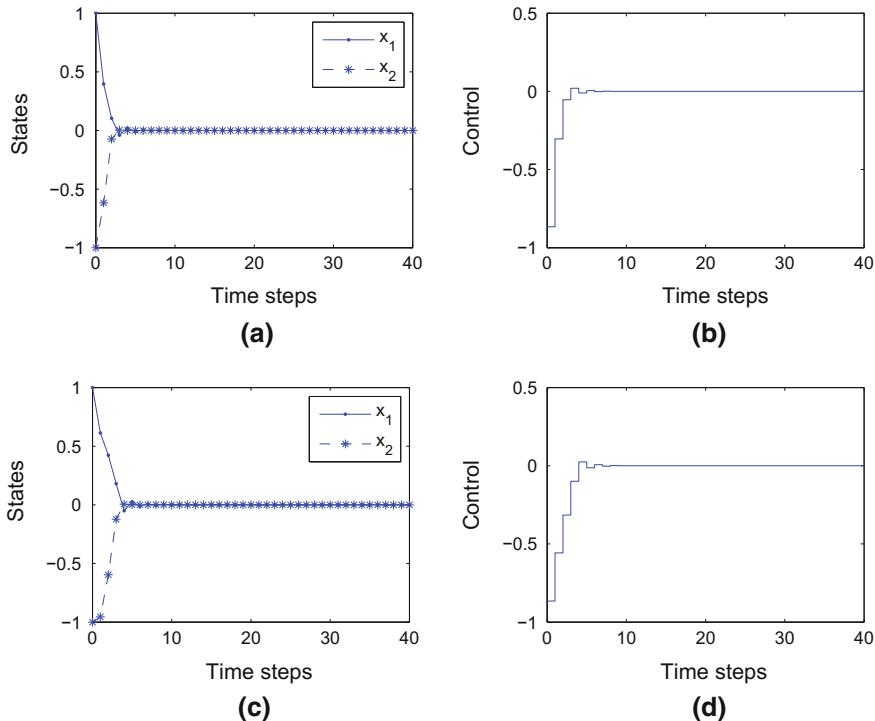


Fig. 3.8 The control and state trajectories. **a** State trajectories for $\xi = 10^{-8}$. **b** Control trajectory for $\xi = 10^{-8}$. **c** State trajectories for $\xi = 10^{-4}$. **d** Control trajectory for $\xi = 10^{-4}$

where Q and R are given as identity matrices with suitable dimensions. The initial state is $x_0 = [1, -1]^\top$.

The iterative ADP algorithm runs for 30 iteration steps to guarantee the convergence of the iterative value function. The curves of the admissible approximation error obtained by (3.3.39) and (3.3.41) are displayed in Figs. 3.5 and 3.6, respectively. From Fig. 3.6, we can see that for some states x_k , the admissible approximation error is smaller than zero so that the convergence criterion (3.3.41) is invalid. While from Fig. 3.5, we can see that the admissible approximation error curve is above zero which implies that the convergence criterion (3.3.39) is effective for all x_k .

To show the effectiveness of the numerical iterative ADP algorithm, we choose four different grid steps. Let $\xi = 10^{-8}, 10^{-4}, 10^{-2}, 10^{-1}$, respectively. The trajectories of the iterative value function are shown in Fig. 3.7a, b, c and d, respectively. For $\xi = 10^{-8}$ and $\xi = 10^{-4}$, implement the approximate optimal control for system (3.3.43), respectively. Let the implementation time be $T_f = 40$. The trajectories of the states and controls are displayed in Fig. 3.8a, b, c and d, respectively. When $\xi = 10^{-2}$, we can see that the iterative value function is not completely converged

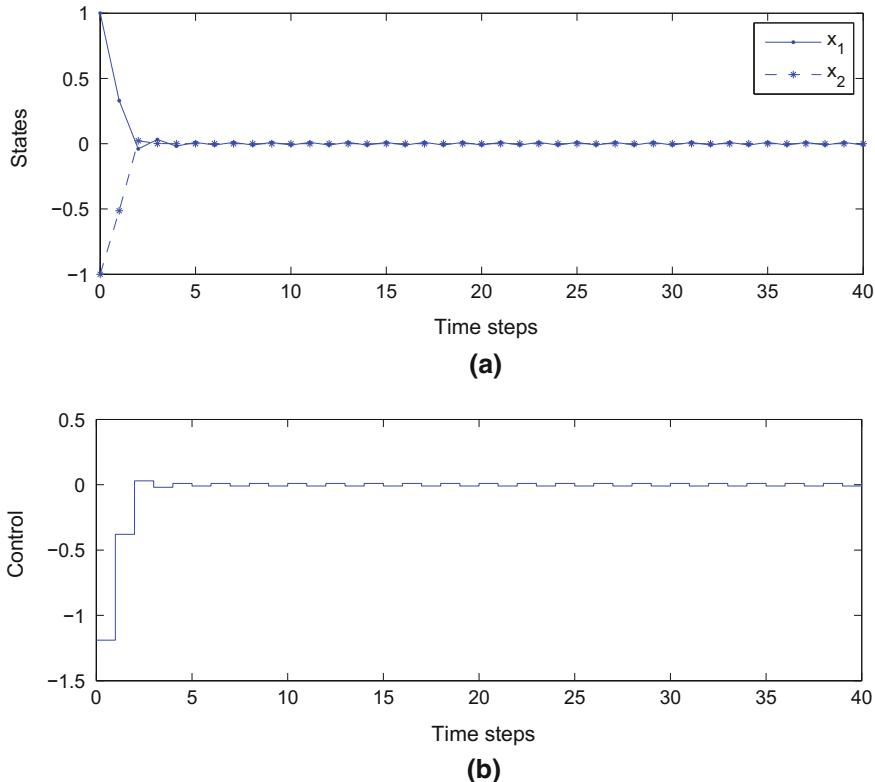


Fig. 3.9 The state and control trajectories. **a** State trajectories for $\varsigma = 10^{-2}$. **b** Control trajectory for $\varsigma = 10^{-2}$

within 30 iteration steps. The trajectories of the state are displayed in Fig. 3.9a, and the corresponding control trajectory is displayed in Fig. 3.9b.

In this section, it is shown that if the inequality (3.3.35) holds, then for $i = 0, 1, \dots$, the numerical iterative control $\hat{v}_i(x_k)$ stabilizes the system (3.3.43), which means that the numerical iterative θ -ADP algorithm can be implemented both online and offline. In Fig. 3.10a–d, we give the system state and control trajectories of the system (3.3.43) under the iterative control law $\hat{v}_0(x_k)$ with $\varsigma = 10^{-8}$ and $\varsigma = 10^{-4}$, respectively. In Fig. 3.11a–d, we give the system state and control trajectories of the system (3.3.43) under the iterative control law $\hat{v}_0(x_k)$ with $\varsigma = 10^{-2}$ and $\varsigma = 10^{-1}$, respectively. When $\varsigma = 10^{-1}$, we can see that the iterative value functions is not convergent. The control system is not stable.

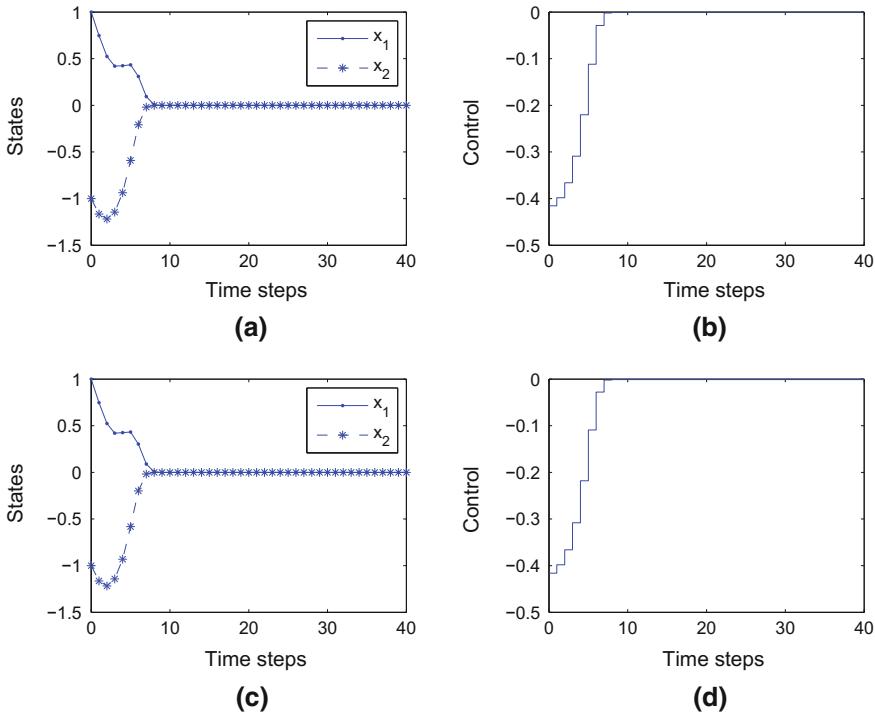


Fig. 3.10 The state and control trajectories under $\hat{v}_0(x_k)$. **a** State trajectories for $\varsigma = 10^{-8}$. **b** Control trajectory for $\varsigma = 10^{-8}$. **c** State trajectories for $\varsigma = 10^{-4}$. **d** Control trajectory for $\varsigma = 10^{-4}$

3.4 General Value Iteration ADP Algorithm with Finite Approximation Errors

Generally speaking, $J^*(x_k)$ given in (3.2.4) is a highly nonlinear and nonanalytic function. The optimal control law is nearly impossible to obtain by directly solving the Bellman equation. In this section, a GVI ADP algorithm will be developed to obtain optimal solution of the Bellman equation indirectly.

3.4.1 Derivation and Properties of the GVI Algorithm with Finite Approximation Errors

In this section, a finite-approximation-error-based general value iteration algorithm is developed. A new convergence analysis method will be established, and a new design method for the convergence criteria will also be developed [43].

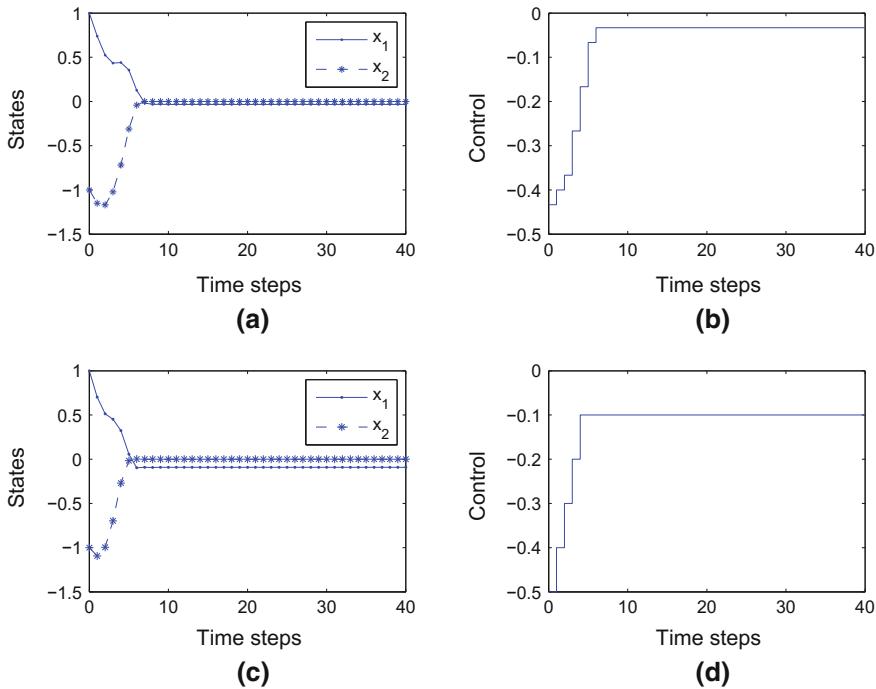


Fig. 3.11 The state and control trajectories under $\hat{v}_0(x_k)$. **a** State trajectories for $\varsigma = 10^{-2}$. **b** Control trajectory for $\varsigma = 10^{-2}$. **c** State trajectories for $\varsigma = 10^{-1}$. **d** Control trajectory for $\varsigma = 10^{-1}$

A. Derivation of the GVI Algorithm with Finite Approximation Errors

In the present GVI algorithm, the value function and control law are updated by iterations, with the iteration index i increasing from 0 to ∞ . Let Ω_x and Ω_u be the domain of definition for state and control, which are defined as $\Omega_x = \{x_k : x_k \in \mathbb{R}^n \text{ and } \|x_k\| < \infty\}$ and $\Omega_u = \{u_k : u_k \in \mathbb{R}^m \text{ and } \|u_k\| < \infty\}$, respectively, where $\|\cdot\|$ denotes the Euclidean norm. For all $x_k \in \Omega_x$, let the initial function $\Psi(x_k) \geq 0$ be an arbitrary positive semidefinite function. For all $x_k \in \Omega_x$, let the initial value function

$$\hat{V}_0(x_k) = \Psi(x_k). \quad (3.4.1)$$

The iterative control law $\hat{v}_0(x_k)$ can be computed as

$$\hat{v}_0(x_k) = \arg \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_0(x_{k+1}) \right\} + \rho_0(x_k), \quad (3.4.2)$$

where $\hat{V}_0(x_{k+1}) = \Psi(x_{k+1})$ and $\rho_0(x_k)$ is the approximation error function of the iterative control law $\hat{v}_0(x_k)$. For $i = 1, 2, \dots$, the iterative ADP algorithm will iterate between

$$\hat{V}_i(x_k) = U(x_k, \hat{v}_{i-1}(x_k)) + \hat{V}_{i-1}(F(x_k, \hat{v}_{i-1}(x_k))) + \pi_i(x_k) \quad (3.4.3)$$

and

$$\hat{v}_i(x_k) = \arg \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_i(x_{k+1}) \right\} + \rho_i(x_k), \quad (3.4.4)$$

where $\pi_i(x_k)$ and $\rho_i(x_k)$ are finite approximation error functions of the iterative value function and control law, respectively. For convenience of analysis, for $i = 0, 1, \dots$, we assume that $\rho_i(x_k) = 0$ and $\pi_i(x_k) = 0$ for $x_k = 0$.

B. Special Case of Accurate GVI Algorithm

Next, we will discuss the properties of GVI iterative ADP algorithm where the iterative value function and the iterative control law can accurately be obtained in each iteration. For $i = 0, 1, \dots$, the GVI algorithm is reduced to the following equations

$$\begin{cases} V_i(x_k) = \min_{u_k} \{U(x_k, u_k) + V_{i-1}(F(x_k, u_k))\}, \\ v_i(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + V_i(F(x_k, u_k))\}, \end{cases} \quad (3.4.5)$$

where $V_0(x_k) = \Psi(x_k)$ is a positive semidefinite function.

Remark 3.4.1 In [4], for $V_0(x_k) = 0$, it was proven that the iterative value function in (3.4.5) is monotonically nondecreasing and converges to the optimum. In [3, 5, 10], the convergence property of value iteration for discounted case has been investigated, and it was proven that the iterative value function will converge to the optimum if the discount factor satisfies $0 < \alpha < 1$. For the undiscounted cost function (3.2.2) and the positive semidefinite function, i.e., $\alpha \equiv 1$ and $V_0(x_k) = \Psi(x_k)$, the convergence analysis methods in [3–5, 7, 10, 18] are no longer applicable. Hence, a new convergence analysis method needs to be developed. In [11, 18, 26], a “function bound” method was proposed for the traditional value iteration algorithm with the zero initial value function. Based on the previous contribution of value iteration algorithms [3–5, 7, 10, 18], and inspired by [18], a new convergence analysis method for the general value iteration algorithm is developed in this section.

Lemma 3.4.1 *Let $\Psi(x_k) \geq 0$ be an arbitrary positive semidefinite function. Under Assumptions 3.2.1–3.2.3, for $i = 1, 2, \dots$, the iterative value function $V_i(x_k)$ in (3.4.5), is positive definite for all $x_k \in \Omega_x$.*

Theorem 3.4.1 *For $i = 0, 1, \dots$, let $V_i(x_k)$ and $v_i(x_k)$ be updated by (3.4.5). Then, the iterative value function $V_i(x_k)$ converges to $J^*(x_k)$ as $i \rightarrow \infty$.*

Proof First, according to Assumptions 3.2.1–3.2.3, $J^*(x_k)$ is a positive definite function of x_k . Let $\varepsilon > 0$ be an arbitrary small positive number. Define

$$\Omega_\varepsilon = \{x_k : x_k \in \Omega_x, \|x_k\| < \varepsilon\}.$$

For all $x_k \in \Omega_\varepsilon$, according to Lemma 3.4.1, for $i = 0, 1, \dots$, we have $V_i(x_k) \rightarrow J^*(x_k)$ as $\varepsilon \rightarrow 0$. Hence, the conclusion holds for all $x_k \in \Omega_\varepsilon$, $\varepsilon \rightarrow 0$. On the other hand, for an arbitrary small $\varepsilon > 0$, $x_k \in \Omega_x \setminus \Omega_\varepsilon$, and $u_k \in \Omega_u$, we have

$$0 < U(x_k, u_k) < \infty, \quad 0 \leq V_0(x_k) < \infty, \quad 0 < J^*(x_k) < \infty, \quad \text{and } 0 \leq J^*(F(x_k, u_k)) < \infty.$$

Hence, there exist constants γ , α , and β such that $J^*(F(x_k, u_k)) \leq \gamma U(x_k, u_k)$ and $\alpha J^*(x_k) \leq V_0(x_k) \leq \beta J^*(x_k)$, where $0 < \gamma < \infty$ and $0 \leq \alpha \leq 1 \leq \beta$, respectively. As $J^*(x_k)$ is unknown, the values of γ , α , and β cannot be obtained directly. We will first prove that for $i = 0, 1, \dots$, the following inequality

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

holds. Mathematical induction is introduced to prove the conclusion. Let $i = 0$. We have

$$\begin{aligned} V_1(x_k) &= \min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\} \\ &\geq \min_{u_k} \left\{ \left(1 + \gamma \frac{\alpha - 1}{1 + \gamma}\right) U(x_k, u_k) + \left(\alpha - \frac{\alpha - 1}{1 + \gamma}\right) J^*(x_{k+1}) \right\} \\ &= \left(1 + \frac{\alpha - 1}{(1 + \gamma^{-1})}\right) J^*(x_k). \end{aligned} \quad (3.4.6)$$

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

$$\begin{aligned} V_{l+1}(x_k) &= \min_{u_k} \{U(x_k, u_k) + V_l(x_{k+1})\} \\ &\geq \min_{u_k} \left\{ U(x_k, u_k) + \left(1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^{l-1}}\right) J^*(x_{k+1}) \right\} \\ &\geq \left(1 + \frac{\alpha - 1}{(1 + \gamma^{-1})^l}\right) J^*(x_k). \end{aligned} \quad (3.4.7)$$

The mathematical induction is complete. On the other hand, according to (3.4.6) and (3.4.7), for $i = 0, 1, \dots$, we can get

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

Letting $i \rightarrow \infty$, we can get the conclusion. The proof is complete.

C. Properties of the Finite-Approximation-Error-based GVI Algorithm

Considering approximation errors, we generally have $\hat{V}_i(x_k) \neq V_i(x_k)$, $\hat{v}_i(x_k) \neq v_i(x_k)$, $i = 0, 1, \dots$, and the convergence property in Theorem 3.4.1 for the accurate general value iteration algorithms becomes invalid. Hence, in this section, a new “design method of the convergence criteria” will be established. It permits the present general value iteration algorithm to design a suitable approximation error adaptively so that the iterative value function converges to a finite neighborhood of the optimal one.

Define the target iterative value function as

$$\Gamma_i(x_k) = \min_{u_k} \{U(x_k, u_k) + \hat{V}_{i-1}(x_{k+1})\}, \quad (3.4.8)$$

where

$$\hat{V}_0(x_k) = \Gamma_0(x_k) = \Psi(x_k).$$

For $i = 1, 2, \dots$, let $\eta_i > 0$ be a finite approximation error such that

$$\hat{V}_i(x_k) \leq \eta_i \Gamma_i(x_k). \quad (3.4.9)$$

From (3.4.9), we can see that the iterative value function $\Gamma_i(x_k)$ can be larger or smaller than $\hat{V}_i(x_k)$ and the convergence properties are different for different values of η_i . As the convergence analysis methods for $0 < \eta_i < 1$ and $\eta_i \geq 1$ are different, the convergence properties for different η_i will be discussed separately.

Theorem 3.4.2 *Let $x_k \in \Omega_x$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. For $i = 1, 2, \dots$, let $\Gamma_i(x_k)$ be expressed as in (3.4.8) and $\hat{V}_i(x_k)$ be expressed as in (3.4.4). If for $i = 1, 2, \dots$, there exists $0 < \eta_i < 1$ such that (3.4.9) holds, then the iterative value function converges to a bounded neighborhood of the optimal cost function $J^*(x_k)$.*

Proof If $0 < \eta_i < 1$, according to (3.4.9), we have $0 \leq \hat{V}_i(x_k) < \Gamma_i(x_k)$. Using mathematical induction, we can prove that for $i = 1, 2, \dots$, the following inequality

$$0 < \hat{V}_i(x_k) < V_i(x_k) \quad (3.4.10)$$

holds. According to Theorem 3.4.1, we have $V_i(x_k) \rightarrow J^*(x_k)$. Then, for $i = 0, 1, \dots$, $\hat{V}_i(x_k)$ is upper bounded and

$$0 < \lim_{i \rightarrow \infty} \hat{V}_i(x_k) \leq \lim_{i \rightarrow \infty} V_i(x_k) = J^*(x_k). \quad (3.4.11)$$

The proof is complete.

Theorem 3.4.3 Let $x_k \in \mathcal{Q}_x$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. For $i = 1, 2, \dots$, let $\Gamma_i(x_k)$ be expressed as (3.4.8) and $\hat{V}_i(x_k)$ be expressed as (3.4.4). Let $0 < \gamma_i < \infty$ be a constant such that

$$V_i(F(x_k, u_k)) \leq \gamma_i U(x_k, u_k). \quad (3.4.12)$$

If for $i = 1, 2, \dots$, there exists $1 \leq \eta_i < \infty$ such that (3.4.9) holds uniformly, then

$$\begin{aligned} \hat{V}_i(x_k) &\leq \eta_i \left[1 + \sum_{j=1}^{i-1} \left(\eta_{i-1} \eta_{i-2} \dots \eta_{i-j+1} (\eta_{i-j} - 1) \right. \right. \\ &\quad \times \left. \left. \frac{\gamma_{i-1} \gamma_{i-2} \dots \gamma_{i-j}}{(\gamma_{i-1} + 1)(\gamma_{i-2} + 1) \dots (\gamma_{i-j} + 1)} \right) \right] V_i(x_k), \end{aligned} \quad (3.4.13)$$

where we define $\sum_j^i (\cdot) = 0$ for all $j > i$, $i, j = 0, 1, \dots$, and

$$\eta_{i-1} \eta_{i-2} \dots \eta_{i-j+1} (\eta_{i-j} - 1) = (\eta_{i-1} - 1) \text{ for } j = 1.$$

Proof The theorem can be proven by mathematical induction. First, let $i = 1$. We can easily obtain $\Gamma_1(x_k) = V_1(x_k)$. According to (3.4.9), we have

$$\hat{V}_1(x_k) \leq \eta_1 V_1(x_k).$$

Thus, the conclusion holds for $i = 1$. Next, let $i = 2$. According to (3.4.8), we have

$$\begin{aligned} \Gamma_2(x_k) &= \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_1(F(x_k, u_k)) \right\} \\ &\leq \min_{u_k} \left\{ \left(1 + \gamma_1 \frac{\eta_1 - 1}{\gamma_1 + 1} \right) U(x_k, u_k) + \left(\eta_1 - \frac{\eta_1 - 1}{\gamma_1 + 1} \right) V_1(x_{k+1}) \right\} \\ &= \left(1 + \gamma_1 \frac{\eta_1 - 1}{\gamma_1 + 1} \right) \min_{u_k} \{ U(x_k, u_k) + V_1(x_{k+1}) \} \\ &= \left(1 + \gamma_1 \frac{\eta_1 - 1}{\gamma_1 + 1} \right) V_2(x_k). \end{aligned}$$

By (3.4.9), we can obtain

$$\hat{V}_2(x_k) \leq \eta_2 \left(1 + \gamma_1 \frac{\eta_1 - 1}{\gamma_1 + 1} \right) V_2(x_k),$$

which shows that (3.4.13) holds for $i = 2$. Assume that (3.4.13) holds for $i = l - 1$, where $l = 2, 3, \dots$. Then, for $i = l$, we obtain

$$\begin{aligned}
 \Gamma_l(x_k) &= \min_{u_k} \{U(x_k, u_k) + \hat{V}_{l-1}(F(x_k, u_k))\} \\
 &\leq \min_{u_k} \left\{ U(x_k, u_k) + \eta_{l-1} \left(1 + \sum_{j=1}^{l-2} \left(\eta_{l-2} \dots \eta_{l-j} (\eta_{l-j-1} - 1) \right. \right. \right. \\
 &\quad \times \left. \left. \left. \frac{\gamma_{l-2} \gamma_{l-3} \dots \gamma_{l-j-1}}{(\gamma_{l-3} + 1) \dots (\gamma_{l-j-1} + 1)} \right) \right) V_{l-1}(x_k) \right\} \\
 &\leq \left(1 + \sum_{j=1}^{l-1} (\eta_{l-1} \eta_{l-2} \dots \eta_{l-j+1} (\eta_{l-j} - 1)) \frac{\gamma_{l-1} \dots \gamma_{l-j}}{(\gamma_{l-1} + 1) \dots (\gamma_{l-j} + 1)} \right) \\
 &\quad \times \min_{u_k} \{U(x_k, u_k) + V_{l-1}(x_k)\} \\
 &= \left(1 + \sum_{j=1}^{l-1} (\eta_{l-1} \eta_{l-2} \dots \eta_{l-j+1} (\eta_{l-j} - 1)) \frac{\gamma_{l-1} \dots \gamma_{l-j}}{(\gamma_{l-1} + 1) \dots (\gamma_{l-j} + 1)} \right) V_l(x_k).
 \end{aligned}$$

Then, by (3.4.9), we can obtain (3.4.13) which proves the conclusion for $i = 0, 1, \dots$. The proof is complete.

From (3.4.13), we can see that for $i = 0, 1, \dots$, there exists an error between $\hat{V}_i(x_k)$ and $V_i(x_k)$. As $i \rightarrow \infty$, the bound of the approximation errors may increase to infinity. Thus, we will give the convergence properties of the iterative ADP algorithm (3.4.2)–(3.4.4) using error bound method. Before presenting the next theorem, the following lemma is necessary.

Lemma 3.4.2 *Let $\{b_i\}$, $i = 1, 2, \dots$, be a sequence of positive numbers. Let $0 < \lambda_i < \infty$ be a bounded positive constant for $i = 1, 2, \dots$ and let $a_i = \lambda_i b_i$. If $\sum_{i=1}^{\infty} b_i$ is finite, then $\sum_{i=1}^{\infty} a_i$ is finite.*

Proof Since for $i = 1, 2, \dots$, λ_i is finite, we can let $\bar{\lambda} = \sup\{\lambda_1, \lambda_2, \dots\}$. Then, $\sum_{i=1}^{\infty} a_i = \sum_{i=1}^{\infty} \lambda_i b_i \leq \bar{\lambda} \sum_{i=1}^{\infty} b_i$ is finite. The proof is complete.

Theorem 3.4.4 *Let $x_k \in \Omega_x$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. If for $i = 1, 2, \dots$, $\hat{V}_i(x_k)$ satisfies (3.4.13) and the approximation error η_i satisfies*

$$1 \leq \eta_i \leq q_{i-1} \frac{\gamma_{i-1} + 1}{\gamma_{i-1}}, \quad (3.4.14)$$

where q_i is an arbitrary constant such that $\frac{\gamma_i}{\gamma_i + 1} < q_i < 1$, then as $i \rightarrow \infty$, the iterative value function $\hat{V}_i(x_k)$ of the general value iteration algorithm (3.4.2)–(3.4.4) converges to a bounded neighborhood of the optimal cost function $J^*(x_k)$.

Proof For (3.4.13) in Theorem 3.4.3, if we let

$$\Delta_i = \sum_{j=1}^{i-1} \left(\eta_{i-1} \eta_{i-2} \dots \eta_{i-j+1} (\eta_{i-j} - 1) \frac{\gamma_{i-1} \gamma_{i-2} \dots \gamma_{i-j}}{(\gamma_{i-1} + 1)(\gamma_{i-2} + 1) \dots (\gamma_{i-j} + 1)} \right),$$

then we can get

$$\begin{aligned} \Delta_i = & \sum_{j=1}^{i-1} \left(\frac{\eta_{i-1} \eta_{i-2} \dots \eta_{i-j} \gamma_{i-1} \gamma_{i-2} \dots \gamma_{i-j}}{(\gamma_{i-1} + 1)(\gamma_{i-2} + 1) \dots (\gamma_{i-j} + 1)} \right), \\ & - \sum_{j=1}^{i-1} \left(\frac{\eta_{i-1} \eta_{i-2} \dots \eta_{i-j+1} \gamma_{i-1} \gamma_{i-2} \dots \gamma_{i-j}}{(\gamma_{i-1} + 1)(\gamma_{i-2} + 1) \dots (\gamma_{i-j} + 1)} \right), \end{aligned} \quad (3.4.15)$$

where we define $\eta_{i-1} \eta_{i-2} \dots \eta_{i-j+1} = 1$ for $j = 1$. If we let

$$\begin{aligned} a_{ij} &= \frac{\eta_{i-1} \eta_{i-2} \dots \eta_{i-j} \gamma_{i-1} \gamma_{i-2} \dots \gamma_{i-j}}{(\gamma_{i-1} + 1)(\gamma_{i-2} + 1) \dots (\gamma_{i-j} + 1)}, \\ b_{ij} &= \frac{\eta_{i-1} \eta_{i-2} \dots \eta_{i-j+1} \gamma_{i-1} \gamma_{i-2} \dots \gamma_{i-j}}{(\gamma_{i-1} + 1)(\gamma_{i-2} + 1) \dots (\gamma_{i-j} + 1)}, \end{aligned} \quad (3.4.16)$$

where $i = 1, 2, \dots$, and $j = 1, 2, \dots, i-1$, then $\Delta_i = \sum_{j=1}^{i-1} a_{ij} - \sum_{j=1}^{i-1} b_{ij}$. If $\sum_{j=1}^{i-1} a_{ij}$

and $\sum_{j=1}^{i-1} b_{ij}$ are both finite as $i \rightarrow \infty$, then $\lim_{i \rightarrow \infty} \Delta_i$ is finite. According to (3.4.16),

we have $\frac{b_{ij}}{b_{i(j-1)}} = \frac{\eta_{i-j+1} \gamma_{i-j}}{(\gamma_{i-j} + 1)}$. If $\frac{b_{ij}}{b_{i(j-1)}} \leq q_{i-j} < 1$, then we can get $\eta_{i-j+1} \leq q_{i-j} \frac{\gamma_{i-j} + 1}{\gamma_{i-j}}$. Let $\ell = i - j$. We can obtain $\eta_{\ell+1} \leq q_{\ell} \frac{\gamma_{\ell} + 1}{\gamma_{\ell}}$, where $\ell = 1, 2, \dots, i-1$. Letting $i \rightarrow \infty$, we can obtain (3.4.14).

Let $q = \sup\{q_1, q_2, \dots\}$, where $0 < q < 1$. We can get $b_{ij} \leq \left(\frac{\gamma_{i-1} + 1}{\gamma_{i-1}} \right) q^{j-1}$. Thus, we can obtain

$$\sum_{j=1}^{i-1} b_{ij} \leq \sum_{j=1}^{i-1} \left(\frac{\gamma_{i-1} + 1}{\gamma_{i-1}} \right) q^{j-1}.$$

As $\gamma_i/(\gamma_i + 1) < q < 1$ and γ_{i-1} is finite for $i = 1, 2, \dots$, letting $i \rightarrow \infty$, $\lim_{i \rightarrow \infty} \sum_{j=1}^{i-1} b_{ij}$ is finite. On the other hand, for $i = 1, 2, \dots$, and for $j = 1, 2, \dots, i-1$, we have

$a_{ij} = \eta_{i-j} b_{ij}$. As for $i = 1, 2, \dots$ and for $j = 1, 2, \dots, i-1$, $1 \leq \eta_{i-j} < \infty$ is finite, according to Lemma 3.4.2, $\lim_{i \rightarrow \infty} \sum_{j=1}^{i-1} a_{ij}$ must be finite. Therefore, $\lim_{i \rightarrow \infty} \Delta_i$ is finite. According to Theorem 3.4.1, we have $\lim_{i \rightarrow \infty} V_i(x_k) = J^*(x_k)$. Hence, the iterative value function $\hat{V}_i(x_k)$ converges to a bounded neighborhood of the optimal cost function $J^*(x_k)$. The proof is complete.

Remark 3.4.2 From Theorem 3.4.4, we can see that there is no constraint for the approximation error η_1 in (3.4.14). If we let the parameter γ_0 satisfy

$$V_0(F(x_k, u_k)) \leq \gamma_0 U(x_k, u_k), \quad (3.4.17)$$

where $V_0(F(x_k, u_k)) = \Psi(F(x_k, u_k))$ and let the approximation error η_1 satisfy $1 \leq \eta_1 < (\gamma_0 + 1)/\gamma_0$, then the convergence criterion (3.4.14) is valid for $i = 0, 1, \dots$

Combining Theorems 3.4.2 and 3.4.4, the convergence criterion of the finite-approximation-error-based general value iteration algorithm can be established.

Theorem 3.4.5 *Let $x_k \in \Omega_x$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. If for $i = 0, 1, \dots$, inequality*

$$0 < \eta_i \leq q_{i-1} \frac{\gamma_{i-1} + 1}{\gamma_{i-1}} \quad (3.4.18)$$

holds, where $0 < q_i < 1$ is an arbitrary constant, then as $i \rightarrow \infty$, the iterative value function $\hat{V}_i(x_k)$ converges to a bounded neighborhood of the optimal cost function $J^(x_k)$.*

3.4.2 Designs of Convergence Criteria with Finite Approximation Errors

In the previous section, we have discussed the convergence property of the general value iteration algorithm. From (3.4.18), for $i = 0, 1, \dots$, the approximation error η_i is a function of γ_i . Hence, if we can obtain the parameter γ_i for $i = 1, 2, \dots$, then we can design an iterative approximation error η_{i+1} to guarantee the iterative value function $\hat{V}_i(x_k)$ to be convergent. Next, the detailed design method of the convergence criteria will be discussed.

According to (3.4.12), we give the following definition. Define Ω_{γ_i} as

$$\Omega_{\gamma_i} = \{\gamma_i : \gamma_i U(x_k, u_k) \geq V_i(F(x_k, u_k)), \forall x_k \in \Omega_x, \forall u_k \in \Omega_u\}.$$

As the existence of approximation errors, the accurate iterative value function $V_i(x_k)$ cannot be obtained in general. Hence, the parameter γ_i cannot be obtained directly

from (3.4.12). In this section, several methods will be introduced on how to estimate γ_i .

Lemma 3.4.3 *Let $v_i(x_k)$ be an arbitrary control law. Define $V_i(x_k)$ as in (3.4.5) and define $\Lambda_i(x_k)$ as*

$$\Lambda_i(x_k) = U(x_k, v_{i-1}(x_k)) + \Lambda_{i-1}(x_{k+1}). \quad (3.4.19)$$

If $V_0(x_k) = \Lambda_0(x_k) = \Psi(x_k)$, then for $i = 1, 2, \dots$, $V_i(x_k) \leq \Lambda_i(x_k)$.

Theorem 3.4.6 *Let $\mu(x_k)$ be an arbitrary admissible control law of nonlinear system (3.2.1). Let $P_i(x_k)$ be the iterative value function satisfying*

$$P_i(x_k) = U(x_k, \mu(x_k)) + P_{i-1}(x_{k+1}) \quad (3.4.20)$$

where $P_0(x_k) = V_0(x_k) = \Psi(x_k)$. If there exists a constant $\tilde{\gamma}_i$ such that

$$\tilde{\gamma}_i U(x_k, u_k) \geq P_i(F(x_k, u_k)), \quad (3.4.21)$$

then $\tilde{\gamma}_i \in \Omega_{\gamma_i}$.

Proof As $\mu(x_k)$ is an arbitrary admissible control law, according to Lemma 3.4.3, we have $P_i(x_k) \geq V_i(x_k)$. If $\tilde{\gamma}_i$ satisfies (3.4.21), then we can get

$$\tilde{\gamma}_i U(x_k, u_k) \geq P_i(F(x_k, u_k)) \geq V_i(F(x_k, u_k)).$$

The proof is complete.

From Theorem 3.4.6, the first estimation method of the parameter γ_i can be described in Algorithm 3.4.1.

Algorithm 3.4.1 Method I for estimating γ_i

- 1: Give an arbitrary admissible control law $\mu(x_k)$ for nonlinear system (3.2.1).
 - 2: Construct an iterative value function $P_i(x_k)$ by (3.4.20), $i = 1, 2, \dots$, where $P_0(x_k) = V_0(x_k) = \Psi(x_k)$.
 - 3: Compute the constant $\tilde{\gamma}_i$ by (3.4.21).
 - 4: Return $\gamma_i = \tilde{\gamma}_i$.
-

Lemma 3.4.4 *For $i = 1, 2, \dots$, let $\Gamma_i(x_k)$ be expressed as in (3.4.8) and $\hat{V}_i(x_k)$ be expressed as in (3.4.3). If for $i = 1, 2, \dots$,*

$$\hat{V}_i(x_k) \geq \Gamma_i(x_k), \quad (3.4.22)$$

then

$$\hat{V}_i(x_k) \geq V_i(x_k),$$

where $V_i(x_k)$ is defined in (3.4.5).

Proof This conclusion can be proven by mathematical induction. First, consider $i = 1$. We have

$$\hat{V}_1(x_k) \geq \Gamma_1(x_k) = \min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\} = V_1(x_k).$$

So, the conclusion holds for $i = 1$. Assume that the conclusion holds for $i = l - 1$, $l = 2, 3, \dots$. Then, for $i = l$, we can obtain

$$\begin{aligned}\hat{V}_l(x_k) &\geq \Gamma_l(x_k) \\ &= \min_{u_k} \left\{ U(x_k, u_k) + \hat{V}_{l-1}(x_{k+1}) \right\} \\ &\geq \min_{u_k} \{U(x_k, u_k) + V_{l-1}(x_{k+1})\} \\ &= V_l(x_k).\end{aligned}$$

The proof is complete.

Now, we can derive the following theorem.

Theorem 3.4.7 For $i = 0, 1, \dots$, $\hat{V}_i(x_k)$ is obtained by (3.4.2)–(3.4.4). If for $i = 0, 1, \dots$, the approximation error of the iterative value function $\pi_i(x_k) \geq 0$, and there exists $\hat{\gamma}_i$ such that

$$\hat{\gamma}_i U(x_k, u_k) \geq \hat{V}_i(F(x_k, u_k)), \quad (3.4.23)$$

then $\hat{\gamma}_i \in \Omega_{\gamma_i}$.

Proof According to the general value iteration algorithm (3.4.2)–(3.4.4). As the existence of the approximation error $\rho_0(x_k)$ in the iterative control law, for $i = 1$, we have

$$\Gamma_1(x_k) = \min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\} \leq U(x_k, \hat{v}_0(x_k)) + \hat{V}_0(x_{k+1}),$$

where $\hat{V}_0(x_k) = \Gamma_0(x_k) = \Psi(x_k)$. If $\pi_0(x_k) \geq 0$, then we can get $\hat{V}_1(x_k) \geq V_1(x_k)$. Using the mathematical induction, we can prove that $\hat{V}_i(x_k) \geq V_i(x_k)$ holds for $i = 1, 2, \dots$. According to (3.4.23), we can get $\hat{\gamma}_i U(x_k, u_k) \geq V_i(F(x_k, u_k))$, which proves $\hat{\gamma}_i \in \Omega_{\gamma_i}$. This completes the proof of the theorem.

From Theorem 3.4.7, we can see that if for $i = 0, 1, \dots$, we can force the approximation error $\pi_i(x_k) \geq 0$, then the parameter γ_i can be estimated. An effective method is to add another approximation error $|\pi_i(x_k)|$ to the iterative value function $\hat{V}_{i+1}(x_k)$. The new iterative value function can be defined as

$$\begin{aligned}\hat{\mathcal{V}}_i(x_k) &= U(x_k, \hat{\nu}_{i-1}(x_k)) + \hat{V}_{i-1}(x_{k+1}) + \pi_i(x_k) + |\pi_i(x_k)| \\ &= \hat{V}_i(x_k) + |\pi_i(x_k)|.\end{aligned}\quad (3.4.24)$$

Hence, the second estimation method of the parameter γ_i can be described in Algorithm 3.4.2.

Algorithm 3.4.2 Method II for estimating γ_i

- 1: For $i = 1, 2, \dots$, obtain $\hat{V}_i(x_k)$ by (3.4.3).
 - 2: Obtain the iterative value function $\hat{\mathcal{V}}_i(x_k)$ by (3.4.24).
 - 3: Let $\hat{V}_i(x_k) = \hat{\mathcal{V}}_i(x_k)$.
 - 4: Obtain the parameter $\hat{\gamma}_i$ by (3.4.23).
 - 5: Return $\gamma_i = \hat{\gamma}_i$.
-

For $i = 1, 2, \dots$, there exists a finite constant $0 < \underline{\eta}_i \leq 1$ such that

$$\hat{V}_i(x_k) \geq \underline{\eta}_i \Gamma_i(x_k). \quad (3.4.25)$$

If $\hat{V}_i(x_k) \geq \Gamma_i(x_k)$, then we define $\underline{\eta}_i = 1$. We can derive the following results.

Lemma 3.4.5 *Let $x_k \in \Omega_x$ be an arbitrary controllable state and Assumptions 3.2.1–3.2.3 hold. Let $\hat{V}_0(x_k) = V_0(x_k) = \Psi(x_k)$. For $i = 1, 2, \dots$, let $\Gamma_i(x_k)$ be expressed as in (3.4.8) and $\hat{V}_i(x_k)$ be expressed as in (3.4.4). Let $0 < \gamma_i < \infty$ be a constant such that (3.4.12). If for $i = 1, 2, \dots$, there exists $0 < \underline{\eta}_i \leq 1$ such that (3.4.25), then*

$$\begin{aligned}\hat{V}_i(x_k) &\geq \underline{\eta}_i \left[1 + \sum_{j=1}^{i-1} \left(\underline{\eta}_{i-1} \underline{\eta}_{i-2} \dots \underline{\eta}_{i-j+1} (\underline{\eta}_{i-j} - 1) \right. \right. \\ &\quad \left. \left. \times \frac{\gamma_{i-1} \gamma_{i-2} \dots \gamma_{i-j}}{(\gamma_{i-1} + 1)(\gamma_{i-2} + 1) \dots (\gamma_{i-j} + 1)} \right) \right] V_i(x_k),\end{aligned}$$

where we define $\sum_j^i (\cdot) = 0$, for all $j > i$, $i, j = 0, 1, \dots$, and we let

$$\underline{\eta}_{i-1} \underline{\eta}_{i-2} \dots \underline{\eta}_{i-j+1} (\underline{\eta}_{i-j} - 1) = (\underline{\eta}_{i-1} - 1), \text{ for } j = 1.$$

Theorem 3.4.8 *For $i = 1, 2, \dots$, let $0 < \gamma_i < \infty$ be a constant such that (3.4.12). Define a new iterative value function as*

$$\underline{V}_i(x_k) = \underline{\eta}_i \left[1 + \sum_{j=1}^{i-1} \left(\underline{\eta}_{i-1} \underline{\eta}_{i-2} \cdots \underline{\eta}_{i-j+1} (\underline{\eta}_{i-j} - 1) \right. \right. \\ \left. \left. \times \frac{\gamma_{i-1} \gamma_{i-2} \cdots \gamma_{i-j}}{(\gamma_{i-1} + 1)(\gamma_{i-2} + 1) \cdots (\gamma_{i-j} + 1)} \right) \right] V_i(x_k),$$

where $0 < \underline{\eta}_i \leq 1$. Then, for $i = 1, 2, \dots$, we have

$$0 < \underline{V}_i(x_k) \leq V_i(x_k). \quad (3.4.26)$$

Proof The conclusion can be proven by mathematical induction. First, let $i = 1$. We have $\underline{V}_1(x_k) \leq \underline{\eta}_1 V_1(x_k)$ which shows (3.4.26) holds for $i = 1$. Assume that the inequality (3.4.26) holds for $i = l - 1, l = 2, 3, \dots$. Then, for $i = l$, we can get

$$\underline{\eta}_l \left[1 + \sum_{j=1}^{l-1} \left(\underline{\eta}_{l-1} \underline{\eta}_{l-2} \cdots \underline{\eta}_{l-j+1} (\underline{\eta}_{l-j} - 1) \frac{\gamma_{l-1} \gamma_{l-2} \cdots \gamma_{l-j}}{(\gamma_{l-1} + 1)(\gamma_{l-2} + 1) \cdots (\gamma_{l-j} + 1)} \right) \right] \\ = \underline{\eta}_l \left[1 + \frac{\gamma_{l-1}}{\gamma_{l-1} + 1} \left(\underline{\eta}_{l-1} \left(1 + \sum_{j=1}^{l-2} \left(\underline{\eta}_{l-2} \underline{\eta}_{l-3} \cdots \underline{\eta}_{l-j} \right. \right. \right. \right. \\ \left. \left. \left. \times \frac{(\underline{\eta}_{l-j-1} - 1) \gamma_{l-2} \gamma_{l-3} \cdots \gamma_{l-j-1}}{(\gamma_{l-2} + 1)(\gamma_{l-3} + 1) \cdots (\gamma_{l-j-1} + 1)} \right) - 1 \right) \right). \quad (3.4.27)$$

As for $i = 1, 2, \dots, 0 < \underline{\eta}_i \leq 1$ and $0 < \gamma_i < \infty$, we have

$$0 < \underline{\eta}_{l-1} \left(1 + \sum_{j=1}^{l-2} \left(\underline{\eta}_{l-2} \underline{\eta}_{l-3} \cdots \underline{\eta}_{l-j} (\underline{\eta}_{l-j-1} - 1) \right. \right. \\ \left. \left. \times \frac{\gamma_{l-2} \gamma_{l-3} \cdots \gamma_{l-j-1}}{(\gamma_{l-2} + 1)(\gamma_{l-3} + 1) \cdots (\gamma_{l-j-1} + 1)} \right) \right) \\ \leq 1. \quad (3.4.28)$$

Substituting (3.4.28) into (3.4.27), we can obtain the conclusion for $i = l$. The proof is complete.

From Theorem 3.4.8, we can estimate the lower bound of the iterative value function $V_i(x_k)$. Hence, we can derive the following theorem.

Theorem 3.4.9 For $i = 1, 2, \dots$, let $\hat{V}_i(x_k)$ be the iterative value function such that (3.4.22). If for $i = 1, 2, \dots$, there exists $\hat{\gamma}_i$ such that (3.4.23), then

$$\underline{\gamma}_i = \frac{\hat{\gamma}_i}{\underline{\Delta}_i} \in \Omega_{\gamma_i}, \quad (3.4.29)$$

where

$$\underline{\Delta}_i = \underline{\eta}_i \left[1 + \sum_{j=1}^{i-1} \left(\underline{\eta}_{i-1} \underline{\eta}_{i-2} \cdots \underline{\eta}_{i-j+1} (\underline{\eta}_{i-j} - 1) \right. \right. \\ \left. \left. \times \frac{\underline{\gamma}_{i-1} \underline{\gamma}_{i-2} \cdots \underline{\gamma}_{i-j}}{(\underline{\gamma}_{i-1} + 1)(\underline{\gamma}_{i-2} + 1) \cdots (\underline{\gamma}_{i-j} + 1)} \right) \right]. \quad (3.4.30)$$

Proof The conclusion can be proven by mathematical induction. First, let $i = 1$. According to (3.4.30), we have

$$\hat{V}_1 U(x_k, u_k) \geq \hat{V}_1(F(x_k, u_k)) \geq \underline{\eta}_1 V_1(F(x_k, u_k)).$$

Assume that the conclusion holds for $i = l - 1$, $l = 2, 3, \dots$. For $i = l$, according to Lemma 3.4.5, we can get $\hat{V}_l(x_k) \geq \underline{\Delta}_l V_l(x_k)$. From (3.4.23), we can obtain $\hat{V}_l U(x_k, u_k) \geq \hat{V}_l(F(x_k, u_k)) \geq \underline{\Delta}_l V_l(F(x_k, u_k))$, which shows that $\underline{\gamma}_l = \frac{\hat{V}_l}{\underline{\Delta}_l} \in \Omega_{\gamma_l}$. The proof is complete.

The third estimation method of the parameter γ_i is described in Algorithm 3.4.3.

Algorithm 3.4.3 Method III for estimating γ_i

- 1: For $i = 1, 2, \dots$, record $\underline{\eta}_0, \underline{\eta}_1, \dots, \underline{\eta}_{i-1}$ and $\underline{\gamma}_0, \underline{\gamma}_1, \dots, \underline{\gamma}_{i-1}$.
 - 2: Obtain $\hat{V}_i(x_k)$ and $\Gamma_i(x_k)$ by (3.4.3) and by (3.4.8), respectively.
 - 3: If $\hat{V}_i(x_k) \geq \Gamma_i(x_k)$, then let $\underline{\eta}_i = 1$; else, obtain the approximation error $\underline{\eta}_i$ by (3.4.25).
 - 4: Obtain $\hat{\gamma}_i$ according to $\hat{V}_i(x_k)$ and $U(x_k, u_k)$. Solve $\underline{\gamma}_i$ by (3.4.29).
 - 5: Return $\gamma_i = \underline{\gamma}_i$.
-

Remark 3.4.3 We can see that if we record $\underline{\eta}_j$ and $\underline{\gamma}_j$, $j = 1, 2, \dots, i-2$, then we can compute $\underline{\Delta}_{i-1}$ by (3.4.30). Then, we can obtain $\underline{\gamma}_{i-1}$ by (3.4.29). For $i = 2, 3, \dots$, the convergence criterion (3.4.18) can be verified. Hence, it is unnecessary to construct a new iterative value function to estimate the parameter γ_i . On the other hand, from the expression of (3.4.30), $0 < \underline{\Delta}_i \leq 1$ holds for $i = 1, 2, \dots$. It shows that the convergence criterion (3.4.29) is feasible. Therefore, it is recommended to use Algorithm 3.4.3 to estimate γ_i . In this section, we will give comparisons of these three methods.

Now, we summarize the finite-approximation-error-based general value iteration algorithm in Algorithm 3.4.4.

Algorithm 3.4.4 General value iteration algorithm with finite approximation errors**Initialization:**

Choose randomly an array of system states x_k in Ω_x . Choose a semi-positive definite function $\Psi(x_k) \geq 0$. Choose a convergence precision ε . Give a sequence $\{\eta_i\}$, $i = 0, 1, \dots$, where $0 < q_i < 1$. Give two constants $0 < \varsigma < 1$, $0 < \xi < 1$.

Iteration:

- Step 1. Let the iteration index $i = 0$.
- Step 2. Let $V_0(x_k) = \Psi(x_k)$ and obtain γ_0 by (3.4.17). Compute $\hat{v}_0(x_k)$ by (3.4.2).
- Step 3. Let $i = i + 1$.
- Step 4. Compute $\hat{V}_i(x_k)$ by (3.4.3) and obtain $\hat{v}_i(x_k)$ by (3.4.4).
- Step 5. Obtain η_i by (3.4.9). If $\eta_i(x_k)$ satisfies (3.4.14), then estimate γ_i by Algorithm Υ , $\Upsilon = 3.4.1, 3.4.2$ or $3.4.3$, and goto next step. Otherwise, decrease $\rho_{i-1}(x_k)$ and $\pi_{i-1}(x_k)$, i.e., $\rho_{i-1}(x_k) = \varsigma \rho_{i-1}(x_k)$ and $\pi_{i-1}(x_k) = \xi \pi_{i-1}(x_k)$, respectively, e.g., by further NN training. Goto Step 4.
- Step 6. If $|\hat{V}_i(x_k) - \hat{V}_{i-1}(x_k)| \leq \varepsilon$, then the optimal cost function is obtained and goto Step 7; else, goto Step 3.
- Step 7. Return $\hat{V}_i(x_k)$ and $\hat{v}_i(x_k)$.

Remark 3.4.4 Generally speaking, in iterative ADP algorithms, the difference between $\hat{V}_i(x_k)$ and $\Gamma_i(x_k)$ is obtained, i.e.,

$$\hat{V}_i(x_k) - \Gamma_i(x_k) = \varepsilon_i(x_k),$$

where $\varepsilon_i(x_k)$ is the approximation error function. According to the definition of η_i in (3.4.9) and the convergence criterion (3.4.14), we can easily obtain the following equivalent convergence criterion

$$\varepsilon_i(x_k) \leq \frac{1}{\gamma_{i-1} + 1} \hat{V}_i(x_k). \quad (3.4.31)$$

From (3.4.31), we can see that for different system state x_k , it requires different approximation error ε_i to guarantee the convergence of $\hat{V}_i(x_k)$. If $\|x_k\|$ is large, then the present general value iteration algorithm permits large approximation errors to converge, and if $\|x_k\|$ is small, then small approximation errors are required to ensure the convergence of the general value iteration algorithm. It is an important property for the neural network implementation of the finite-approximation-error-based general value iteration algorithm. For large $\|x_k\|$, approximation errors of neural networks can be large. As $\|x_k\| \rightarrow 0$, the approximation errors of neural networks are also required to be zero. As is known, the approximation of neural networks cannot be globally accurate with no approximation errors, so the implementation of the general value iteration algorithm by neural networks may be invalid as $x_k \rightarrow 0$. On the other hand, in the real world, neural networks are generally trained under a global uniform training precision or approximation error. Thus, the present general value iteration algorithm requires that the training precision is high and the approximation errors are small which guarantee the iterative value function convergent for most of the state space.

3.4.3 Simulation Study

In this section, we examine the performance of the present algorithm in a torsional pendulum system [22, 23, 28]. The dynamics of the pendulum is as follows

$$\begin{cases} \frac{d\theta}{dt} = \omega, \\ J \frac{d\omega}{dt} = u - Mgl \sin \theta - f_d \frac{d\theta}{dt}, \end{cases}$$

where $M = 1/3$ kg and $l = 2/3$ m are the mass and length of the pendulum bar, respectively. The system states are the current angle θ and the angular velocity ω . Let $J = 4/3Ml^2$ and $f_d = 0.2$ be the rotary inertia and frictional factor, respectively. Let $g = 9.8$ m/s² be the gravity. Discretization of the system function and value function

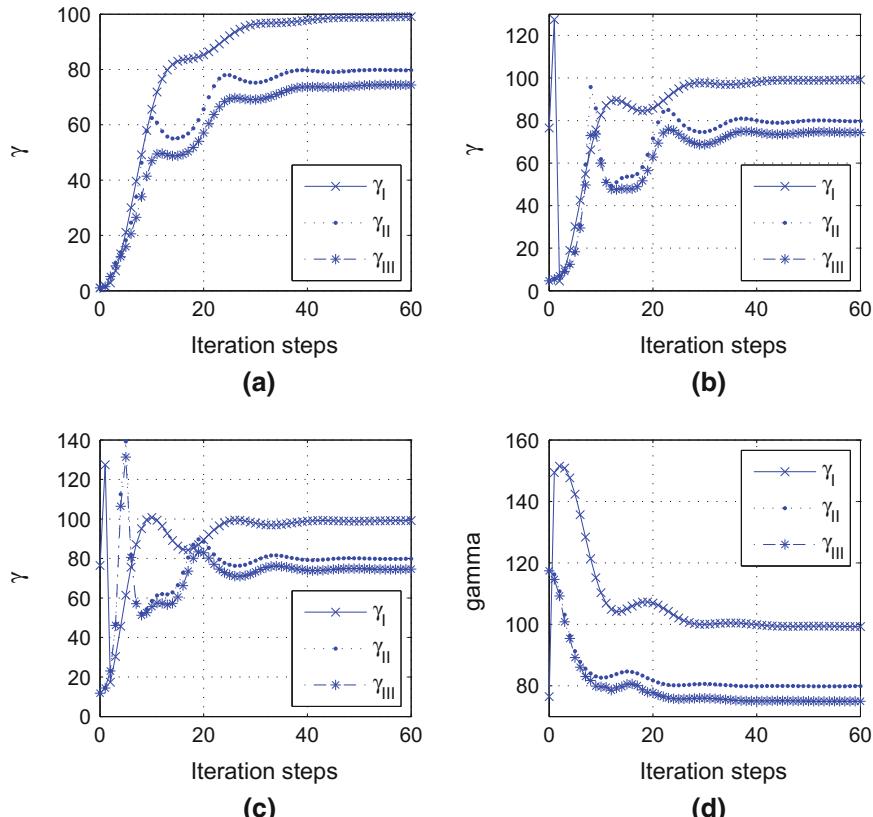


Fig. 3.12 The trajectories of γ 's by $\Psi^1(x_k)$ – $\Psi^4(x_k)$. **a** γ_1 , γ_{II} , and γ_{III} by $\Psi^1(x_k)$. **b** γ_1 , γ_{II} , and γ_{III} by $\Psi^2(x_k)$. **c** γ_1 , γ_{II} , and γ_{III} by $\Psi^3(x_k)$. **d** γ_1 , γ_{II} , and γ_{III} by $\Psi^4(x_k)$

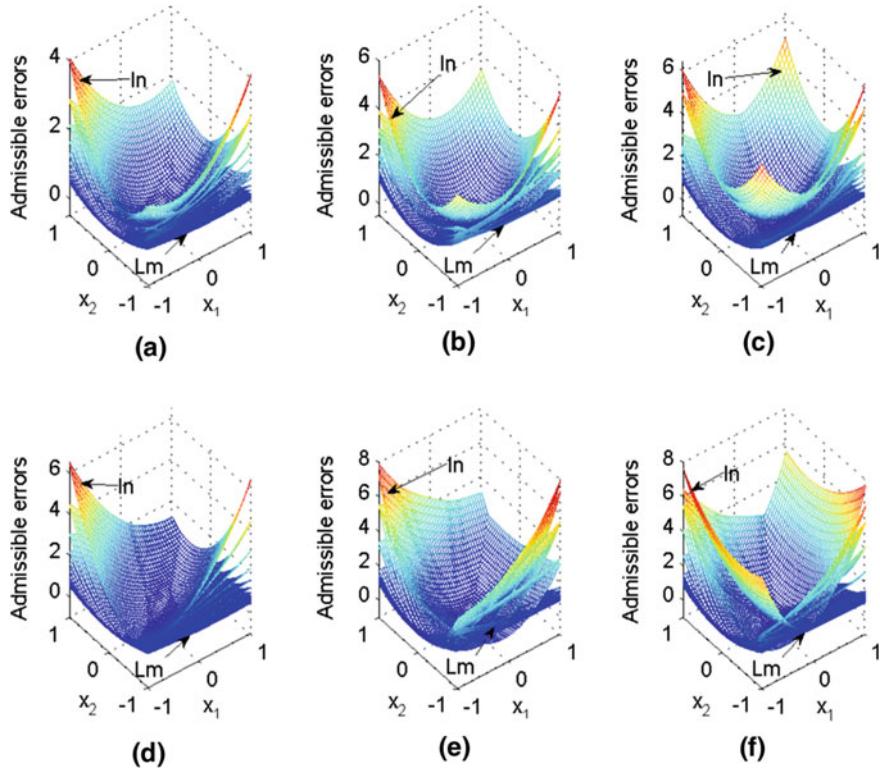


Fig. 3.13 The admissible errors corresponding to $\Psi^1(x_k)$ and $\Psi^2(x_k)$. **a** Admissible errors by γ_1 and $\Psi^1(x_k)$. **b** Admissible errors by γ_{II} and $\Psi^1(x_k)$. **c** Admissible errors by γ_{III} and $\Psi^1(x_k)$. **d** Admissible errors by γ_1 and $\Psi^2(x_k)$. **e** Admissible errors by γ_{II} and $\Psi^2(x_k)$. **f** Admissible errors by γ_{III} and $\Psi^2(x_k)$

using Euler and trapezoidal methods with the sampling interval $\Delta t = 0.1$ sec leads to

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

where $x_{1k} = \theta_k$ and $x_{2k} = \omega_k$. Let the initial state be $x_0 = [1, -1]^T$. We choose 10000 states in Ω_x . The critic and the action networks are also chosen as three-layer backpropagation (BP) neural networks, where the structures are set as 2–12–1 and 2–12–1, respectively.

To illustrate the effectiveness of the present algorithm, we also choose four different initial value functions which are expressed by $\Psi^j(x_k) = x_k^T P_j x_k$, $j = 1, \dots, 4$. Let $P_1 = 0$. Let P_2-P_4 be initialized by positive definite matrices given by

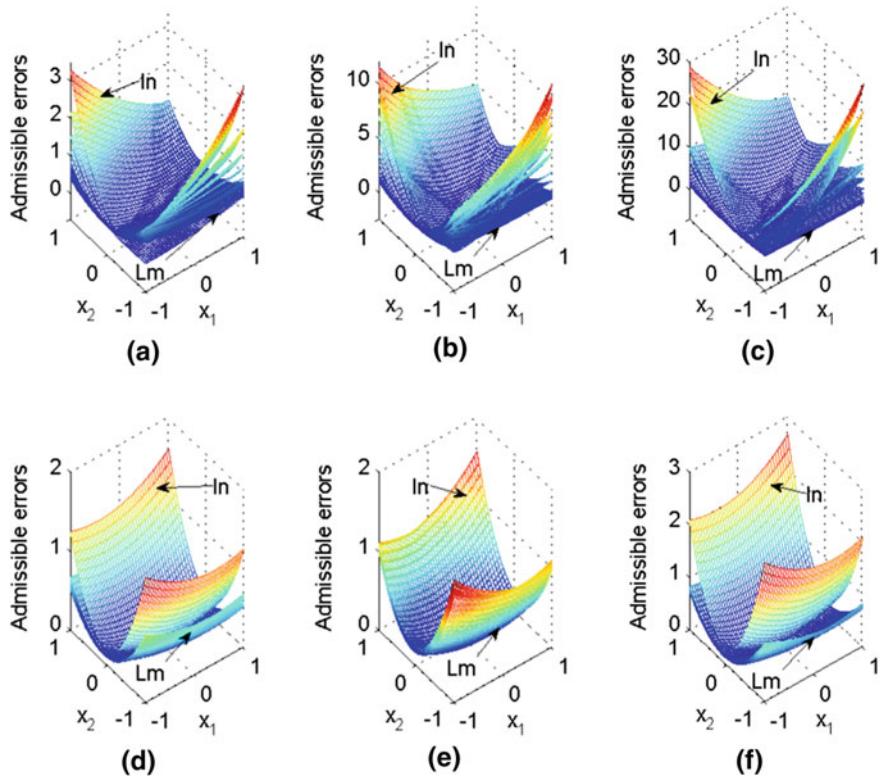


Fig. 3.14 The plots of the admissible errors corresponding to $\Psi^1(x_k)$ and $\Psi^2(x_k)$. **a** Admissible errors by γ_I and $\Psi^3(x_k)$. **b** Admissible errors by γ_{II} and $\Psi^3(x_k)$. **c** Admissible errors by γ_{III} and $\Psi^3(x_k)$. **d** Admissible errors by γ_I and $\Psi^4(x_k)$. **e** Admissible errors by γ_{II} and $\Psi^4(x_k)$. **f** Admissible errors by γ_{III} and $\Psi^4(x_k)$

$$P_2 = \begin{bmatrix} 2.35 & 3.31 \\ 3.31 & 9.28 \end{bmatrix}, \quad P_3 = \begin{bmatrix} 5.13 & -5.72 \\ -5.72 & 15.13 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 100.78 & 5.96 \\ 5.96 & 20.51 \end{bmatrix},$$

respectively. Let $q_i = 0.9999$ for $i = 0, 1, \dots$, and let $\varsigma = \varrho = 0.5$. We use the action network to obtain the admissible control law [23] with the expression given by $\mu(x_k) = W_{\text{initial}}\phi(V_{\text{initial}}x_k + b_{\text{initial}})$, where $\phi(\cdot)$ denotes the sigmoid activation function [23, 28]. The weight matrices are obtained as

$$W_{\text{initial}} = \begin{bmatrix} 4.15 & -0.14 & 2.64 & 0.93 & -3.74 & 0.39 & 1.13 \\ -3.67 & -0.67 & -0.25 & -2.59 & 0.27 & -1.58 & 3.49 \\ 1.40 & 0.24 & 1.13 & -2.69 & -4.77 \\ 4.91 & 0.60 & -3.55 & 2.97 & 0.22 \end{bmatrix}^T,$$

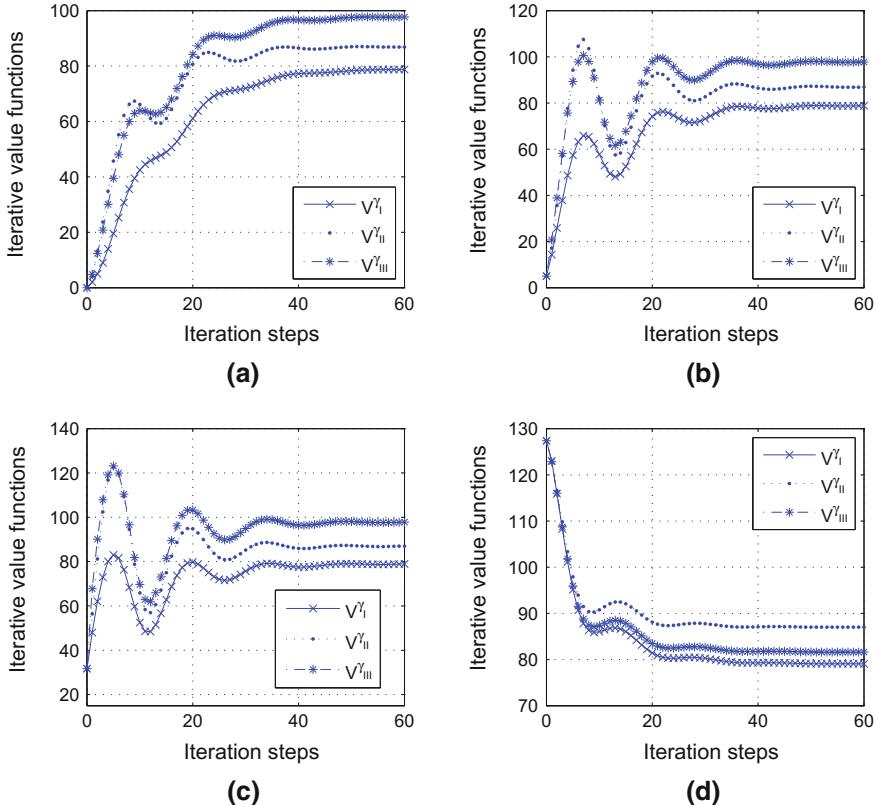


Fig. 3.15 The trajectories of the iterative value functions. **a** V^{γ_1} , V^{γ_2} and V^{γ_3} by $\Psi^1(x_k)$. **b** V^{γ_1} , V^{γ_2} and V^{γ_3} by $\Psi^2(x_k)$. **c** V^{γ_1} , V^{γ_2} and V^{γ_3} by $\Psi^3(x_k)$. **d** V^{γ_1} , V^{γ_2} and V^{γ_3} by $\Psi^4(x_k)$

$$V_{\text{initial}} = [0.01, 1.89, -0.12, 0.10, -0.02, 0.18, -0.01, -0.01, -1.75, 0.03, -0.00, 0.04],$$

and

$$b_{\text{initial}} = [-5.88, 0.81, -2.72, -2.07, -0.03, -0.23, -0.41, 1.59, 0.88, 1.97, -2.92, -4.28]^T.$$

Initialized by $\Psi^j(x_k)$, $j = 1, \dots, 4$, we choose 50000 data in $\Omega_x \times \Omega_u$ to compute γ_1 , γ_{II} , and γ_{III} . The corresponding trajectories are presented in Fig. 3.12a, b, c and d, respectively.

According to γ_1 , γ_{II} , and γ_{III} , the admissible errors with $\Psi^1(x_k)$ are shown in Fig. 3.13a, b and c, respectively. The admissible errors corresponding to $\Psi^2(x_k)$ are shown in Fig. 3.13d, e and f, respectively. According to γ_1 , γ_{II} , and γ_{III} , the admissible errors with $\Psi^3(x_k)$ are shown in Fig. 3.13g, h and i, respectively. The admissible errors with $\Psi^4(x_k)$ are shown in Fig. 3.13j, k and l, respectively.

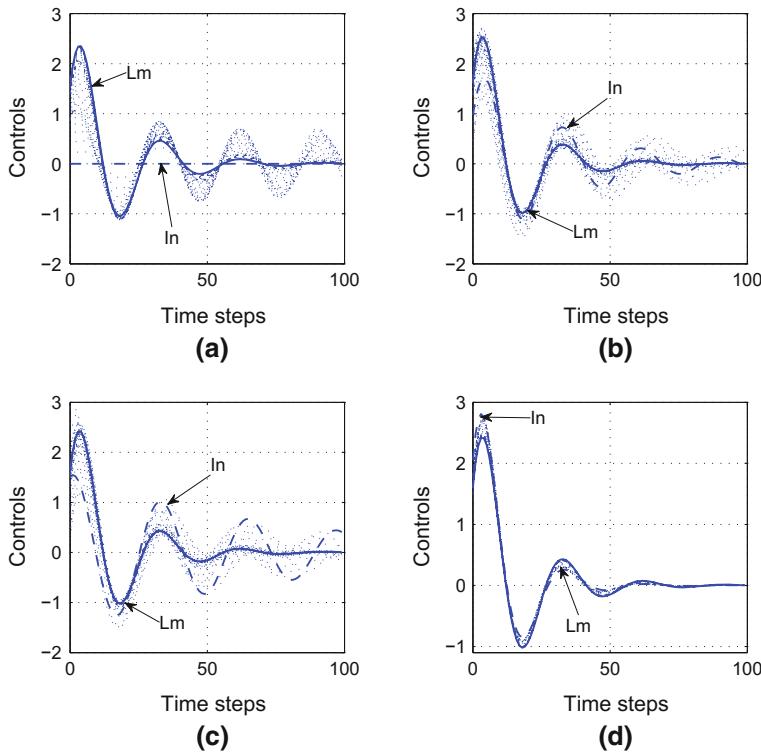


Fig. 3.16 Iterative control signals. **a** Control signals by $\Psi^1(x_k)$ and γ_I . **b** Control signals by $\Psi^2(x_k)$ and γ_{II} . **c** Control signals by $\Psi^3(x_k)$ and γ_{III} . **d** Control signals by $\Psi^4(x_k)$ and γ_{III}

sible errors corresponding to $\Psi^3(x_k)$ are shown in Fig. 3.14a, b and c, respectively. The admissible errors corresponding to $\Psi^4(x_k)$ are shown in Fig. 3.14d, e and f, respectively.

Initialized by $\Psi^1(x_k)$ – $\Psi^4(x_k)$, the trajectories of the iterative value functions for V^{γ_I} , $V^{\gamma_{II}}$ and $V^{\gamma_{III}}$ are shown in Fig. 3.15a, b, c and d, respectively.

The iterative control and state trajectories by $\Psi^1(x_k)$ and γ_I are shown in Figs. 3.16a and 3.17a, respectively. The iterative control and state trajectories by $\Psi^2(x_k)$ and γ_{II} are shown in Figs. 3.16b and 3.17b, respectively. The iterative control and state trajectories by $\Psi^3(x_k)$ and γ_{III} are shown in Figs. 3.16c and 3.17c, respectively. The iterative control and state trajectories by $\Psi^4(x_k)$ and γ_{III} are shown in Figs. 3.16d and 3.17d, respectively.

From Figs. 3.15–3.17, we can see that for different initial value functions under γ_I , γ_{II} and γ_{III} , the iterative value functions can converge to a finite neighborhood of the optimal cost function. The corresponding iterative controls and iterative states are also convergent. Therefore, the effectiveness of the present finite-approximation-error-based general value iteration algorithm has been shown for nonlinear systems.

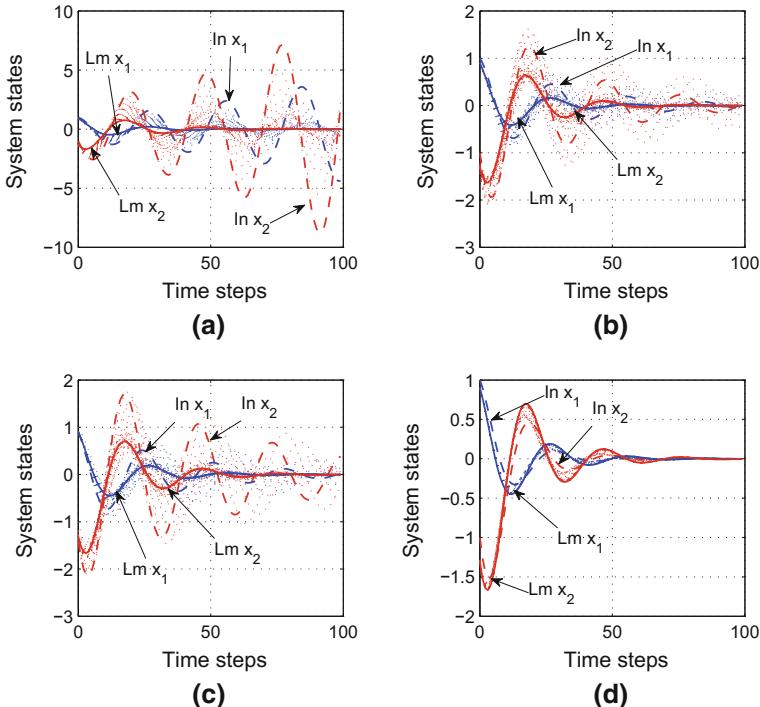


Fig. 3.17 Iterative state trajectories. **a** Trajectories by $\Psi^1(x_k)$ and γ_I . **b** Trajectories by $\Psi^2(x_k)$ and γ_{II} . **c** Trajectories by $\Psi^3(x_k)$ and γ_{III} . **d** Trajectories by $\Psi^4(x_k)$ and γ_{III}

On the other hand, if the initial weights of the neural networks, such as weights of the initial admissible control law, are changed, then the convergence property will also be different. Now, we also use the action network to obtain the admissible control law [23] with the expression given by

$$\mu'(x_k) = W'_{\text{initial}} \phi(V'_{\text{initial}} x_k + b'_{\text{initial}}),$$

and the weight matrices are obtained as

$$W'_{\text{initial}} = \begin{bmatrix} -1.92 & -2.35 & 4.59 & -0.44 & -2.10 & -3.97 \\ 4.45 & 4.27 & -1.55 & 4.89 & 4.49 & -2.81 \\ 1.41 & -4.61 & -3.37 & 2.99 & -3.48 & -2.39 \\ 4.64 & 1.53 & 3.32 & 3.81 & 3.30 & 4.24 \end{bmatrix}^T,$$

$$V'_{\text{initial}} = [-0.064, 0.002, 0.013, 0.012, -0.101, 0.015, 0.008, 0.019, -0.178, -0.002, -0.123, -0.091]$$

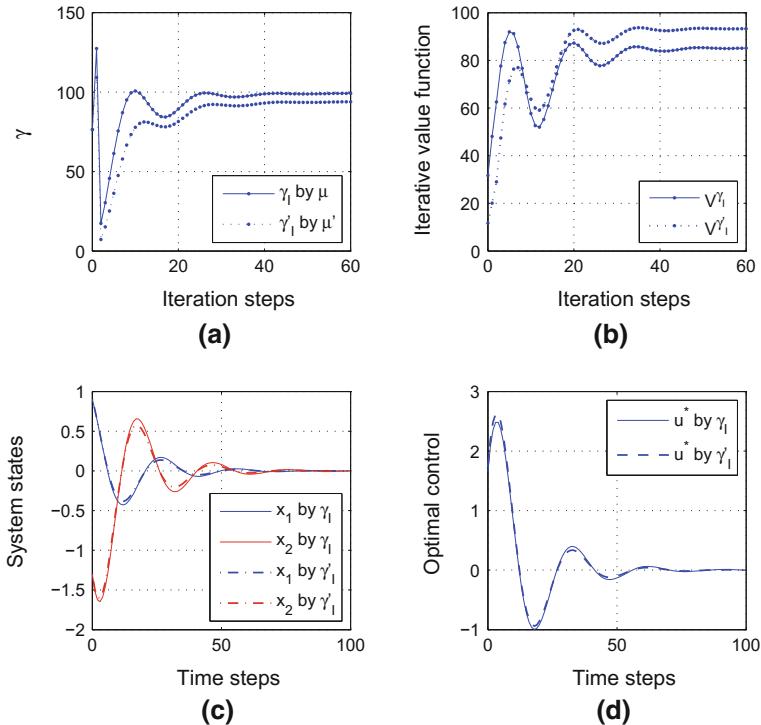


Fig. 3.18 Comparisons for different $\mu(x_k)$. **a** γ_1 and γ'_1 by $\mu(x_k)$ and $\mu'(x_k)$. **b** V^{γ_1} and $V^{\gamma'_1}$. **c** System states by γ_1 and γ'_1 . **d** Optimal control by γ_1 and γ'_1

and

$$\begin{aligned} b'_{\text{initial}} = & [-4.87, 3.90, -3.09, 2.21, 1.35, 0.44, \\ & -0.45, -1.33, -2.16, 3.08, -3.92, -4.86]^T. \end{aligned}$$

We choose $\Psi^3(x_k)$ as the initial value function for facilitating our analysis of simulation results. The trajectories of γ_1 by $\mu(x_k)$ and γ'_1 by $\mu'(x_k)$ are shown in Fig. 3.18a. We can see that if the initial weights of the neural networks for admissible control law are changed, then the parameter γ_1 is also changed. For different γ_1 , according to Theorem 3.4.4, the iterative value function can be convergent under different approximation errors. The trajectories of the iterative value function by γ_1 and γ'_1 are shown in Fig. 3.18b, respectively, where we can see that for different initial weights of the neural networks for the admissible control law, the iterative value function will converge to different values. Hence, the initial weights of the neural network will lead to different convergence properties. The optimal trajectories of system states and control by γ_1 and γ'_1 are shown in Fig. 3.18c and d, respectively.

3.5 Conclusions

In this chapter, several optimal control problems are solved by using value iteration ADP algorithms for infinite horizon discrete-time nonlinear systems with finite approximation errors. The iterative control laws which can guarantee the iterative value functions to reach optimums are obtained by using the iterative ADP algorithms. Then, a novel numerical adaptive learning control scheme based on the iterative ADP algorithms is presented to solve the numerical optimal control problems, which extends the implementation for both online and offline. Finally, a GVI ADP algorithm is developed with necessary analysis results, which overcomes the disadvantage of the traditional VI algorithm.

References

1. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791
2. Agricola I, Friedrich T (2008) Elementary geometry. American Mathematical Society, Providence
3. Almudevar A, Arruda EF (2012) Optimal approximation schedules for a class of iterative algorithms with an application to multigrid value iteration. *IEEE Trans Autom Control* 57(12):3132–3146
4. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2008) Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):943–949
5. Arruda EF, Ourique FO, LaCombe J, Almudevar A (2013) Accelerating the convergence of value iteration by using partial transition functions. *Eur J Oper Res* 229(1):190–198
6. Beard R (1995) Improving the closed-loop performance of nonlinear systems. Ph.D. thesis, Rensselaer Polytechnic Institute
7. Bertsekas DP (2007) Dynamic programming and optimal control, 3rd edn. Athena Scientific, Belmont
8. Cheng T, Lewis FL, Abu-Khalaf M (2007) A neural network solution for fixed-final time optimal control of nonlinear systems. *Automatica* 43(3):482–490
9. Dorf RC, Bishop RH (2011) Modern control systems, 12th edn. Prentice-Hall, Upper Saddle River
10. Feinberg EA, Huang J (2014) The value iteration algorithm is not strongly polynomial for discounted dynamic programming. *Oper Res Lett* 42(2):130–131
11. Grune L, Rantzer A (2008) On the infinite horizon performance of receding horizon controllers. *IEEE Trans Autom Control* 53(9):2100–2111
12. Hagan MT, Demuth HB, Beale MH (1996) Neural network design. PWS Publishing Company, Boston
13. Haykin S (2009) Neural networks and learning machines, 3rd edn. Prentice-Hall, Upper Saddle River
14. Jin N, Liu D, Huang T, Pang Z (2007) Discrete-time adaptive dynamic programming using wavelet basis function neural networks. In: Proceedings of the IEEE symposium on approximate dynamic programming and reinforcement learning, pp 135–142
15. Kushner HJ (2010) Numerical algorithms for optimal controls for nonlinear stochastic systems with delays. *IEEE Trans Autom Control* 55(9):2170–2176
16. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst Mag* 9(3):32–50

17. Lewis FL, Vrabie D, Vamvoudakis KG (2012) Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst Mag* 32(6):76–105
18. Lincoln B, Rantzer A (2006) Relaxing dynamic programming. *IEEE Trans Autom Control* 51(8):1249–1260
19. Liu D, Zhang Y, Zhang H (2005) A self-learning call admission control scheme for CDMA cellular networks. *IEEE Trans Neural Netw* 16(5):1219–1228
20. Liu D, Javaherian H, Kovalenko O, Huang T (2008) Adaptive critic learning techniques for engine torque and air-fuel ratio control. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):988–993
21. Liu D, Wang D, Zhao D, Wei Q, Jin N (2012) Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming. *IEEE Trans Autom Sci Eng* 9(3):628–634
22. Liu D, Wei Q (2013) Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems. *IEEE Trans Cybern* 43(2):779–789
23. Liu D, Wei Q (2014) Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Trans Neural Netw Learn Syst* 25(3):621–634
24. Murray JJ, Cox CJ, Lendaris GG, Saeks R (2002) Adaptive dynamic programming. *IEEE Trans Syst Man Cybern Part C Appl Rev* 32(2):140–153
25. Navarro-Lopez EM (2007) Local feedback passivation of nonlinear discrete-time systems through the speed-gradient algorithm. *Automatica* 43(7):1302–1306
26. Rantzer A (2006) Relaxed dynamic programming in switching systems. *IEE Proc Control Theory Appl* 153(5):567–574
27. Seiffertt J, Sanyal S, Wunsch DC (2008) Hamilton-Jacobi-Bellman equations and approximate dynamic programming on time scales. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):918–923
28. Si J, Wang YT (2001) On-line learning control by association and reinforcement. *IEEE Trans Neural Netw* 12(2):264–276
29. Sira-Ramirez H (1991) Non-linear discrete variable structure systems in quasi-sliding mode. *Int J Control* 54(5):1171–1187
30. Vamvoudakis KG, Lewis FL (2011) Multi-player non-zero-sum games: online adaptive learning solution of coupled Hamilton-Jacobi equations. *Automatica* 47(8):1556–1569
31. Vrabie D, Vamvoudakis KG, Lewis FL (2013) Optimal adaptive control and differential games by reinforcement learning principles. *IET, London*
32. Wang FY, Zhang H, Liu D (2009) Adaptive dynamic programming: an introduction. *IEEE Comput Intell Mag* 4(2):39–47
33. Wang FY, Jin N, Liu D, Wei Q (2011) Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε -error bound. *IEEE Trans Neural Netw* 22(1):24–36
34. Wei Q, Liu D (2012) An iterative ε -optimal control scheme for a class of discrete-time nonlinear systems with unfixed initial state. *Neural Netw* 32:236–244
35. Wei Q, Liu D (2012) A new optimal control method for discrete-time nonlinear systems with approximation errors. In: *Proceedings of the world congress on intelligent control and automation*, pp 185–190
36. Wei Q, Liu D (2012) Adaptive dynamic programming with stable value iteration algorithm for discrete-time nonlinear systems. In: *Proceedings of the IEEE international joint conference on neural networks*, pp 1–6
37. Wei Q, Liu D (2013) Numerical adaptive learning control scheme for discrete-time non-linear systems. *IET Control Theory Appl* 7(18):1472–1486
38. Wei Q, Liu D (2014) A novel iterative θ -Adaptive dynamic programming for discrete-time nonlinear systems. *IEEE Trans Autom Sci Eng* 11(4):1176–1190
39. Wei Q, Liu D (2014) Stable iterative adaptive dynamic programming algorithm with approximation errors for discrete-time nonlinear systems. *Neural Comput Appl* 24(6):1355–1367

40. Wei Q, Liu D (2015) A novel policy iteration based deterministic Q-learning for discrete-time nonlinear systems. *Sci China Inf Sci* 58(12):1–15
41. Wei Q, Liu D (2015) Neural-network-based adaptive optimal tracking control scheme for discrete-time nonlinear systems with approximation errors. *Neurocomputing* 149:106–115
42. Wei Q, Liu D, Lewis FL (2015) Optimal distributed synchronization control for continuous-time heterogeneous multi-agent differential graphical games. *Inf Sci* 317:96–113
43. Wei Q, Wang FY, Liu D, Yang X (2014) Finite-approximation-error-based discrete-time iterative adaptive dynamic programming. *IEEE Trans Cybern* 44(12):2820–2833
44. Wei Q, Zhang H, 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
45. Werbos PJ (1977) Advanced forecasting methods for global crisis warning and models of intelligence. *Gen Syst Yearb* 22:25–38
46. Werbos PJ (1991) A menu of designs for reinforcement learning over time. In: Miller WT, Sutton RS, Werbos PJ (eds) *Neural networks for control*. MIT Press, Cambridge, pp 67–95
47. Werbos PJ (1992) Approximate dynamic programming for real-time control and neural modeling. In: White DA, Sofge DA (eds) *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches* (Chapter 13). Van Nostrand Reinhold, New York
48. Yang Q, Vance JB, Jagannathan S (2008) Control of nonaffine nonlinear discrete-time systems using reinforcement-learning-based linearly parameterized neural networks. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):994–1001
49. Younkin G, Hesla E (2008) Origin of numerical control. *IEEE Ind Appl Mag* 14(5):10–12
50. Zhang C, Ordonez R (2007) Numerical optimization-based extremum seeking control with application to ABS design. *IEEE Trans Autom Control* 52(3):454–467
51. Zhang H, Wei Q, Luo Y (2008) A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):937–942
52. Zhang H, Luo Y, Liu D (2009) The RBF neural network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraint. *IEEE Trans Neural Netw* 20(9):1490–1503
53. Zhang H, Wei Q, Liu D (2011) An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games. *Automatica* 47(1):207–214

Chapter 4

Policy Iteration for Optimal Control of Discrete-Time Nonlinear Systems

4.1 Introduction

Value iteration ADP algorithm and policy iteration ADP algorithm are the effective methods to obtain solutions of the Bellman equation indirectly [6, 8–11, 13, 19–21, 24]. Many discussions on value iteration have been presented in the last two chapters. On the other hand, policy iteration is the main focus of this chapter.

Policy iteration algorithm for continuous-time systems was proposed in [14]. In [14], Murray et al. proved that for continuous-time affine nonlinear systems, the iterative value function will converge to the optimum non-increasingly and each of the iterative controls stabilizes the nonlinear systems using policy iteration algorithms. This is a merit of policy iteration algorithm and hence achieved many applications for solving optimal control problems of nonlinear systems. In 2005, Abu-Khalaf and Lewis [1] proposed a policy iteration algorithm for continuous-time nonlinear systems with control constraints. Zhang et al. [25] applied policy iteration algorithm to solve continuous-time nonlinear two-person zero-sum games. Vamvoudakis et al. [17] proposed a multiagent differential graphical games for continuous-time linear systems using policy iteration algorithms. Bhasin et al. [3] proposed an online actor-critic-identifier architecture to obtain optimal control law for uncertain continuous-time nonlinear systems by policy iteration algorithms. Till now, nearly all the online iterative ADP algorithms are policy iteration algorithms. However, most of the discussions on the policy iteration algorithms are based on continuous-time systems [3, 17, 23, 25]. The discussions on policy iteration algorithms for discrete-time control systems are scarce. Only in [8, 9, 16], a brief sketch of policy iteration algorithm for discrete-time systems was mentioned while the stability and convergence properties were not discussed. To the best of our knowledge, there is not much discussion focused on policy iteration algorithms for discrete-time systems. Therefore, in this chapter, policy iteration method for optimal control is developed for discrete-time nonlinear systems [12].

4.2 Policy Iteration Algorithm

Consider the following deterministic discrete-time systems

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \dots, \quad (4.2.1)$$

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

Let

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

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

$$J(x_0, \underline{u}_0) = \sum_{k=0}^{\infty} U(x_k, u_k), \quad (4.2.2)$$

where $U(x_k, u_k)$ is the utility function.

We will study optimal control problems for (4.2.1). The goal is to find an optimal control scheme which stabilizes system (4.2.1) and simultaneously minimizes the cost function (4.2.2). For convenience of analysis, results of this chapter are based on the following assumption.

Assumption 4.2.1 The origin $x_k = 0$ is an equilibrium state of system (4.2.1) 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 $x_k = 0$; the system function $F(x_k, u_k)$ is Lipschitz continuous on a compact set $\Omega \subset \mathbb{R}^n$ containing the origin; the system (4.2.1) is controllable on Ω ; the utility function $U(x_k, u_k)$ is a positive-definite function of x_k and u_k .

As system (4.2.1) is assumed to be controllable, there exists a stable control sequence $\underline{u}_k = (u_k, u_{k+1}, \dots)$ that moves the state x_k to zero. For optimal control problems, the designed feedback control must not only stabilize the system (4.2.1) on Ω , but also guarantee that the cost function (4.2.2) is finite, i.e., the control must be admissible (cf. Definition 2.2.1 or [2]). Let $\mathcal{A}(\Omega)$ denote the set which contains all the admissible control sequences associated with the controllable set Ω of states. Then, the optimal cost function can be defined as

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

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

$$J^*(x_k) = \min_{u_k} \{U(x_k, u_k) + J^*(F(x_k, u_k))\}. \quad (4.2.3)$$

Define the law of optimal control as $u^*(\cdot)$. Each $u_k^* = u^*(x_k)$ is obtained by

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

Hence, the Bellman equation (4.2.3) can be written as

$$\begin{aligned} J^*(x_k) &= U(x_k, u_k^*) + J^*(F(x_k, u_k^*)) \\ &= U(x_k, u^*(x_k)) + J^*(F(x_k, u^*(x_k))). \end{aligned} \quad (4.2.4)$$

We can see that if we want to obtain the optimal control sequence u_k , we must obtain the optimal cost function $J^*(x_k)$. Generally speaking, $J^*(x_k)$ is unknown before the whole control sequence u_k is considered. If we adopt the traditional dynamic programming method to obtain the optimal cost function at every time step, then we have to face the “curse of dimensionality”. Furthermore, the optimal control is discussed in infinite horizon. This means the length of the control sequence is infinite, which implies that the optimal control sequence is nearly impossible to obtain analytically by the Bellman equation (4.2.4). To overcome this difficulty, a novel discrete-time policy iteration ADP algorithm will be developed to obtain the optimal control sequence for nonlinear systems. The goal of the present policy iteration ADP algorithm is to construct iterative control laws $v_i(x_k)$, which move an arbitrary initial state x_0 to the equilibrium $x_k = 0$, and simultaneously guarantee that the iterative value functions reach the optimum.

4.2.1 Derivation of Policy Iteration Algorithm

In the present policy iteration algorithm, the value functions and control laws are updated by iterations, with the iteration index i increasing from 0 to ∞ . Starting from an arbitrary admissible control law $v_0(x_k)$, the PI algorithm can be described as follows.

For $i = 1, 2, \dots$, the iterative value function $V_i(x_k)$ is constructed from $v_{i-1}(x_k)$ by solving the following generalized Bellman equation

$$V_i(x_k) = U(x_k, v_{i-1}(x_k)) + V_i(F(x_k, v_{i-1}(x_k))), \quad (4.2.5)$$

and the iterative control law $v_i(x_k)$ is updated by

$$\begin{aligned} v_i(x_k) &= \arg \min_{u_k} \{U(x_k, u_k) + V_i(x_{k+1})\} \\ &= \arg \min_{u_k} \{U(x_k, u_k) + V_i(F(x_k, u_k))\}. \end{aligned} \quad (4.2.6)$$

In the above PI algorithm, (4.2.5) is called policy evaluation (or value update) and, (4.2.6) is called policy improvement (or policy update) [16, 18]. The iterative

Table 4.1 The iterative process of the PI algorithm in (4.2.5) and (4.2.6)

$v_0 \rightarrow V_1$ (4.2.5) evaluation	$v_1 \rightarrow V_2$ (4.2.5) evaluation	$v_2 \rightarrow V_3$ (4.2.5) evaluation	...
$V_1 \rightarrow v_1$ (4.2.6) minimization	$V_2 \rightarrow v_2$ (4.2.6) minimization	$V_3 \rightarrow v_3$ (4.2.6) minimization	...
$i = 1$	$i = 2$	$i = 3$...

process of the PI algorithm in (4.2.5) and (4.2.6) is illustrated in Table 4.1, where the iteration flows as in Fig. 2.1. The algorithm starts with an admissible control $v_0(x_k)$, and it obtains $v_1(x_k)$ at the end of the first iteration. Comparing between Tables 2.2 and 4.1, we note that “calculation” performed in Table 2.2 using (2.2.11) is a simpler operation than “evaluation” in Table 4.1 based on (4.2.5). In (4.2.5), one needs to solve for $V_i(x_k)$ from a nonlinear functional equation, whereas in (2.2.11), it is a simple calculation.

From the policy iteration algorithm (4.2.5) and (4.2.6), we can see that the iterative value function $V_i(x_k)$ is used to approximate $J^*(x_k)$, and the iterative control law $v_i(x_k)$ is used to approximate $u^*(x_k)$. As (4.2.5) is generally not a Bellman equation, the iterative value functions the iterative control laws

$$V_i(x_k) \neq J^*(x_k), \quad v_i(x_k) \neq u^*(x_k), \quad \forall i = 1, 2, \dots$$

Therefore, it is necessary to determine whether the PI algorithm is convergent or not, which means that whether $V_i(x_k)$ and $v_i(x_k)$ converge to the optimal ones as $i \rightarrow \infty$. In the next section, we will show such properties of the above policy iteration algorithm.

4.2.2 Properties of Policy Iteration Algorithm

For the policy iteration algorithm of continuous-time nonlinear systems, it is shown that any of the iterative control laws can stabilize the system [14]. This is a merit of the policy iteration algorithm. For discrete-time systems, stability of the system can also be guaranteed under the iterative control law.

Lemma 4.2.1 *For $i = 0, 1, \dots$, let $V_i(x_k)$ and $v_i(x_k)$ be obtained by the policy iteration algorithm (4.2.5) and (4.2.6), where $v_0(x_k)$ is an arbitrary admissible control law. If Assumption 4.2.1 holds, then the iterative control law $v_i(x_k)$, $\forall i = 0, 1, \dots$, stabilizes the nonlinear system (4.2.1).*

The lemma can easily be proved by considering (4.2.5), which shows that $V_i(x_k)$ is a Lyapunov function.

For continuous-time policy iteration algorithms [14], according to the derivative of the difference of iterative value functions with respect to time t , it is proven that the iterative value functions are non-increasingly convergent to the optimal solution of the Bellman equation. For discrete-time systems, the analysis method for the continuous-time is no longer applicable. Thus, a new convergence proof is established in this chapter. We will show that using the policy iteration algorithm of discrete-time nonlinear systems, the iterative value functions are also nonincreasingly convergent to the optimum.

Theorem 4.2.1 For $i = 0, 1, \dots$, let $V_i(x_k)$ and $v_i(x_k)$ be obtained by (4.2.5) and (4.2.6). If Assumption 4.2.1 holds, then the iterative value function $V_i(x_k)$, $\forall x_k \in \mathbb{R}^n$, is a monotonically nonincreasing sequence, i.e.,

$$V_{i+1}(x_k) \leq V_i(x_k), \quad \forall i \geq 0. \quad (4.2.7)$$

Proof For $i = 0, 1, \dots$, define a new iterative value function $\Gamma_{i+1}(x_k)$ as

$$\Gamma_{i+1}(x_k) = U(x_k, v_i(x_k)) + V_i(x_{k+1}), \quad (4.2.8)$$

where $v_i(x_k)$ is obtained by (4.2.6). According to (4.2.5), (4.2.6), and (4.2.8), we can obtain

$$\Gamma_{i+1}(x_k) \leq V_i(x_k), \quad \forall x_k. \quad (4.2.9)$$

Then, we derive

$$\begin{aligned} \Gamma_{i+1}(x_k) &= U(x_k, v_i(x_k)) + V_i(x_{k+1}) \\ &\geq U(x_k, v_i(x_k)) + \Gamma_{i+1}(x_{k+1}) \\ &\geq \sum_{j=0}^{N-1} U(x_{k+j}, v_i(x_{k+j})) + \Gamma_{i+1}(x_{k+N}), \end{aligned}$$

where $N > 0$ is an integer. Let $N \rightarrow \infty$. We have

$$V_i(x_k) \geq \Gamma_{i+1}(x_k) \geq \sum_{j=0}^{\infty} U(x_{k+j}, v_i(x_{k+j})) + \lim_{N \rightarrow \infty} \Gamma_{i+1}(x_{k+N}). \quad (4.2.10)$$

From (4.2.10), we get

$$V_{i+1}(x_k) = \sum_{j=0}^{\infty} U(x_{k+j}, v_i(x_{k+j})) \leq \Gamma_{i+1}(x_k). \quad (4.2.11)$$

Therefore, according to (4.2.9), we have

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

The proof is complete.

Since $V_i(x_k)$ is finite, we have $U(x_{k+N}, v_i(x_{k+N})) \rightarrow 0$ as $N \rightarrow \infty$ which implies $x_{k+N} \rightarrow 0$ as $N \rightarrow \infty$. Thus, $v_i(x_k)$ is a stable control law. On the other hand, we get

$$\sum_{j=0}^{\infty} U(x_{k+j}, v_i(x_{k+j})) < \infty.$$

Hence, $v_i(x_k)$ is an admissible control law.

Corollary 4.2.1 For $i = 0, 1, \dots$, let $V_i(x_k)$ and $v_i(x_k)$ be obtained by (4.2.5) and (4.2.6), where $v_0(x_k)$ is an arbitrary admissible control law. If Assumption 4.2.1 holds, then the iterative control law $v_i(x_k)$ is admissible, $\forall i = 0, 1, \dots$

From Theorem 4.2.1, the iterative value function $V_i(x_k) \geq 0$ is monotonically non-increasing and bounded below for iteration index $i = 1, 2, \dots$. Hence, we can derive the following theorem.

Theorem 4.2.2 For $i = 0, 1, \dots$, let $V_i(x_k)$ and $v_i(x_k)$ be obtained by (4.2.5) and (4.2.6). If Assumption 4.2.1 holds, then the iterative value function $V_i(x_k)$ converges to the optimal cost function $J^*(x_k)$, as $i \rightarrow \infty$, i.e.,

$$\lim_{i \rightarrow \infty} V_i(x_k) = J^*(x_k), \quad \forall x_k, \quad (4.2.12)$$

which satisfies the Bellman equation (4.2.3).

Proof The statement can be proven by the following three steps.

(1) Show that the limit of the iterative value function $V_i(x_k)$ satisfies the Bellman equation, as $i \rightarrow \infty$.

According to Theorem 4.2.1, as $V_i(x_k)$ is non-increasing and lower bounded, the limit of the iterative value function $V_i(x_k)$ exists as $i \rightarrow \infty$. Define the value function $V_\infty(x_k)$ as the limit of the iterative function $V_i(x_k)$, i.e.,

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

According to the definition of $\Gamma_{i+1}(x_k)$ in (4.2.8), we have

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

According to (4.2.11), we can get

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

We can obtain

$$V_\infty(x_k) \leq V_{i+1}(x_k) \leq \Gamma_{i+1}(x_k) = \min_{u_k} \{U(x_k, u_k) + V_i(x_{k+1})\}.$$

Thus, letting $i \rightarrow \infty$, we obtain

$$V_\infty(x_k) \leq \min_{u_k} \{U(x_k, u_k) + V_\infty(x_{k+1})\}. \quad (4.2.13)$$

Let $\varepsilon > 0$ be an arbitrary positive number. Since $V_i(x_k)$ is nonincreasing, there exists a positive integer p such that

$$V_p(x_k) - \varepsilon \leq V_\infty(x_k) \leq V_p(x_k). \quad (4.2.14)$$

Hence, we can get

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

Since ε is arbitrary, we have

$$V_\infty(x_k) \geq \min_{u_k} \{U(x_k, u_k) + V_\infty(x_{k+1})\}. \quad (4.2.15)$$

Combining (4.2.13) and (4.2.15), we can obtain

$$V_\infty(x_k) = \min_{u_k} \{U(x_k, u_k) + V_\infty(x_{k+1})\}, \quad (4.2.16)$$

i.e., $V_\infty(x_k)$ satisfies the Bellman equation (4.2.3).

Next, let $\mu(x_k)$ be an arbitrary admissible control law, and define a new value function $P(x_k)$, which satisfies

$$P(x_k) = U(x_k, \mu(x_k)) + P(x_{k+1}). \quad (4.2.17)$$

Then, we can proceed to the second step of the proof.

(2) *Show that for an arbitrary admissible control law $\mu(x_k)$, the value function $V_\infty(x_k) \leq P(x_k)$.*

For $i = 1, 2, \dots$, define a new iterative value function $\Pi_i(x_k)$ as

$$\Pi_i(x_k) = U(x_k, \mu(x_k)) + \Pi_{i-1}(x_{k+1}),$$

where $\Pi_0(x_k) = V_\infty(x_k)$. Then, for $i = 1$, we get

$$\begin{aligned}
\Pi_1(x_k) &= U(x_k, \mu(x_k)) + \Pi_0(x_{k+1}) \\
&= U(x_k, \mu(x_k)) + V_\infty(x_{k+1}) \\
&\geq \min_{u_k} \{U(x_k, u_k) + V_\infty(x_{k+1})\} \\
&= V_\infty(x_k).
\end{aligned}$$

Using mathematical induction, we can obtain

$$V_\infty(x_k) \leq \Pi_i(x_k), \quad \forall i. \quad (4.2.18)$$

We can also obtain

$$\Pi_\infty(x_k) = U(x_k, \mu(x_k)) + \Pi_\infty(x_{k+1})$$

and $P(x_k) = \Pi_\infty(x_k)$, which together with (4.2.18) implies

$$V_\infty(x_k) \leq P(x_k). \quad (4.2.19)$$

(3) Show that the value function $V_\infty(x_k)$ equals to the optimal cost function $J^*(x_k)$.

According to the definition of $J^*(x_k)$ in (4.2.3), for $i = 0, 1, \dots$, we have

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

Let $i \rightarrow \infty$. We can obtain

$$V_\infty(x_k) \geq J^*(x_k). \quad (4.2.20)$$

On the other hand, for an arbitrary admissible control law $\mu(x_k)$, (4.2.19) holds. Let $\mu(x_k) = u^*(x_k)$, where $u^*(x_k)$ is the optimal control law. Thus, when $\mu(x_k) = u^*(x_k)$, we can get

$$V_\infty(x_k) \leq P(x_k) = J^*(x_k). \quad (4.2.21)$$

According to (4.2.20) and (4.2.21), we can obtain (4.2.12). The proof is complete.

Corollary 4.2.2 *Let $x_k \in \mathbb{R}^n$ be an arbitrary state vector. If Theorem 4.2.2 holds, then the iterative control law $v_i(x_k)$ converges to the optimal control law as $i \rightarrow \infty$, i.e.,*

$$u^*(x_k) = \lim_{i \rightarrow \infty} v_i(x_k).$$

Remark 4.2.1 In Theorems 4.2.1 and 4.2.2, we have proven that the iterative value functions are monotonically non-increasing and convergent to the optimal cost function as $i \rightarrow \infty$. The stability of the nonlinear systems can also be guaranteed under the iterative control laws. Hence, the convergence and stability properties of the policy iteration algorithms for continuous-time nonlinear systems are also valid for the pol-

icy iteration algorithms for discrete-time nonlinear systems. However, we emphasize that the analysis techniques between the continuous-time and discrete-time policy iteration algorithms are quite different. First, the Hamilton–Jacobi–Bellman equation of continuous-time systems and the Bellman equation of discrete-time systems are different inherently. Second, the analysis method for continuous-time policy iteration algorithm is based on derivative techniques [1, 3, 14, 17, 23, 25]. The analysis method for continuous-time policy iteration algorithm is no longer applicable to discrete-time situation. In this chapter, we will establish properties for the discrete-time policy iteration algorithms, which form the theoretical basis for applications to optimal control problems of discrete-time nonlinear systems.

Remark 4.2.2 In [2], value iteration algorithm of ADP is proposed to obtain the optimal solution of the Bellman equation, for the following discrete-time affine nonlinear systems

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

with the cost function

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

where Q and R are defined as the penalizing matrices for system state and control vectors, respectively. Let $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ be both positive-definite matrices. Starting from $V_0(x_k) \equiv 0$, the value iteration algorithm can be expressed as

$$\begin{cases} u_i(x_k) = -\frac{1}{2} R^{-1} g^T(x_k) \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}}, \\ V_{i+1}(x_k) = x_k^T Q x_k + u_i^T(x_k) R u_i(x_k) + V_i(x_{k+1}). \end{cases} \quad (4.2.22)$$

It was proven in [2] that $V_i(x_k)$ is a monotonically nondecreasing sequence and bounded, and hence converges to $J^*(x_k)$. However, using the value iteration equation (4.2.22), the stability of the system under the iterative control law cannot be guaranteed. We should point out that using the policy iteration algorithm (4.2.5) and (4.2.6), the stability property can be guaranteed. In Sect. 4.3, numerical results and comparisons between the policy and value iteration algorithms will be presented to show these properties.

Therefore, if an admissible control of the nonlinear system is known, then policy iteration algorithm is preferred to obtain the optimal control law with good convergence and stability properties. Accordingly, the initial admissible control law is a key to the success of the policy iteration algorithm. In the next section, an effective method will be discussed to obtain the initial admissible control law using neural networks.

4.2.3 Initial Admissible Control Law

As the policy iteration algorithm requires to start with an admissible control law, the method of obtaining the admissible control law is important to the implementation of the algorithm. Actually, for all the policy iteration algorithms of ADP, including [1, 3, 14, 17, 23, 25], the initial admissible control law is necessary to implement their algorithms. Unfortunately, to the best of our knowledge, there is no theory on how to design the initial admissible control law. In this section, we will give an effective method to obtain the initial admissible control law by experiments using neural networks.

First, let $\Psi(x_k) \geq 0$ be an arbitrary semipositive definite function. For $i = 0, 1, \dots$, we define a new value function as

$$\Phi_{i+1}(x_k) = U(x_k, \mu(x_k)) + \Phi_i(x_{k+1}), \quad (4.2.23)$$

where $\Phi_0(x_k) = \Psi(x_k)$. Then, we can derive the following theorem.

Theorem 4.2.3 Suppose Assumption 4.2.1 holds. Let $\Psi(x_k) \geq 0$ be an arbitrary semipositive-definite function. Let $\mu(x_k)$ be an arbitrary control law for system (4.2.1) which satisfies $\mu(0) = 0$. Define the iterative value function $\Phi_i(x_k)$ as in (4.2.23), where $\Phi_0(x_k) = \Psi(x_k)$. Then, $\mu(x_k)$ is an admissible control law if and only if the limit of $\Phi_i(x_k)$ exists as $i \rightarrow \infty$.

Proof We first prove the sufficiency of the statement. Assume that $\mu(x_k)$ is an admissible control law. According to (4.2.23),

$$\begin{aligned} \Phi_{i+1}(x_k) &= U(x_k, \mu(x_k)) + \Phi_i(x_{k+1}) \\ &= U(x_k, \mu(x_k)) + U(x_{k+1}, \mu(x_{k+1})) + \Phi_{i-1}(x_{k+2}) \\ &= \dots \\ &= \sum_{j=0}^i U(x_{k+j}, \mu(x_{k+j})) + \Psi(x_{k+i+1}), \end{aligned} \quad (4.2.24)$$

where $\Psi(x_k) = \Phi_0(x_k)$. Letting $i \rightarrow \infty$, it follows

$$\lim_{i \rightarrow \infty} \Phi_i(x_k) = \sum_{j=0}^{\infty} U(x_{k+j}, \mu(x_{k+j})) + \lim_{i \rightarrow \infty} \Psi(x_{k+i+1}). \quad (4.2.25)$$

Since $\mu(x_k)$ is an admissible control law,

$$\sum_{j=0}^{\infty} U(x_{k+j}, \mu(x_{k+j})) \quad (4.2.26)$$

is finite and $\Psi(x_{k+i+1}) \rightarrow 0$ as $i \rightarrow \infty$ ($x_{k+i+1} \rightarrow 0$). Hence, $\lim_{i \rightarrow \infty} \Phi_i(x_k)$ is finite.

On the other hand, if the limit of $\Phi_i(x_k)$ exists as $i \rightarrow \infty$, according to (4.2.25), as $\Psi(x_k)$ is finite, we can get that (4.2.26) is finite. Since the utility function $U(x_k, u_k)$ is positive definite, then we can obtain

$$U(x_{k+j}, \mu(x_{k+j})) \rightarrow 0 \text{ as } j \rightarrow \infty.$$

As $\mu(x_k) = 0$ for $x_k = 0$, we can get that $x_k \rightarrow 0$ as $k \rightarrow \infty$, which means that system (4.2.1) is stable and $\mu(x_k)$ is an admissible control law. The necessity of the statement is proven, and the proof of the theorem is complete.

According to Theorem 4.2.3, we can establish an effective iteration algorithm by repeating experiments using neural networks. The detailed implementation of the iteration algorithm is expressed in Algorithm 4.2.1.

Algorithm 4.2.1 Computation of the initial admissible control law

Initialization:

Choose a semi-positive definite function $\Psi(x_k) \geq 0$.

Initialize two neural networks (critic networks) *cnet1* and *cnet2* with small random weights.

Let $\Phi_0(x_k) = \Psi(x_k)$.

Give the maximum iteration number of computation i_{\max} .

Iteration:

Step 1. Establish an action neural network with small random weights to generate an initial control law $\mu(x_k)$ with $\mu(x_k) = 0$ for $x_k = 0$. Obtain $\{x_k\}$ and $\{x_{k+1}\}$.

Step 2. Let $i = 0$. Train the critic network *cnet1* so that its output becomes

$$\{U(x_k, \mu(x_k)) + \Phi_0(x_{k+1})\}.$$

Step 3. Copy *cnet1* to *cnet2*, i.e., let *cnet2* = *cnet1*.

Step 4. Let $i = i + 1$. Use *cnet2* to get $\Phi_i(x_{k+1})$ and train the critic network *cnet1* so that its output becomes

$$\{U(x_k, \mu(x_k)) + \Phi_i(x_{k+1})\}.$$

Step 5. Use the trained *cnet1* to get $\Phi_{i+1}(x_k)$ and use *cnet2* to get $\Phi_i(x_k)$. If $|\Phi_{i+1}(x_k) - \Phi_i(x_k)| < \varepsilon$, then goto Step 7; else, goto next step.

Step 6. If $i > i_{\max}$, then goto Step 1; else, goto Step 3.

Step 7. Let $v_0(x_k) = \mu(x_k)$.

Remark 4.2.3 We can see that the above training procedure can easily be implemented by computer program. After creating an action network, the weights of the critic networks *cnet1* and *cnet2* can be updated iteratively. If the weights of the critic network are convergent, then $\mu(x_k)$ must be an admissible control law which is generated by a neural network with random weights. As the weights of action network are

chosen randomly, the admissibility of the control law is unknown before the weights of the critic networks are convergent. Hence, the iteration process of Algorithm 4.2.1 is implemented off-line.

4.2.4 Summary of Policy Iteration ADP Algorithm

According to the above preparations, we can summarize the discrete-time policy iteration algorithm in Algorithm 4.2.2.

Algorithm 4.2.2 Discrete-time policy iteration algorithm

Initialization:

- Choose randomly an array of initial states x_0 .
- Choose a computation precision ε .
- Give the initial admissible control law $v_0(x_k)$.
- Give the maximum iteration number of computation i_{\max} .
- Let the iteration index $i = 0$.

Iteration:

- Step 1. Let $i = i + 1$. Construct the iterative value functions $V_i(x_k)$, which satisfies the following generalized Bellman equation

$$V_i(x_k) = U(x_k, v_{i-1}(x_k)) + V_i(F(x_k, v_{i-1}(x_k))).$$

- Step 2. Update the iterative control law $v_{i+1}(x_k)$ by

$$v_i(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + V_i(x_{k+1})\}.$$

- Step 3. If $|V_i(x_k) - V_{i+1}(x_k)| < \varepsilon$, goto Step 5; else, goto Step 4.

- Step 4. If $i < i_{\max}$, then goto Step 1; else, goto Step 6.

- Step 5. The optimal control law is achieved as $u^*(x_k) = v_i(x_k)$. Stop the algorithm.

- Step 6. The optimal control law is not achieved within i_{\max} iterations.
-

4.3 Numerical Simulation and Analysis

To evaluate the performance of our policy iteration algorithm, four examples are chosen: (1) a linear system, (2) a discrete-time nonlinear system, (3) a torsional pendulum system, and (4) a complex nonlinear system.

Example 4.3.1 In the first example, a linear system is considered. The results will be compared to the traditional linear quadratic regulation (LQR) method to verify the effectiveness of the present algorithm. We consider the following linear system

$$x_{k+1} = Ax_k + Bu_k, \quad (4.3.1)$$

where $x_k = [x_{1k}, x_{2k}]^\top$ and $u_k \in \mathbb{R}$. Let the system matrices be expressed as

$$A = \begin{bmatrix} 0 & 0.1 \\ 0.3 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}. \quad (4.3.2)$$

The initial state is $x_0 = [1, -1]^\top$. Let the cost function be expressed by (4.2.2). The utility function is the quadratic form that is expressed as $U(x_k, u_k) = x_k^\top Q x_k + u_k^\top R u_k$, where Q is the 2×2 identity matrix, and $R = 0.5$.

Neural networks are used to implement the present policy iteration algorithm. The critic network and the action network are chosen as three-layer BP neural networks with the structures of 2–8–1 and 2–8–1, respectively. For each iteration step, the critic network and the action network are trained for 80 steps so that the neural network training error becomes less than 10^{-5} . The initial admissible control law is obtained by Algorithm 4.2.1, where the weights and thresholds of action network are obtained as

$$V_{a,\text{initial}} = \begin{bmatrix} -4.1525 & -1.1980 \\ 0.3693 & -0.8828 \\ 1.8071 & 2.8088 \\ 0.4104 & -0.9845 \\ 0.7319 & -1.7384 \\ 1.2885 & -2.5911 \\ -0.3403 & 0.8154 \\ -0.5647 & 1.3694 \end{bmatrix},$$

$$W_{a,\text{initial}} = [-0.0010, -0.2566, 0.0001, -0.1409, -0.0092, 0.0001, 0.3738, 0.0998],$$

and

$$b_{a,\text{initial}} = [3.5272, -0.9609, -1.8038, -0.0970, 0.8526, 1.1966, -1.0948, 2.5641]^\top,$$

respectively. Implement the policy iteration algorithm for 6 iterations to reach the computation precision $\varepsilon = 10^{-5}$, and the convergence trajectory of the iterative value functions is shown in Fig. 4.1a. During each iteration, the iterative control law is updated. Applying the iterative control law to the given system (4.3.1) for $T_f = 15$ time steps, we can obtain the states and the controls, which are displayed in Fig. 4.1b, c respectively.

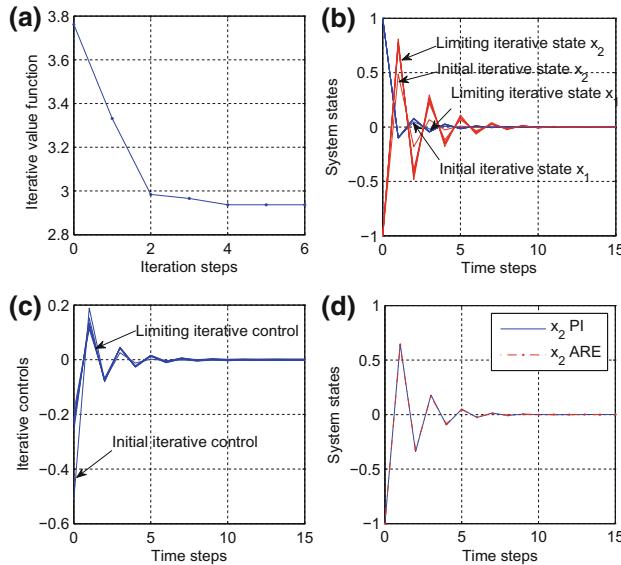


Fig. 4.1 Numerical results of Example 4.3.1. **a** The convergence trajectory of iterative value function. **b** The states. **c** The controls. **d** The trajectories of x_2 under the optimal control of policy iteration and ARE

We can see that the optimal control law of system (4.3.1) is obtained after 6 iterations. On the other hand, it is known that the solution of the optimal control problem for linear systems is quadratic in the state given by $J^*(x_k) = x_k^T P x_k$, where P is the solution of the algebraic Riccati equation (ARE) [7]. The solution of the ARE for the given linear system (4.3.1) can be determined as $P = \begin{bmatrix} 1.091 & -0.309 \\ -0.309 & 2.055 \end{bmatrix}$. The optimal control can be obtained as $u^*(x_k) = [-0.304, 1.029] x_k$. Applying this optimal control law to the linear system (4.3.1), we can obtain the same optimal control results. The optimal trajectories of system state x_2 under the optimal control laws by the present policy iteration algorithm, and ARE are displayed in Fig. 4.1d, respectively.

Example 4.3.2 The second example is chosen from the example in [2, 20]. We consider the following nonlinear system

$$x_{k+1} = f(x_k) + g(x_k)u_k, \quad (4.3.3)$$

where $x_k = [x_{1k}, x_{2k}]^T$ and u_k are the state and control variables, respectively. The system functions are given as $f(x_k) = \begin{bmatrix} 0.2x_{1k} \exp(x_{2k}^2) \\ 0.3x_{2k}^3 \end{bmatrix}$, $g(x_k) = \begin{bmatrix} 0 \\ -0.2 \end{bmatrix}$. The initial state is $x_0 = [2, -1]^T$. In this example, two utility functions, which are quadratic and nonquadratic forms, will be considered, respectively. The first utility function

is a quadratic form which is the same as the one in Example 4.3.1, where Q and R are chosen as identity matrices. The configurations of the critic network and the action network are chosen the same as the ones in Example 4.3.1. For each iteration step, the critic network and the action network are trained for 100 steps so that the neural network training error becomes less than 10^{-5} . Implement the policy iteration algorithm for 6 iterations to reach the computation precision of 10^{-5} . The convergence trajectories of the iterative value functions are shown in Fig. 4.2a. Applying the optimal control law to the given system (4.3.3) for $T_f = 10$ time steps, we can obtain the iterative states trajectories and control, which are displayed in Fig. 4.2b–d, respectively.

In [2], it is proven that the optimal control law can be obtained by the value iteration algorithm described in (4.2.22). The convergence trajectory of the iterative value function is shown in Fig. 4.3a. The optimal trajectories of system states and control are displayed in Fig. 4.3b–d, respectively.

Next, we change the utility function into a nonquadratic form as in [21] with modifications, where the utility function is expressed as

$$U(x_k, u_k) = \ln(x_k^T Q x_k + 1) + \ln(x_k^T Q x_k + 1) u_k^T R u_k.$$

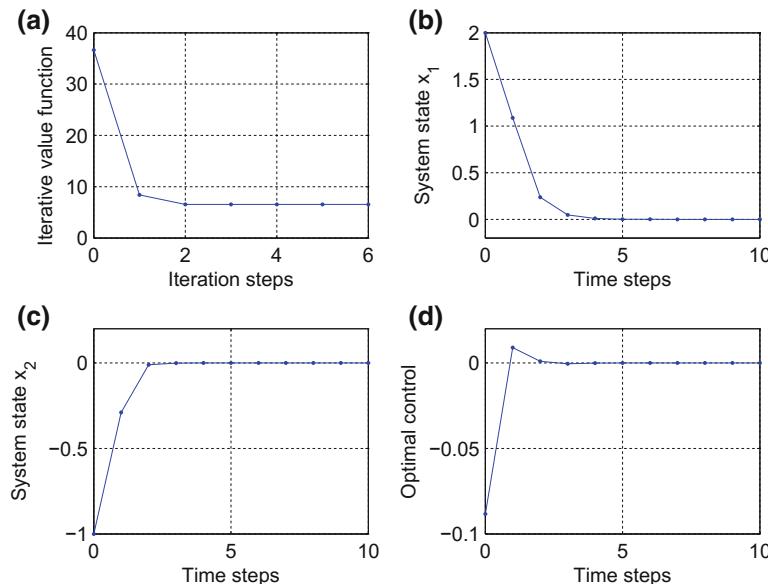


Fig. 4.2 Numerical results using policy iteration algorithm. **a** The convergence trajectory of iterative value function. **b** The optimal trajectory of state x_1 . **c** The optimal trajectory of state x_2 . **d** The optimal control

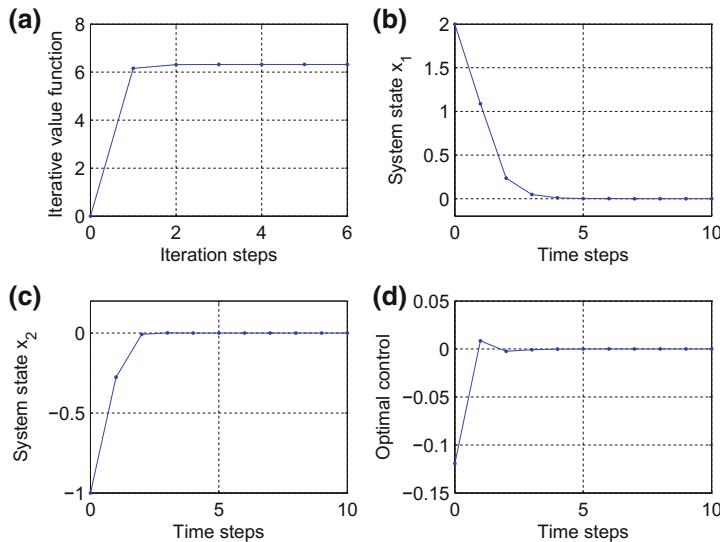


Fig. 4.3 Numerical results using value iteration algorithm. **a** The convergence trajectory of iterative value function. **b** The optimal trajectory of state x_1 . **c** The optimal trajectory of state x_2 . **d** The optimal control

Let the other parameters keep unchanged. Using the present policy iteration algorithm, we can also obtain the optimal control law for the system. The value function is shown in Fig. 4.4a. The optimal trajectories of iterative states and controls are shown in Fig. 4.4b–d, respectively.

Remark 4.3.1 From the numerical results, we can see that for quadratic and non-quadratic utility functions, the optimal control law of the nonlinear system can both be effectively obtained. On the other hand, we have shown that using the value iteration algorithm in [2], we can also obtain the optimal control law of the system. But we should point out that the convergence properties of the iterative value functions by the policy and value iteration algorithms are obviously different. Thus, stability behaviors of control systems under the two iterative control algorithms will be quite different. In the next example, detailed comparisons will be displayed.

Example 4.3.3 We now examine the performance of the present algorithm in a torsional pendulum system [15]. The dynamics of the pendulum is as follows

$$\begin{cases} \frac{d\theta}{dt} = \omega, \\ J \frac{d\omega}{dt} = u - Mgl \sin \theta - f_d \frac{d\theta}{dt}, \end{cases}$$

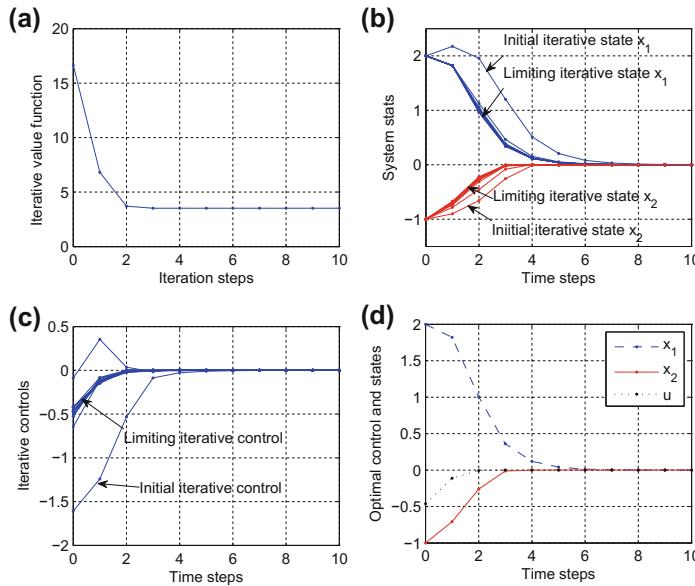


Fig. 4.4 Numerical results for nonquadratic utility function using policy iteration algorithm. **a** The convergence trajectory of iterative value function. **b** The states. **c** The controls. **d** The optimal states and control

where $M = 1/3 \text{ kg}$ and $l = 2/3 \text{ m}$ are the mass and length of the pendulum bar, respectively. The system states are the current angle θ and the angular velocity ω . Let $J = 4/3 M l^2$ and $f_d = 0.2$ be the rotary inertia and frictional factor, respectively. Let $g = 9.8 \text{ m/s}^2$ be the gravity. Discretization of the system function and cost function using Euler and trapezoidal methods [4] with the sampling interval $\Delta t = 0.1 \text{ s}$ leads to

$$\begin{bmatrix} x_{1(k+1)} \\ x_{2(k+1)} \end{bmatrix} = \begin{bmatrix} 0.1x_{2k} + x_{1k} \\ -0.49 \times \sin(x_{1k}) - 0.1 \times f_d \times x_{2k} + x_{2k} \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u_k, \quad (4.3.4)$$

where $x_{1k} = \theta_k$ and $x_{2k} = \omega_k$. Let the initial state be $x_0 = [1, -1]^T$. Let the utility function be quadratic form and the structures of the critic and action networks are 2–12–1 and 2–12–1. The initial admissible control law is obtained by Algorithm 4.2.1, where the weight matrices are obtained as

$$V_{a,\text{initial}} = \begin{bmatrix} -0.6574 & 1.1803 \\ 1.5421 & -2.9447 \\ -4.3289 & -3.7448 \\ 5.7354 & 2.8933 \\ 0.4354 & -0.8078 \\ -1.9680 & 3.6870 \\ 1.9285 & 1.4044 \\ -4.9011 & -4.3527 \\ 1.0914 & -0.0344 \\ -1.5746 & 2.8033 \\ 1.4897 & -0.0315 \\ 0.2992 & -0.0784 \end{bmatrix},$$

$$W_{a,\text{initial}} = [-2.1429, 1.9276, 0.0060, 0.0030, 4.5618, 3.2266, 0.0005, -0.0012, 1.3796, -0.5338, 0.5043, -5.1110],$$

and

$$b_{a,\text{initial}} = [0.8511, 4.2189, 5.7266, -6.0599, 0.4998, -5.7323, -0.6220, -0.5142, -0.3874, 3.3985, 0.6668, -0.2834]^T.$$

For each iteration step, the critic network and the action network are trained for 400 steps so that the neural network training error becomes less than 10^{-5} . Implement the policy iteration algorithm for 16 iterations to reach the computation precision 10^{-5} , and the convergence trajectory of the iterative value function is shown in Fig. 4.5a. Apply the iterative control laws to the given system for $T_f = 100$ time steps and the trajectories of the iterative control and states are displayed in Fig. 4.5b, c respectively. The optimal trajectories of system states and control are displayed in Fig. 4.5d.

From the numerical results, we can see that using the present policy iteration algorithm, any of the iterative control law can stabilize the system. But for the value iteration algorithm, the situation is quite different. For the value iteration algorithm, we choose the initial value function $V_0(x_k) \equiv 0$ as in [2]. Run the value iteration algorithm (4.2.22) for 30 iterations to reach the computation precision 10^{-5} and trajectory of the value function is expressed in Fig. 4.6a. Applying the iterative control law to the given system (4.3.4) for $T_f = 100$ time steps, we can obtain the iterative states and iterative controls, which are displayed in Fig. 4.6b, c respectively. The optimal trajectories of control and system states are displayed in Fig. 4.6d.

For unstable control systems, although the optimal control law can be obtained by value and policy iteration algorithms, for value iteration algorithm, we can see that not all the controls can stabilize the system. Moreover, the properties of the iterative

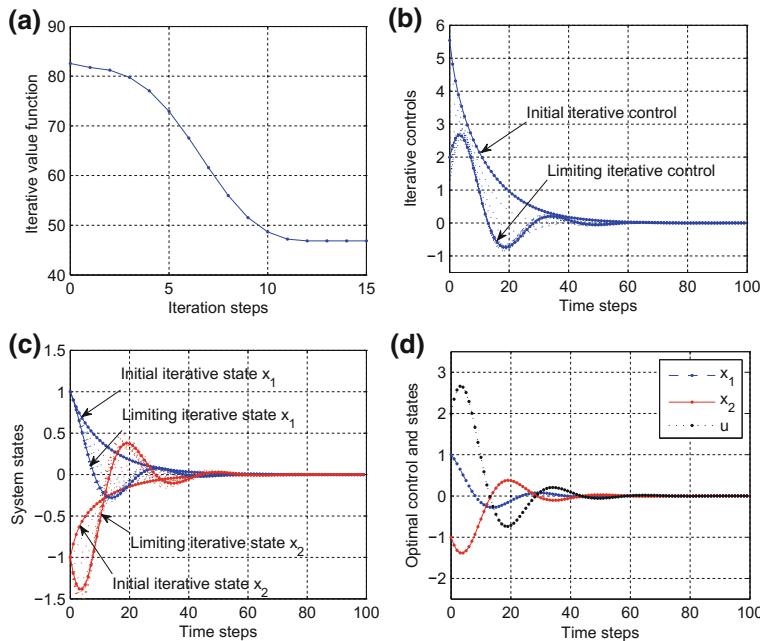


Fig. 4.5 Numerical results of Example 4.3.2 using policy iteration algorithm. **a** The convergence trajectory of iterative value function. **b** The controls. **c** The states. **d** The optimal states and control

controls obtained by the value iteration algorithm cannot be analyzed, and thus, the value iteration algorithm can only be implemented off-line. For the present policy iteration algorithm, the stability property can be guaranteed. Hence, we can declare the effectiveness of the policy iteration algorithm in this chapter.

Example 4.3.4 As a real world application of the present method, the problem of nonlinear satellite attitude control has been selected [5, 22]. Satellite dynamics is represented by

$$\frac{d\omega}{dt} = \Upsilon^{-1} (N_{\text{net}} - \omega \times \Upsilon \omega),$$

where Υ , ω , and N_{net} are inertia tensor, angular velocity vector of the body frame with respect to inertial frame, and the vector of the total torque applied on the satellite, respectively. The selected satellite is an inertial pointing satellite. Hence, we are interested in its attitude with respect to the inertial frame. All vectors are represented in the body frame and the sign \times denotes the cross product of two vectors. Let

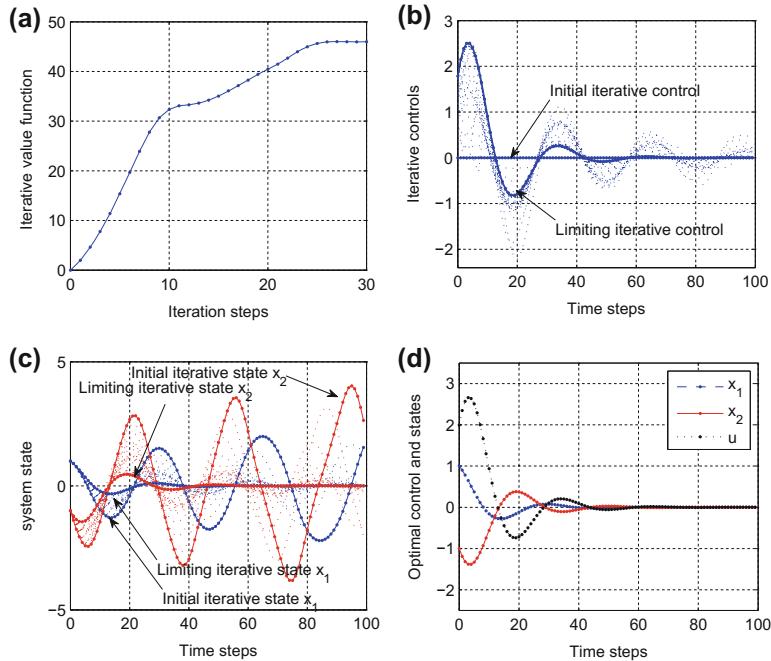


Fig. 4.6 Numerical results of Example 4.3.2 using value iteration algorithm. **a** The convergence trajectory of iterative value function. **b** The controls. **c** The states. **d** The optimal states and control

$N_{\text{net}} = N_{\text{ctrl}} + N_{\text{dis}}$, where N_{ctrl} is the control, and N_{dis} is disturbance. Following [22] and its order of transformation, the kinematic equation of the satellite becomes

$$\frac{d}{dt} \begin{bmatrix} \phi \\ \theta \\ \Psi \end{bmatrix} = \begin{bmatrix} 1 & \sin \phi \tan \phi & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi / \cos \theta & \cos \phi / \cos \theta \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \quad (4.3.5)$$

where ϕ , θ , and Ψ are the three Euler angles describing the attitude of the satellite with respect to x , y , and z axes of the inertial coordinate system, respectively. Subscripts x , y , and z are the corresponding elements of the angular velocity vector ω . The three Euler angles and the three elements of the angular velocity form the elements of the state space for the satellite attitude control problem. The state equation is given as follows,

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

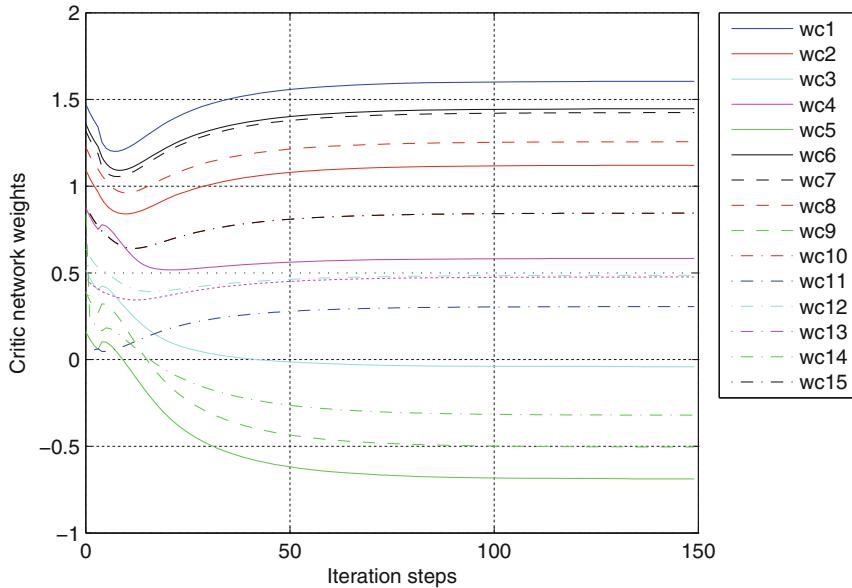


Fig. 4.7 The convergence trajectories of the critic network

where

$$\begin{aligned} x &= [\phi, \theta, \Psi, \omega_x, \omega_y, \omega_z]^T, \\ u &= [u_x, u_y, u_z]^T, \\ f(x) &= \begin{bmatrix} M_{3 \times 1} \\ \Upsilon^{-1}(N_{\text{dis}} - \omega \times \Upsilon \omega) \end{bmatrix}, \\ g(x) &= \begin{bmatrix} 0_{3 \times 3} \\ \Upsilon^{-1} \end{bmatrix}, \end{aligned}$$

and $M_{3 \times 1}$ denotes the right hand side of (4.3.5). The three-by-three null matrix is denoted by $0_{3 \times 3}$. The moment of inertia matrix of the satellite is chosen as

$$\Upsilon = \begin{bmatrix} 100 & 2 & 0.5 \\ 2 & 100 & 1 \\ 0.5 & 1 & 110 \end{bmatrix} \text{ kg} \cdot \text{m}^2.$$

The initial states are 60° , 20° , and 70° for the Euler angles of ϕ , θ , and Ψ , respectively, and -0.001 , 0.001 , and 0.001 rad/s for the angular rates around x , y , and z axes, respectively. For convenience of analysis, we assume that there is no disturbance in the system. Discretization of the system function and value function using Euler and trapezoidal methods [4] with the sampling interval $\Delta t = 0.25$ s leads to

$$x_{k+1} = (\Delta t f(x_k) + x_k) + \Delta t g(x_k) u_k.$$

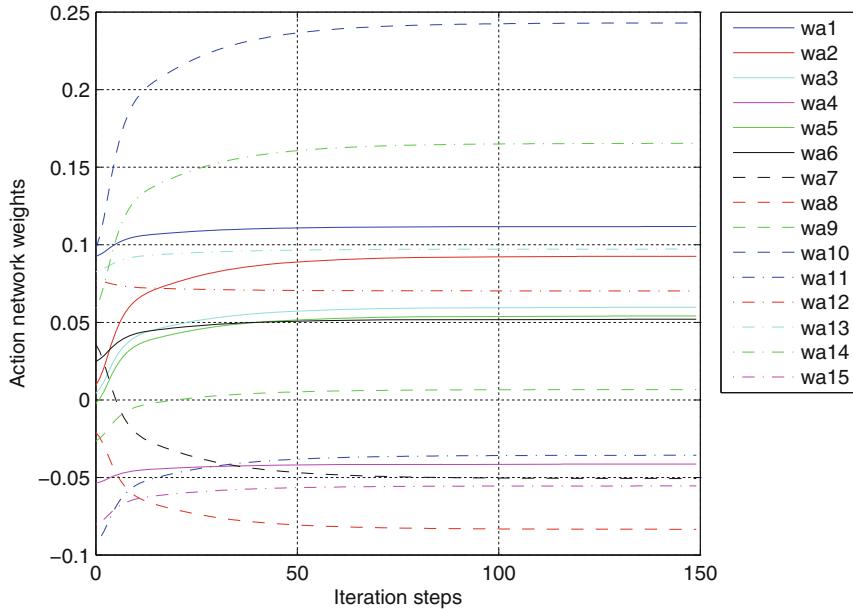


Fig. 4.8 The convergence trajectories of the first column of weights of the action network

Let the utility function be quadratic form where the state and control weight matrices are selected as

$$Q = \text{diag}\{0.25, 0.25, 0.25, 25, 25, 25\}$$

and

$$R = \text{diag}\{25, 25, 25\},$$

respectively.

Neural networks are used to implement the present policy iteration algorithm. The critic network and the action network are chosen as three-layer BP neural networks with the structures of 6–15–1 and 6–15–3, respectively. For each iteration step, the critic network and the action network are trained for 800 steps so that the neural network training error becomes less than 10^{-5} . Implement the policy iteration algorithm for 150 iterations. We have proven that the weights of critic and action networks are convergent in each iteration and thus convergent to the optimal ones. The convergence trajectories of critic network weights are shown Fig. 4.7. The weight convergence trajectories of the first column of action network are shown in Fig. 4.8.

The iterative value functions are shown in Fig. 4.9a. After the weights of the critic and action networks are convergent, we apply the neuro-optimal controller to the given system for $T_f = 1500$ time steps. The optimal state trajectories of ϕ , θ and ψ are shown in Fig. 4.9b. The trajectories of angular velocities ω_x , ω_y , and ω_z are shown

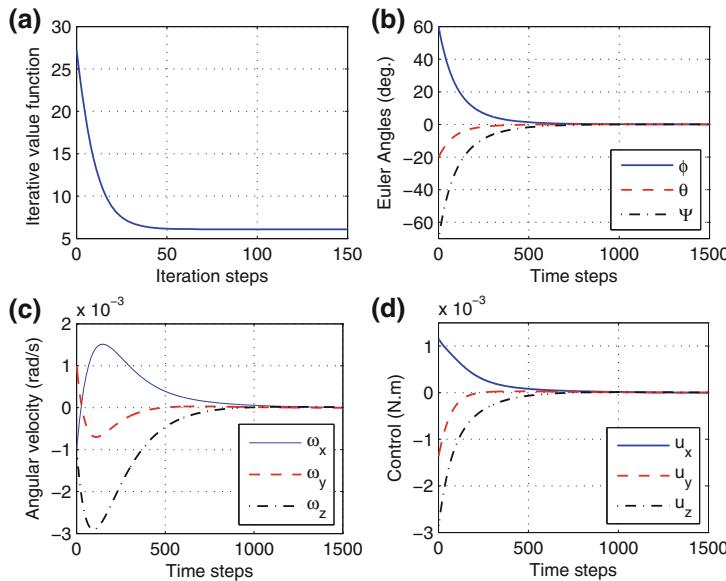


Fig. 4.9 Numerical results of Example 4.3.4 using policy iteration algorithm. **a** The convergence trajectory of iterative value function. **b** The trajectories of angular velocities ϕ , θ , and Ψ . **c** The trajectories of angular velocities ω_x , ω_y , and ω_z . **d** The optimal control trajectories

in Fig. 4.9c, and the optimal control signals are shown in Fig. 4.9d. The numerical results illustrate the effectiveness of the present policy iteration algorithm.

4.4 Conclusions

In this chapter, an effective policy iteration ADP algorithm is developed to solve the infinite horizon optimal control problems for discrete-time nonlinear systems. It is shown that any of the iterative control laws can stabilize the control system. It is proven that the iterative value functions are monotonically nonincreasing and convergent to the optimal solution of the Bellman equation. Neural networks are used to approximate the iterative value functions and compute the iterative control laws, respectively, for facilitating the implementation of the iterative ADP algorithm. Finally, four numerical examples are given to illustrate the performance of the method developed in this chapter.

References

1. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791
2. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2008) Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans Syst Man Cybern-Part B: Cybern* 38(4):943–949
3. Bhasin S, Kamalapurkar R, Johnson M, Vamvoudakis KG, Lewis FL, Dixon WE (2013) A novel actorcriticidentifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica* 49(1):82–92
4. Gupta SK (1995) Numerical methods for engineers. New Age International, India
5. Heydari A, Balakrishnan SN (2013) Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics. *IEEE Trans Neural Netw Learn Syst* 24(1):145–157
6. Huang T, Liu D (2013) A self-learning scheme for residential energy system control and management. *Neural Comput Appl* 22(2):259–269
7. Lewis FL, Syrmos VL (1995) Optimal control. Wiley, New York
8. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst Mag* 9(3):32–50
9. Lewis FL, Vrabie D, Vamvoudakis KG (2012) Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst* 32(6):76–105
10. Li H, Liu D (2012) Optimal control for discrete-time affine non-linear systems using general value iteration. *IET Control Theory Appl* 6(18):2725–2736
11. Liu D, Wei Q (2013) Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems. *IEEE Trans Cybern* 43(2):779–789
12. Liu D, Wei Q (2014) Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Trans Neural Netw Learn Syst* 25(3):621–634
13. Modares H, Lewis FL, Naghibi-Sistani MB (2013) Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks. *IEEE Trans Neural Netw Learn Syst* 24(10):1513–1525
14. Murray JJ, Cox CJ, Lendaris GG, Saeks R (2002) Adaptive dynamic programming. *IEEE Trans Syst Man Cybern-Part C: Appl Rev* 32(2):140–153
15. Si J, Wang YT (2001) Online learning control by association and reinforcement. *IEEE Trans Neural Netw* 12(2):264–276
16. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
17. Vamvoudakis KG, Lewis FL, Hudas GR (2012) Multi-agent differential graphical games: online adaptive learning solution for synchronization with optimality. *Automatica* 48(8):1598–1611
18. Vrabie D, Vamvoudakis KG, Lewis FL (2013) Optimal adaptive control and differential games by reinforcement learning principles. IET, London
19. Wang D, Liu D, Wei Q (2012) Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach. *Neurocomputing* 78(1):14–22
20. Wang FY, Jin N, Liu D, Wei Q (2011) Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ϵ -error bound. *IEEE Trans Neural Netw* 22(1):24–36
21. Wei Q, Zhang H, Dai J (2009) Model-free multiobjective approximate dynamic programming for discrete-time nonlinear systems with general performance index functions. *Neurocomputing* 72(7):1839–1848
22. Wertz JR (1978) Spacecraft attitude determination and control. Kluwer, Netherlands

23. Zhang H, Cui L, Zhang X, Luo Y (2011) Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Trans Neural Netw* 22(12):2226–2236
24. Zhang H, Song R, Wei Q, Zhang T (2011) Optimal tracking control for a class of nonlinear discrete-time systems with time delays based on heuristic dynamic programming. *IEEE Trans Neural Netw* 22(12):1851–1862
25. Zhang H, Wei Q, Liu D (2011) An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games. *Automatica* 47(1):207–214

Chapter 5

Generalized Policy Iteration ADP for Discrete-Time Nonlinear Systems

5.1 Introduction

According to [14, 15], most of the discrete-time reinforcement learning methods can be described as generalized policy iteration (GPI) algorithms. There are two revolving iteration procedures for GPI algorithms, which are policy evaluation, making the value function consistent with the current policy, and policy improvement, making a new policy that improves on the previous policy [14]. GPI allows one of these two iterations to be performed without completing the other step a priori. Investigations of GPI algorithms are important for the development of ADP. There exist inherent differences between GPI algorithms and the value and policy iteration algorithms. Analysis methods for traditional value and policy iteration algorithms are not valid for GPI algorithms anymore. For a long time, discussions on the properties of GPI algorithms for discrete-time control systems are scarce. To the best of our knowledge, only in [18], the properties of GPI algorithms were analyzed, while the stability property of the system under the iterative control law in [18] cannot be guaranteed. Therefore, together with the GPI algorithms to be developed in this chapter, we will establish convergence, admissibility, and optimality property analysis results as well [11, 18].

5.2 Generalized Policy Iteration-Based Adaptive Dynamic Programming Algorithm

Consider the following discrete-time nonlinear systems

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, \dots, \quad (5.2.1)$$

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. Let $F(x_k, u_k)$ denote the system function, which is known. For any $k = 0, 1, \dots$, let $\underline{u}_k = (u_k, u_{k+1}, \dots)$ be an arbitrary sequence of controls from k to ∞ . The cost function for state x_0 under the control sequence $\underline{u}_0 = (u_0, u_1, \dots)$ is defined as

$$J(x_0, \underline{u}_0) = \sum_{k=0}^{\infty} U(x_k, u_k), \quad (5.2.2)$$

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

We will study optimal control problems for (5.2.1). The goal is to find an optimal control law which stabilizes system (5.2.1) and simultaneously minimizes the cost function (5.2.2). For convenience of analysis, results of this chapter are based on the following assumptions (cf. Assumptions 2.2.1–2.2.3 in Chap. 2).

Assumption 5.2.1 $F(0, 0) = 0$, and the state feedback control law $u(\cdot)$ satisfies $u(0) = 0$, i.e., $x_k = 0$ is an equilibrium state of system (5.2.1) under the control $u_k = 0$.

Assumption 5.2.2 $F(x_k, u_k)$ is Lipschitz continuous on a compact set $\Omega \subset \mathbb{R}^n$ containing the origin.

Assumption 5.2.3 System (5.2.1) is controllable in the sense that there exists a continuous control law on Ω that asymptotically stabilizes the system.

Assumption 5.2.4 The utility function $U(x_k, u_k)$ is a continuous positive-definite function of x_k and u_k .

For a given control law $\mu(\cdot)$, the control in (5.2.1) is given by $u_k = \mu(x_k)$ and the cost function (5.2.2) is rewritten as

$$J^\mu(x_0) = \sum_{k=0}^{\infty} U(x_k, \mu(x_k)).$$

The optimal cost function is denoted by

$$J^*(x_0) = \inf_{\mu} \{J^\mu(x_0)\} = \inf_{\underline{u}_0} \{J(x_0, \underline{u}_0)\}, \quad (5.2.3)$$

where

$$\underline{u}_0 = (u_0, u_1, \dots) = (\mu(x_0), \mu(x_1), \dots).$$

According to Bellman's principle of optimality, for all $x_k \in \Omega$, $J^*(x_k)$ satisfies Bellman equation

$$\begin{aligned} J^*(x_k) &= \min_{u_k} \{U(x_k, u_k) + J^*(x_{k+1})\} \\ &= \min_{u_k} \{U(x_k, u_k) + J^*(F(x_k, u_k))\}. \end{aligned} \quad (5.2.4)$$

The optimal control law at time k is given by

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

Then, for all $x_k \in \Omega$, the Bellman equation can be written as

$$J^*(x_k) = U(x_k, u_k^*) + J^*(F(x_k, u_k^*)).$$

We have studied value iteration (VI) algorithms as well as policy iteration (PI) algorithms in previous chapters. Both VI and PI algorithms have their own advantages. It is desirable to develop a scheme that combines the two so that we can enjoy the benefits of both algorithms. This motivates the work in this chapter, and we call the new scheme as generalized policy iteration (GPI) [11].

5.2.1 Derivation and Properties of the GPI Algorithm

A. Derivation of the GPI Algorithm

The present generalized policy iteration algorithm contains two iteration procedures, which are called the i -iteration and the j -iteration, respectively. We introduce two iteration indices i and j_i and both of them increase from 0.

Let $\mathcal{A}(\Omega)$ be the set of admissible control laws. Let $v_{(-1)}(x_k) \in \mathcal{A}(\Omega)$ be an arbitrary admissible control law for all $x_k \in \Omega$. Let $\{N_1, N_2, \dots\}$ be a sequence of positive integers.

For $i = 0$, construct a value function $V_0(x_k)$ based on $v_{(-1)}(x_k)$ from the following generalized Bellman equation

$$V_0(x_k) = U(x_k, v_{(-1)}(x_k)) + V_0(F(x_k, v_{(-1)}(x_k))), \quad (5.2.5)$$

and calculate the control law $v_0(x_k)$ for all $x_k \in \Omega$ by

$$\begin{aligned} v_0(x_k) &= \arg \min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\} \\ &= \arg \min_{u_k} \{U(x_k, u_k) + V_0(F(x_k, u_k))\}. \end{aligned} \quad (5.2.6)$$

For $i = 1, 2, \dots$ and for all $x_k \in \Omega$, the generalized policy iteration algorithm can be expressed by the following two iterative procedures.

j -iteration:

For $j_i = 1, 2, \dots, N_i$, for all $x_k \in \Omega$, we update the iterative value function by

$$\begin{aligned} V_{i, j_i}(x_k) &= U(x_k, v_{i-1}(x_k)) + V_{i, j_i-1}(x_{k+1}) \\ &= U(x_k, v_{i-1}(x_k)) + V_{i, j_i-1}(F(x_k, v_{i-1}(x_k))), \end{aligned} \quad (5.2.7)$$

where

$$V_{i,0}(x_{k+1}) = V_{i-1}(x_{k+1})$$

and $V_0(\cdot)$ is obtained in (5.2.5). For all $x_k \in \Omega$, define the iterative value function at the end of the j -iteration as

$$V_i(x_k) = V_{i,N_i}(x_k). \quad (5.2.8)$$

i -iteration:

For all $x_k \in \Omega$, the iterative control law is improved by

$$\begin{aligned} v_i(x_k) &= \arg \min_{u_k} \{U(x_k, u_k) + V_i(x_{k+1})\} \\ &= \arg \min_{u_k} \{U(x_k, u_k) + V_i(F(x_k, u_k))\}. \end{aligned} \quad (5.2.9)$$

From the above generalized policy iteration algorithm, we can see that for each j -iteration, it computes the iterative value function for the control law $v_{i-1}(x_k)$, which tries to solve the following generalized Bellman equation

$$V_{i,j_i}(x_k) = U(x, v_{i-1}(x_k)) + V_{i,j_i}(F(x_k, v_{i-1}(x_k))). \quad (5.2.10)$$

This iterative procedure is called “policy evaluation” [8, 14]. In this procedure, the iterative value function $V_{i,j_i}(x_k)$ is updated, while the control law is kept unchanged. For each i -iteration, based on the iterative value function $V_i(x_k)$ for some control law, we use it to find another policy that is better, or at least not worse. This procedure is known as “policy improvement” [8, 14]. In this iterative procedure, the control law is updated.

B. Properties of the GPI Algorithm

In this section, under the assumption that perfect function approximation is available, we will prove that for any $N_i > 0$ and for all $x_k \in \Omega$, the iterative value function $V_{i,j_i}(x_k)$ will converge to $J^*(x_k)$ as $i \rightarrow \infty$. The admissibility property of the iterative control law $v_i(x_k)$ will also be presented.

Theorem 5.2.1 *Let $v_0(x_k) \in \mathcal{A}(\Omega)$ be the admissible control law obtained by (5.2.6). For $i = 1, 2, \dots$ and for all $x_k \in \Omega$, let the iterative value function $V_{i,j_i}(x_k)$ and the iterative control law $v_i(x_k)$ be obtained by (5.2.7)–(5.2.9). Let $\{N_1, N_2, \dots\}$ be a sequence of nonnegative integers. Then, we have the following properties.*

(i) *For $i = 1, 2, \dots$ and $j_i = 1, 2, \dots, N_i + 1$, we have*

$$V_{i,j_i}(x_k) \leq V_{i,j_i-1}(x_k), \quad \forall x_k \in \Omega, \quad (5.2.11)$$

where $V_{i,N_i+1}(x_k)$ is also defined by (5.2.7).

- (ii) For $i = 1, 2, \dots$, let j_i and $j_{(i+1)}$ be arbitrary constant integers which satisfy $0 \leq j_i \leq N_i$ and $0 \leq j_{(i+1)} \leq N_{i+1}$, respectively. Then, we have

$$V_{i+1,j_{(i+1)}}(x_k) \leq V_{i,j_i}(x_k), \quad \forall x_k \in \Omega. \quad (5.2.12)$$

Proof The theorem can be proven in two steps. We first prove (5.2.11) by mathematical induction. Let $i = 1$. From $V_{1,0}(\cdot) = V_0(\cdot)$ and (5.2.7), we have for $j_1 = 1$,

$$\begin{aligned} V_{1,1}(x_k) &= U(x_k, v_0(x_k)) + V_{1,0}(F(x_k, v_0(x_k))) \\ &= U(x_k, v_0(x_k)) + V_0(F(x_k, v_0(x_k))) \\ &= \min_{u_k} \{U(x_k, u_k) + V_0(F(x_k, u_k))\} \\ &\leq U(x_k, v_{(-1)}(x_k)) + V_0(F(x_k, v_{(-1)}(x_k))) \\ &= V_0(x_k) \\ &= V_{1,0}(x_k). \end{aligned} \quad (5.2.13)$$

Assume that

$$V_{1,j_1}(x_k) \leq V_{1,j_1-1}(x_k)$$

holds for $i = 1$ and $j_1 = l_1$, $l_1 = 1, 2, \dots, N_1$. Then, for $j_1 = l_1 + 1$, we have

$$\begin{aligned} V_{1,l_1+1}(x_k) &= U(x_k, v_0(x_k)) + V_{1,l_1}(F(x_k, v_0(x_k))) \\ &\leq U(x_k, v_0(x_k)) + V_{1,l_1-1}(F(x_k, v_0(x_k))) \\ &= V_{1,l_1}(x_k). \end{aligned} \quad (5.2.14)$$

Hence, (5.2.11) holds for $i = 1$. According to (5.2.9), the iterative control law $v_1(x_k)$ is expressed by

$$\begin{aligned} v_1(x_k) &= \arg \min_{u_k} \{U(x_k, u_k) + V_1(x_{k+1})\} \\ &= \arg \min_{u_k} \{U(x_k, u_k) + V_1(F(x_k, u_k))\}, \end{aligned} \quad (5.2.15)$$

where

$$V_1(x_{k+1}) = V_{1,N_1}(x_{k+1}).$$

Next, let $i = 2$. From $V_{2,0}(\cdot) = V_1(\cdot)$, $V_1(\cdot) = V_{1,N_1}(\cdot)$, and (5.2.7), we can get for $j_2 = 1$ and for all $x_k \in \Omega$,

$$\begin{aligned} V_{2,1}(x_k) &= U(x_k, v_1(x_k)) + V_{2,0}(F(x_k, v_1(x_k))) \\ &= U(x_k, v_1(x_k)) + V_1(F(x_k, v_1(x_k))) \end{aligned}$$

$$\begin{aligned}
&= \min_{u_k} \{U(x_k, u_k) + V_1(F(x_k, u_k))\} \\
&\leq U(x_k, v_0(x_k)) + V_1(F(x_k, v_0(x_k))) \\
&= U(x_k, v_0(x_k)) + V_{1,N_1}(F(x_k, v_0(x_k))) \\
&= V_{1,N_1+1}(x_k) \\
&\leq V_{1,N_1}(x_k) \\
&= V_1(x_k) \\
&= V_{2,0}(x_k).
\end{aligned} \tag{5.2.16}$$

So, (5.2.11) holds for $i = 2$ and $j_2 = 1$. Assume that (5.2.11) holds for $j_2 = l_2$, $l_2 = 1, 2, \dots, N_2$. Then, for $j_2 = l_2 + 1$, we obtain

$$\begin{aligned}
V_{2,l_2+1}(x_k) &= U(x_k, v_1(x_k)) + V_{2,l_2}(F(x_k, v_1(x_k))) \\
&\leq U(x_k, v_1(x_k)) + V_{2,l_2-1}(F(x_k, v_1(x_k))) \\
&= V_{2,l_2}(x_k).
\end{aligned} \tag{5.2.17}$$

Then, (5.2.11) holds for $i = 2$. Assume that (5.2.11) holds for $i = r$, $r = 1, 2, \dots$, i.e.,

$$V_{r,j_r}(x_k) \leq V_{r,j_r-1}(x_k).$$

For all $x_k \in \mathcal{Q}$, the iterative control law can be updated by

$$\begin{aligned}
v_r(x_k) &= \arg \min_{u_k} \{U(x_k, u_k) + V_r(x_{k+1})\} \\
&= \arg \min_{u_k} \{U(x_k, u_k) + V_r(F(x_k, u_k))\},
\end{aligned}$$

where $V_r(x_{k+1}) = V_{r,N_r}(x_{k+1})$. Then, for $i = r + 1$, we have $V_{r+1,0}(x_k) = V_r(x_k)$. According to (5.2.7), for $j_{r+1} = 1$, we have

$$\begin{aligned}
V_{r+1,1}(x_k) &= U(x_k, v_r(x_k)) + V_{r+1,0}(F(x_k, v_r(x_k))) \\
&= U(x_k, v_r(x_k)) + V_r(F(x_k, v_r(x_k))) \\
&= \min_{u_k} \{U(x_k, u_k) + V_r(F(x_k, u_k))\} \\
&\leq U(x_k, v_{r-1}(x_k)) + V_r(F(x_k, v_{r-1}(x_k))) \\
&= U(x_k, v_{r-1}(x_k)) + V_{r,N_r}(F(x_k, v_{r-1}(x_k))) \\
&= V_{r,N_r+1}(x_k) \\
&\leq V_{r,N_r}(x_k) \\
&= V_r(x_k) \\
&= V_{r+1,0}(x_k).
\end{aligned}$$

Table 5.1 The iterative process of the GPI algorithm in (5.2.5)–(5.2.9)

$v_{(-1)} \rightarrow V_0$ (5.2.5) evaluation	$v_0 \rightarrow V_{1,0} \rightarrow V_{1,1}$ $\rightarrow \dots \rightarrow V_{1,N_1} = V_1$ (5.2.7)–(5.2.8) N_1 -step evaluation	$v_1 \rightarrow V_{2,0} \rightarrow V_{2,1}$ $\rightarrow \dots \rightarrow V_{2,N_2} = V_2$ (5.2.7)–(5.2.8) N_2 -step evaluation	...
$V_0 \rightarrow v_0$ (5.2.6) minimization	$V_1 \rightarrow v_1$ (5.2.9) minimization	$V_2 \rightarrow v_2$ (5.2.9) minimization	...
$i = 0$	$i = 1$	$i = 2$...

So, (5.2.11) holds for $i = r + 1$ and $j_{r+1} = 1$. Assume that (5.2.11) holds for $j_{r+1} = l_{r+1}$, $l_{r+1} = 1, 2, \dots, N_{r+1}$. Then, for $j_{r+1} = l_{r+1} + 1$, we have

$$\begin{aligned} V_{r+1,l_{r+1}+1}(x_k) &= U(x_k, v_r(x_k)) + V_{r+1,l_{r+1}}(F(x_k, v_r(x_k))) \\ &\leq U(x_k, v_r(x_k)) + V_{r+1,l_{r+1}-1}(F(x_k, v_r(x_k))) \\ &= V_{r+1,l_{r+1}}(x_k). \end{aligned}$$

Hence, (5.2.11) holds for $i = 1, 2, \dots$ and $j_i = 1, 2, \dots, N_i + 1$. The mathematical induction is complete.

Next, we will prove inequality (5.2.12). For $i = 1$, let $1 \leq j_1 \leq N_1$ and $1 \leq j_2 \leq N_2$. According to (5.2.13)–(5.2.17), for all $x_k \in \Omega$, we have

$$\begin{cases} V_1(x_k) = V_{1,N_1}(x_k) \leq V_{1,j_1}(x_k) \leq V_{1,0}(x_k) = V_0(x_k), \\ V_2(x_k) = V_{2,N_2}(x_k) \leq V_{2,j_2}(x_k) \leq V_{2,0}(x_k) \leq V_1(x_k), \end{cases} \quad (5.2.18)$$

which shows that (5.2.12) holds for $i = 1$. Using mathematical induction, it is easy to prove that (5.2.12) holds for $i = 1, 2, \dots$. This completes the proof of the theorem.

The iterative process of the GPI algorithm in (5.2.5)–(5.2.9) is illustrated in Table 5.1. The algorithm starts with an admissible control law $v_{(-1)}(x_k)$ and it obtains $v_0(x_k)$ at the initialization step. Comparing between Tables 4.1 and 5.1, we note that starting from $i = 1$, “evaluation” in Table 4.1 is replaced by an iterative process in Table 5.1 for the calculation of $V_i(x_k)$. Clearly, “ N_i -step evaluation” in Table 5.1 is an approximation to the “evaluation” in Table 4.1. It can be shown following similar steps as in the proof of Theorem 2.3.3 that the limit $\lim_{j_i \rightarrow \infty} V_{i,j_i}(x_k) = V_{i,\infty}(x_k)$ exists and $V_{i,\infty}(x_k)$ is a solution to the evaluation equation in (5.2.10) since it is the same update as in VI algorithms (see Theorem 5.2.2). On the other hand, at the initialization step, it is also possible to design algorithms so that $V_0(\cdot)$ is also obtained by an iterative evaluation procedure instead of using (5.2.5) (See Algorithm 5.2.1).

Lemma 5.2.1 *Suppose that Assumptions 5.2.1–5.2.4 hold. For $i = 1, 2, \dots$, and for $j_i = 0, 1, \dots, N_i$, the iterative value function $V_{i,j_i}(x_k)$ is a positive-definite function of x_k .*

Proof Let $v_k^{(-1)} = \{v_{(-1)}(x_k), v_{(-1)}(x_{k+1}), \dots\}$. As $v_{(-1)}(x_k) \in \mathcal{A}(\Omega)$ is admissible, according to (5.2.5), for all $x_k \in \Omega$, the iterative value function

$$V_0(x_k) = \sum_{l=0}^{\infty} U(x_{k+l}, v_{(-1)}(x_{k+l})) \quad (5.2.19)$$

is finite. For $x_k = 0$, we have $v_{(-1)}(x_k) = 0$. According to Assumption 5.2.1, we can get $x_{k+1} = F(x_k, v_{(-1)}(x_k)) = F(0, 0) = 0$. By mathematical induction, for $l = 0, 1, \dots$, we have $x_{k+l} = 0$. According to (5.2.19) and Assumption 5.2.4, we can get $V_0(x_k) = 0$ when $x_k = 0$. On the other hand, according to Assumption 5.2.4, we have $V_0(x_k) \rightarrow \infty$, as $x_k \rightarrow \infty$. As $U(x_k, u_k) > 0$ for all $x_k \neq 0$, we have $V_0(x_k) > 0$ for all $x_k \neq 0$. Hence, $V_0(x_k)$ is a positive-definite function. According to (5.2.7)–(5.2.9), using mathematical induction, we can easily obtain that $V_{i,j_i}(x_k)$ is also positive definite. The proof is complete.

Theorem 5.2.2 For $i = 1, 2, \dots$ and $j_i = 0, 1, \dots, N_i$, let the iterative value function $V_{i,j_i}(x_k)$ and iterative control law $v_i(x_k)$ be obtained by (5.2.7)–(5.2.9). If for $i = 1, 2, \dots$, we let $N_i \rightarrow \infty$, then for all $x_k \in \Omega$ the iterative value function $V_{i,j_i}(x_k)$ is convergent as $j_i \rightarrow \infty$, i.e.,

$$V_{i,\infty}(x_k) = U(x, v_{i-1}(x_k)) + V_{i,\infty}(F(x_k, v_{i-1}(x_k))), \quad (5.2.20)$$

where

$$V_{i,\infty}(x_k) \triangleq \lim_{j_i \rightarrow \infty} V_{i,j_i}(x_k).$$

Proof According to (5.2.11), for $i = 1, 2, \dots$ and for all $x_k \in \Omega$, the iterative value function $V_{i,j_i}(x_k)$ is monotonically nonincreasing as j_i increases from 0 to N_i . On the other hand, according to Lemma 5.2.1, $V_{i,j_i}(x_k)$ is a positive-definite function for $i = 1, 2, \dots$ and $j_i = 0, 1, \dots, N_i$, i.e., $V_{i,j_i}(x_k) > 0, \forall x_k \neq 0$. This means that the iterative value function $V_{i,j_i}(x_k)$ is monotonically nonincreasing and lower bounded. Hence, for all $x_k \in \Omega$, the limit of $V_{i,j_i}(x_k)$ exists when $j_i \rightarrow \infty$. Then, we can obtain (5.2.20) directly. This completes the proof of the theorem.

Corollary 5.2.1 For $i = 1, 2, \dots$ and $j_i = 0, 1, \dots, N_i$, let the iterative value function $V_{i,j_i}(x_k)$ and iterative control law $v_i(x_k)$ be obtained by (5.2.7)–(5.2.9). Then, for $i = 1, 2, \dots$ and for all $x_k \in \Omega$, the iterative control law $v_i(x_k)$ is admissible.

Proof Considering $N_i > 0$. For $\bar{j}_i = N_i + 1, N_i + 2, \dots$ and for all $x_k \in \Omega$, we construct a value function $\mathcal{V}_{i,\bar{j}_i}(x_k)$ as

$$\mathcal{V}_{i,\bar{j}_i}(x_k) = U(x_k, v_{i-1}(x_k)) + \mathcal{V}_{i,\bar{j}_i-1}(F(x_k, v_{i-1}(x_k))), \quad (5.2.21)$$

where $\mathcal{V}_{i,N_i}(x_k) = V_{i,N_i}(x_k)$. According to (5.2.21), we can obtain

$$\mathcal{V}_{i,\bar{j}_i}(x_k) = \sum_{l=0}^{\bar{j}_i-N_i-1} U(x_{k+l}, v_{i-1}(x_{k+l})) + \mathcal{V}_{i,N_i}(x_{k+\bar{j}_i-N_i}).$$

According to Theorem 5.2.2, for all $x_k \in \mathcal{Q}$, the iterative value function $\mathcal{V}_{i,\infty}(x_k)$, which is expressed by

$$\mathcal{V}_{i,\infty}(x_k) = \lim_{\bar{j}_i \rightarrow \infty} \sum_{l=0}^{\bar{j}_i-N_i-1} U(x_{k+l}, v_{i-1}(x_{k+l})) + \lim_{\bar{j}_i \rightarrow \infty} \mathcal{V}_{i,N_i}(x_{k+\bar{j}_i-N_i}),$$

is finite. According to Assumption 5.2.4, the utility function $U(x_k, v_{i-1}(x_k)) > 0$, $\forall x_k \neq 0$. Then,

$$\lim_{k \rightarrow \infty} U(x_k, v_{i-1}(x_k)) = 0,$$

implies $x_k \rightarrow 0$ as $k \rightarrow \infty$. On the other hand, according to Lemma 5.2.1, $\mathcal{V}_{i,N_i}(x_k) = V_{i,N_i}(x_k)$ is positive definite. Thus, we can get

$$\lim_{\bar{j}_i \rightarrow \infty} \mathcal{V}_{i,N_i}(x_{k+\bar{j}_i-N_i}) = 0.$$

As

$$\sum_{l=0}^{N_i} U(x_{k+l}, v_{i-1}(x_{k+l}))$$

is finite, we have

$$\sum_{l=0}^{\infty} U(x_{k+l}, v_{i-1}(x_{k+l})) = \sum_{l=0}^{N_i} U(x_{k+l}, v_{i-1}(x_{k+l})) + \mathcal{V}_{i,\infty}(x_{k+N_i+1})$$

is also finite. The above shows that $v_{i-1}(x_k)$ is admissible. It is easy to conclude that $v_i(x_k)$ is also admissible. The proof is complete.

Next, the convergence property of the generalized policy iteration algorithm will be established. As the iteration index i increases to ∞ , we will show that the optimal cost function and optimal control law can be achieved using the present generalized policy iteration algorithm. Before the main theorem, some lemmas are necessary.

Lemma 5.2.2 (cf. [3]) *If a monotonically nonincreasing sequence $\{a_n\}$, $n = 0, 1, \dots$, contains an arbitrary convergent subsequence, then sequence $\{a_n\}$ is convergent.*

Lemma 5.2.3 *For $i = 1, 2, \dots$, let the iterative value function $V_i(x_k)$ be defined as in (5.2.8). Then, the iterative value function sequence $\{V_i(x_k)\}$ is monotonically nonincreasing and convergent.*

Lemma 5.2.3 is a direct consequence of Theorem 5.2.1.

Theorem 5.2.3 For $i = 0, 1, \dots$ and for all $x_k \in \Omega$, let $V_{i,j_i}(x_k)$ and $v_i(x_k)$ be obtained by (5.2.5)–(5.2.9). If Assumptions 5.2.1–5.2.4 hold, then for any $N_i > 0$, the iterative value function $V_{i,j_i}(x_k)$ converges to the optimal cost function $J^*(x_k)$, as $i \rightarrow \infty$, i.e.,

$$\lim_{i \rightarrow \infty} V_{i,j_i}(x_k) = J^*(x_k), \quad (5.2.22)$$

which satisfies the Bellman equation (5.2.4).

Proof Define a sequence of iterative value functions as

$$\begin{aligned} \{V_{i,j_i}(x_k)\} &\triangleq \{V_0(x_k), V_{1,0}(x_k), V_{1,1}(x_k), \dots, V_{1,N_1}(x_k), \\ &\quad V_1(x_k), V_{2,0}(x_k), \dots, V_{2,N_2}(x_k), \dots\}. \end{aligned}$$

If we let $\{V_i(x_k)\} \triangleq \{V_0(x_k), V_1(x_k), \dots\}$, then $\{V_i(x_k)\}$ is a subsequence of $\{V_{i,j_i}(x_k)\}$ which is monotonically nonincreasing. According to Lemma 5.2.3, the limit of $\{V_i(x_k)\}$ exists. From Lemma 5.2.2, we can get that if the sequence $\{V_i(x_k)\}$ is convergent, then $\{V_{i,j_i}(x_k)\}$ is also convergent. Since a sequence $\{V_{i,j_i}(x_k)\}$ can converge to at most one point [3], the sequence $\{V_{i,j_i}(x_k)\}$ and its subsequence $\{V_i(x_k)\}$ possess the same limit, i.e.,

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

Thus, we will prove

$$\lim_{i \rightarrow \infty} V_i(x_k) = J^*(x_k). \quad (5.2.23)$$

The statement (5.2.23) can be proven in three steps.

(1) Show that the limit of the iterative value function $V_i(x_k)$ satisfies the Bellman equation, as $i \rightarrow \infty$.

According to Lemma 5.2.3, for all $x_k \in \Omega$, we can define the value function $V_\infty(x_k)$ as the limit of the iterative value function $V_i(x_k)$, i.e., $V_\infty(x_k) = \lim_{i \rightarrow \infty} V_i(x_k)$. According to (5.2.7)–(5.2.9), we have

$$\begin{aligned} V_i(x_k) &= V_{i,N_i}(x_k) \\ &\leq V_{i,1}(x_k) \\ &= U(x_k, v_{i-1}(x_k)) + V_{i-1}(F(x_k, v_{i-1}(x_k))) \\ &= \min_{u_k} \{U(x_k, u_k) + V_{i-1}(F(x_k, u_k))\}. \end{aligned}$$

Then, we can obtain

$$V_\infty(x_k) = \lim_{i \rightarrow \infty} V_i(x_k) \leq V_i(x_k) \leq \min_{u_k} \{U(x_k, u_k) + V_{i-1}(F(x_k, u_k))\}.$$

Let $i \rightarrow \infty$. For all $x_k \in \Omega$, we can obtain

$$V_\infty(x_k) \leq \min_{u_k} \{U(x_k, u_k) + V_\infty(F(x_k, u_k))\}. \quad (5.2.24)$$

Let $\varepsilon > 0$ be an arbitrary positive number. Since $V_i(x_k)$ is nonincreasing for $i \geq 0$ and

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

there exists a positive integer p such that $V_p(x_k) - \varepsilon \leq V_\infty(x_k) \leq V_p(x_k)$. Hence, we can get

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

Since $\varepsilon > 0$ is arbitrary, for all $x_k \in \Omega$, we have

$$V_\infty(x_k) \geq \min_{u_k} \{U(x_k, u_k) + V_\infty(F(x_k, u_k))\}. \quad (5.2.25)$$

Combining (5.2.24) and (5.2.25), for all $x_k \in \Omega$, we can obtain

$$V_\infty(x_k) = \min_{u_k} \{U(x_k, u_k) + V_\infty(F(x_k, u_k))\}. \quad (5.2.26)$$

Next, for all $x_k \in \Omega$, let $\mu(x_k)$ be an arbitrary admissible control law, and define a new value function $P(x_k)$ as

$$P(x_k) = U(x_k, \mu(x_k)) + P(F(x_k, \mu(x_k))). \quad (5.2.27)$$

Then, we can proceed to the second step of the proof.

(2) *Show that for an arbitrary admissible control law $\mu(x_k)$, the value function $P(x_k)$ satisfies*

$$P(x_k) \geq V_\infty(x_k). \quad (5.2.28)$$

Please refer to part (2) of the proof of Theorem 4.2.2.

(3) *Show that the value function $V_\infty(x_k)$ equals to the optimal cost function $J^*(x_k)$.*

According to the definition of the optimal cost function $J^*(x_k)$ given in (5.2.3), for $i = 0, 1, \dots$ and for all $x_k \in \Omega$, we have $V_i(x_k) \geq J^*(x_k)$. Then, let $i \rightarrow \infty$. We can obtain $V_\infty(x_k) \geq J^*(x_k)$.

On the other hand, for an arbitrary admissible control law $\mu(x_k)$, (5.2.28) holds. For all $x_k \in \Omega$, let $\mu(x_k) = u^*(x_k)$, where $u^*(x_k)$ is the optimal control law. Then, we can get $V_\infty(x_k) \leq J^*(x_k)$. Hence, (5.2.22) holds. The proof is complete.

Corollary 5.2.2 *For $i = 0, 1, \dots$, let $V_{i,j_i}(x_k)$ and $v_i(x_k)$ be obtained by (5.2.5)–(5.2.9). If for $i = 1, 2, \dots$, let $j_i \equiv 1$, then the iterative value function $V_{i,j_i}(x_k)$ converges to the optimal cost function $J^*(x_k)$, as $i \rightarrow \infty$.*

5.2.2 GPI Algorithm and Relaxation of Initial Conditions

In the previous section, the monotonicity, convergence, optimality, and admissibility properties of the generalized policy iteration algorithm have been analyzed. From the generalized policy iteration algorithm (5.2.5)–(5.2.9), we can see that to implement the present algorithm, it requires an admissible control law $v_{(-1)}(x_k) \in \mathcal{A}(\Omega)$ to construct the initial value function $V_0(x_k)$ that satisfies (5.2.5). Usually, $v_{(-1)}(x_k) \in \mathcal{A}(\Omega)$ and $V_0(x_k)$ in (5.2.5) are difficult to obtain, which creates difficulties in the implementation of the present algorithm. In this section, some effective methods will be presented to relax the initial value function of the algorithm.

First, we consider the situation that the admissible control law $v_{(-1)}(x_k)$ is known. We develop a finite-step policy evaluation algorithm to relax the initial value function $V_0(x_k)$, where it is approximated by a finite iteration procedure instead of (5.2.5). The detailed implementation of the algorithm is expressed in Algorithm 5.2.1.

Lemma 5.2.4 *For $x_k \in \Omega$, let $\Psi(x_k) \geq 0$ be an arbitrary positive semidefinite function. Let $v_{(-1)}(x_k)$ be an arbitrary admissible control law and let $V_{0,j_0}(x_k)$ be the iterative value function updated by (5.2.29)–(5.2.31), where $V_{0,0}(x_k) = \Psi(x_k)$. Then, $V_{0,j_0}(x_k)$ is convergent as $j_0 \rightarrow \infty$.*

Proof According to (5.2.31), for all $x_k \in \Omega$, we have

$$\begin{aligned} V_{0,j_0+1}(x_k) - V_{0,j_0}(x_k) &= U(x_k, v_{(-1)}(x_k)) + V_{0,j_0}(x_{k+1}) \\ &\quad - (U(x_k, v_{(-1)}(x_k)) + V_{0,j_0-1}(x_{k+1})) \\ &= V_{0,j_0}(x_{k+1}) - V_{0,j_0-1}(x_{k+1}). \end{aligned} \quad (5.2.32)$$

According to (5.2.32), we can get

$$\left\{ \begin{array}{l} V_{0,j_0+1}(x_k) - V_{0,j_0}(x_k) = V_{0,1}(x_{k+j_0}) - V_{0,0}(x_{k+j_0}), \\ V_{0,j_0}(x_k) - V_{0,j_0-1}(x_k) = V_{0,1}(x_{k+j_0-1}) - V_{0,0}(x_{k+j_0-1}), \\ \vdots \\ V_{0,1}(x_k) - V_{0,0}(x_k) = V_{0,1}(x_k) - V_{0,0}(x_k). \end{array} \right.$$

Algorithm 5.2.1 Finite-step policy evaluation algorithm for initial value function**Initialization:**

Choose randomly an array of system states x_k in Ω , i.e., $X_k = \{x_k^{(1)}, x_k^{(2)}, \dots, x_k^{(p)}\}$, where p is a large positive integer.

Choose an arbitrary positive semidefinite function $\Psi(x_k) \geq 0$.

Determine an initial admissible control law $v_{(-1)}(x_k)$.

Iteration:

Step 1. Let the iteration index $j_0 = 0$ and let $V_{0,0}(x_k) = \Psi(x_k)$.

Step 2. For all $x_k \in X_k$, update the control law $v_0^{j_0}(x_k)$ by

$$v_0^{j_0}(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + V_{0,j_0}(F(x_k, u_k))\}, \quad (5.2.29)$$

and improve the iterative value function by

$$\begin{aligned} V_{1,0}^{j_0}(x_k) &= \min_{u_k} \{U(x_k, u_k) + V_{0,j_0}(F(x_k, u_k))\} \\ &= U(x_k, v_0^{j_0}(x_k)) + V_{0,j_0}(F(x_k, v_0^{j_0}(x_k))). \end{aligned} \quad (5.2.30)$$

Step 3. If $V_{1,0}^{j_0}(x_k) - V_{0,j_0}(x_k) \leq 0$ for all $x_k \in X_k$, goto Step 5; else, goto Step 4.

Step 4. Let $j_0 = j_0 + 1$. For all $x_k \in X_k$, update the iterative value function by

$$V_{0,j_0}(x_k) = U(x_k, v_{(-1)}(x_k)) + V_{0,j_0-1}(F(x_k, v_{(-1)}(x_k))). \quad (5.2.31)$$

Goto Step 2.

Step 5. Let $V_0(x_k) = V_{1,0}^{j_0}(x_k)$ and $v_0(x_k) = v_0^{j_0}(x_k)$.

The right-hand side can be written as

$$\left\{ \begin{array}{l} V_{0,1}(x_{k+j_0}) - V_{0,0}(x_{k+j_0}) = U(x_{k+j_0}, v_{(-1)}(x_{k+j_0})) + V_{0,0}(x_{k+j_0+1}) - V_{0,0}(x_{k+j_0}), \\ V_{0,1}(x_{k+j_0-1}) - V_{0,0}(x_{k+j_0-1}) = U(x_{k+j_0-1}, v_{(-1)}(x_{k+j_0-1})) + V_{0,0}(x_{k+j_0}) - V_{0,0}(x_{k+j_0-1}), \\ \vdots \\ V_{0,1}(x_k) - V_{0,0}(x_k) = U(x_k, v_{(-1)}(x_k)) + V_{0,0}(x_{k+1}) - V_{0,0}(x_k). \end{array} \right.$$

By adding all these equations together, we get

$$V_{0,j_0+1}(x_k) = \sum_{l=0}^{j_0} U(x_{k+l}, v_{(-1)}(x_{k+l})) + \Psi(x_{k+j_0+1}).$$

Let $j_0 \rightarrow \infty$. We can obtain

$$\lim_{j_0 \rightarrow \infty} V_{0,j_0+1}(x_k) = \sum_{l=0}^{\infty} U(x_{k+l}, v_{(-1)}(x_{k+l})).$$

As $v_{(-1)}(x_k)$ is an admissible control law,

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

is finite. Hence, we conclude that $\lim_{j_0 \rightarrow \infty} V_{0,j_0}(x_k)$ is finite, which means $V_{0,j_0}(x_k)$ is convergent as $j_0 \rightarrow \infty$. This completes the proof of the lemma.

Using the admissible control law $v_{(-1)}(x_k)$, from Lemma 5.2.4, $V_{0,j_0+1}(x_k) = V_{0,j_0}(x_k)$ holds as $j_0 \rightarrow \infty$. It means that there must exist $N_0 > 0$ such that $V_{1,0}^{N_0}(x_k) \leq V_{0,N_0}(x_k)$. Hence, if we have obtained an admissible control law, then we can construct the initial value function by finite-step policy evaluation to replace the value function $V_0(x_k)$ in (5.2.5). On the other hand, Algorithm 5.2.1 requires an admissible control law $v_{(-1)}(x_k)$ to implement. Usually, the admissible control law of the nonlinear system is difficult to obtain. To overcome this difficulty, a policy improvement algorithm can be implemented by experiment. The details are given in Algorithm 5.2.2.

Algorithm 5.2.2 Policy improvement algorithm for initial value function

Initialization:

Choose randomly an array of system states x_k in Ω , i.e., $X_k = \{x_k^{(1)}, x_k^{(2)}, \dots, x_k^{(p)}\}$, where p is a large positive integer. Let $\varsigma_0 = 0$.

Iteration:

Step 1. Choose arbitrarily a large positive definite function $\bar{\Psi}^{\varsigma_0}(x_k) \geq 0$, and let $V_{0,0}^{\varsigma_0}(x_k) = \bar{\Psi}^{\varsigma_0}(x_k)$.

Step 2. For all $x_k \in X_k$, update the control law $v_0^{\varsigma_0}(x_k)$ by

$$v_0^{\varsigma_0}(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + \bar{\Psi}^{\varsigma_0}(F(x_k, u_k))\}, \quad (5.2.33)$$

and for all $x_k \in X_k$, improve the iterative value function by

$$\begin{aligned} V_{1,0}^{\varsigma_0}(x_k) &= \min_{u_k} \{U(x_k, u_k) + V_{0,0}^{\varsigma_0}(F(x_k, u_k))\} \\ &= U(x_k, v_0^{\varsigma_0}(x_k)) + V_{0,0}^{\varsigma_0}(F(x_k, v_0^{\varsigma_0}(x_k))). \end{aligned} \quad (5.2.34)$$

Step 3. For all $x_k \in X_k$, if the inequality

$$V_{1,0}^{\varsigma_0}(x_k) - \bar{\Psi}^{\varsigma_0}(x_k) \leq 0 \quad (5.2.35)$$

holds, then goto Step 4; else, let $\varsigma_0 = \varsigma_0 + 1$ and goto Step 1.

Step 4. Let $V_0(x_k) = V_{1,0}^{\varsigma_0}(x_k)$ and $v_0(x_k) = v_0^{\varsigma_0}(x_k)$.

Theorem 5.2.4 For all $x_k \in \Omega$, let the iterative control law $v_1^{\varsigma_0}(x_k)$ be expressed as in (5.2.33) and let the iterative value function $V_{1,0}^{\varsigma_0}(x_k)$ be expressed as in (5.2.34). If the iterative value functions satisfy (5.2.35), then the convergence properties (5.2.11) and (5.2.12) of Theorem 5.2.1 hold for $i = 1, 2, \dots$ and $j_i = 0, 1, \dots, N_i$.

Proof Let $i = 1$ and $j_1 = 0$. As $V_0(x_k) = V_{1,0}^{\varsigma_0}(x_k)$ and $v_0(x_k) = v_0^{\varsigma_0}(x_k)$, according to (5.2.7) and (5.2.35), we have

$$\begin{aligned}
V_{1,1}(x_k) &= U(x_k, v_0(x_k)) + V_{1,0}(F(x_k, v_0(x_k))) \\
&= U(x_k, v_0^{\varsigma_0}(x_k)) + V_{1,0}^{\varsigma_0}(F(x_k, v_0^{\varsigma_0}(x_k))) \\
&\leq U(x_k, v_0^{\varsigma_0}(x_k)) + V_{0,0}^{\varsigma_0}(F(x_k, v_0^{\varsigma_0}(x_k))) \\
&= V_{1,0}(x_k).
\end{aligned}$$

Following similar steps as in the proof of Theorem 5.2.1, the convergence properties (5.2.11) and (5.2.12) can be shown for $i = 1, 2, \dots$ and $j_i = 0, 1, \dots, N_i$. This completes the proof of the theorem.

Remark 5.2.1 From Algorithm 5.2.2, we can see that the admissible control law $v_{(-1)}(x_k)$ in Algorithm 5.2.1 is avoided. This is a merit of Algorithm 5.2.2. However, in Algorithm 5.2.2, we should find a positive-definite function $\bar{\Psi}^{\varsigma_0}(x_k)$ that satisfies (5.2.35). As $\bar{\Psi}^{\varsigma_0}(x_k)$ is randomly chosen, it may take some iterations to determine $\bar{\Psi}^{\varsigma_0}(x_k)$. This is a disadvantage of the algorithm.

Based on the above preparations, we summarize the generalized policy iteration ADP algorithm in Algorithm 5.2.3.

Algorithm 5.2.3 Generalized policy iteration algorithm

Initialization:

Choose randomly an array of system states x_k in Ω , i.e., $X_k = \{x_k^{(1)}, x_k^{(2)}, \dots, x_k^{(p)}\}$, where p is a large positive integer.

Choose a computation precision ε .

Construct a sequence $\{N_i\}$, where $N_i > 0$, $i = 1, 2, \dots$, are nonnegative integers.

Iteration:

Step 1. Let the iteration index $i = 1$.

Step 2. Obtain $V_0(x_k)$ and $v_0(x_k)$ by Algorithm Υ , $\Upsilon = 5.2.1$ or $5.2.2$, instead of (5.2.5) and (5.2.6).

Step 3. Let j_i increase from 1 to N_i . For all $x_k \in X_k$, do **Policy Evaluation** according to (5.2.7),

$$V_{i,j_i}(x_k) = U(x_k, v_{i-1}(x_k)) + V_{i,j_i-1}(F(x_k, v_{i-1}(x_k))),$$

with $V_{i,0}(x_k) = V_{i-1}(x_k)$.

Step 4. Let $V_i(x_k) = V_{i,N_i}(x_k)$ as in (5.2.8).

Step 5. For all $x_k \in X_k$, do **Policy Improvement** as in (5.2.9),

$$v_i(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + V_i(F(x_k, u_k))\}.$$

Step 6. For all $x_k \in X_k$, if $|V_{i-1}(x_k) - V_i(x_k)| < \varepsilon$, then the optimal cost function and optimal control law are obtained, and goto Step 7; else, let $i = i + 1$ and goto Step 3.

Step 7. Return $V_i(x_k)$ and $v_i(x_k)$ as the optimal cost function $J^*(x_k)$ and the optimal control law $u^*(x_k)$.

5.2.3 Simulation Studies

To evaluate the performance of our generalized policy iteration algorithm, two examples are given for numerical experiments to solve the approximate optimal control problems.

Example 5.2.1 First, let us consider the following spring–mass–damper system [7]

$$M \frac{d^2y}{dt^2} + b \frac{dy}{dt} + \kappa y = u,$$

where y is the position and u is the control input. Let $M = 0.1$ kg denote the mass of object. Let $\kappa = 2$ kgf/m be the stiffness coefficient of spring and let $b = 0.1$ be the wall friction. Let $x_1 = y$, $x_2 = dy/dt$. Discretizing the system function with the sampling interval $\Delta t = 0.1$ s leads to

$$\begin{bmatrix} x_{1(k+1)} \\ x_{2(k+1)} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ -\frac{\kappa}{M} \Delta t & 1 - \frac{b}{M} \Delta t \end{bmatrix} \begin{bmatrix} x_{1k} \\ x_{2k} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\Delta t}{M} \end{bmatrix} u_k. \quad (5.2.36)$$

Let the initial state be $x_0 = [1, -1]^\top$. Let the cost function be expressed by (5.2.2). The utility function is expressed as $U(x_k, u_k) = x_k^\top x_k + u_k^2$.

Let the state space be $\Omega = \{x_k : -1 \leq x_{1k} \leq 1, -1 \leq x_{2k} \leq 1\}$. We randomly choose an array of $p = 5000$ states in Ω to implement the generalized policy iteration algorithm to obtain the optimal control law. Neural networks are used to implement the present generalized policy iteration algorithm. The critic network and the action network are chosen as three-layer backpropagation (BP) neural networks with the structures of 2–8–1 and 2–8–1, respectively. Define the two neural networks as group “NN1”. For system (5.2.36), we can obtain an admissible control law $u(x_k) = Kx_k$, where $K = [0.13, -0.17]$. Let $\Psi(x_k) = x_k^\top P_0 x_k$, where

$$P_0 = \begin{bmatrix} 80 & 1 \\ 1 & 2 \end{bmatrix}.$$

As the initial admissible control law $u(x_k) = Kx_k$ is known, the finite-step policy evaluation in Algorithm 5.2.1 is implemented. We can see that it takes 3 iterations to obtain $V_0(x_k)$ and $v_0(x_k)$ and the results for the initial iteration are displayed in Fig. 5.1. (See the trajectories of the iterative value functions for $i = 0$). Let iteration index $i_{\max} = 10$. To illustrate the effectiveness of the algorithm, we choose four different iteration sequences $\{N_i^\gamma\}$, $\gamma = 1, 2, 3, 4$. For $\gamma = 1$ and $i = 1, 2, \dots, 10$, we let $N_i^1 = 1$. For $\gamma = 2$, iteration sequence is chosen as $\{N_i^2\} = \{3, 4, 4, 1, 2, 2, 3, 3, 2, 4\}$. For $\gamma = 3$, iteration sequence is chosen as $\{N_i^3\} = \{6, 1, 9, 3, 5, 7, 5, 4, 1, 3\}$. For $\gamma = 4$ and $i = 1, 2, \dots, 10$, let $N_i^4 = 20$. Train the critic and the action networks under the learning rate 0.01 and set the neural network training error threshold as 10^{-6} . Under the iteration indices i and j_i , the trajectories of iterative value functions $V_{i,j_i}(x_k)$ for $x_k = x_0$ are

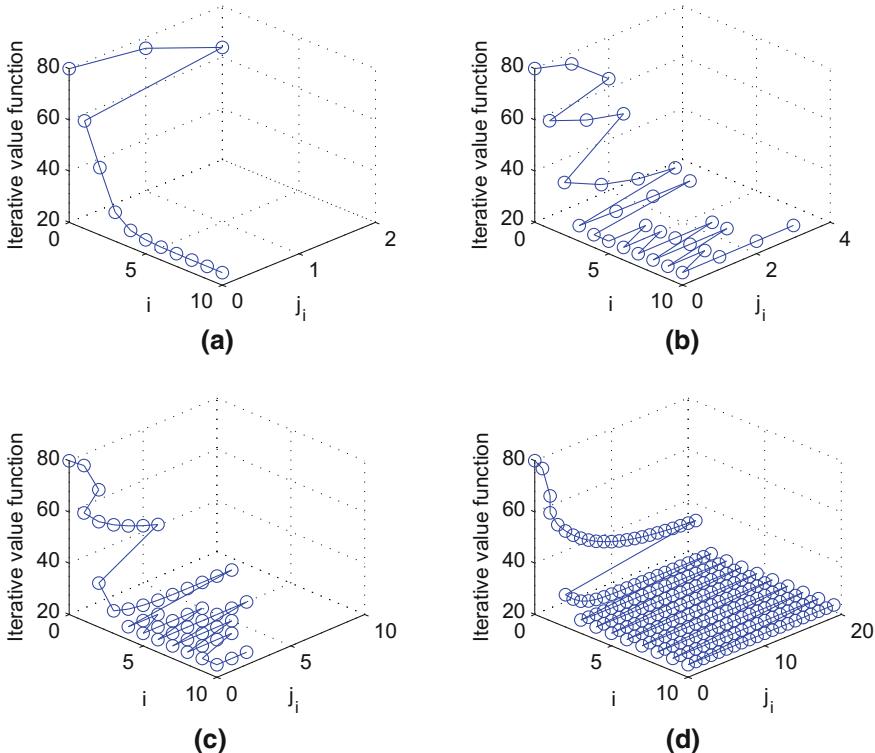


Fig. 5.1 Iterative value functions $V_{i,j_i}(x_k)$ for $i = 1, 2, \dots, 10$ and $x_k = x_0$. **a** $V_{i,j_i}(x_k)$ for $\{N_i^1\}$. **b** $V_{i,j_i}(x_k)$ for $\{N_i^2\}$. **c** $V_{i,j_i}(x_k)$ for $\{N_i^3\}$. **d** $V_{i,j_i}(x_k)$ for $\{N_i^4\}$

shown in Fig. 5.1. The curves of the iterative value functions $V_i(x_k)$ are shown in Fig. 5.2, where we use “In” to denote initial iteration and use “Lm” to denote limiting iteration.

For $N_i^1 = 1$, the generalized policy iteration algorithm is reduced to value iteration algorithm [5, 6, 16, 17]. From Figs. 5.1a and 5.2a, we can see that the iterative value function converges to the approximate optimum which verifies the effectiveness of the present algorithm. For $N_i^4 = 20$, we can see that for each $i = 1, \dots, 10$, the iterative value function $V_{i,j_i}(x_k)$ is convergent for j_i . In this case, the generalized policy iteration algorithm can be considered as a policy iteration algorithm [5, 8, 10] for each i , where the convergence can be verified. For arbitrary sequence $\{N_i\}$, such as $\{N_i^2\}$ and $\{N_i^3\}$, From Figs. 5.1b, c and 5.2b, c, the iterative value function can also approximate the optimum. Hence, value and policy iteration algorithms are special cases of the present generalized policy iteration algorithm and the convergence properties of the present algorithm can be verified. The stability property of system (5.2.36) under the iterative control law $v_i(x_k)$ is shown in Figs. 5.3 and 5.4, respectively.

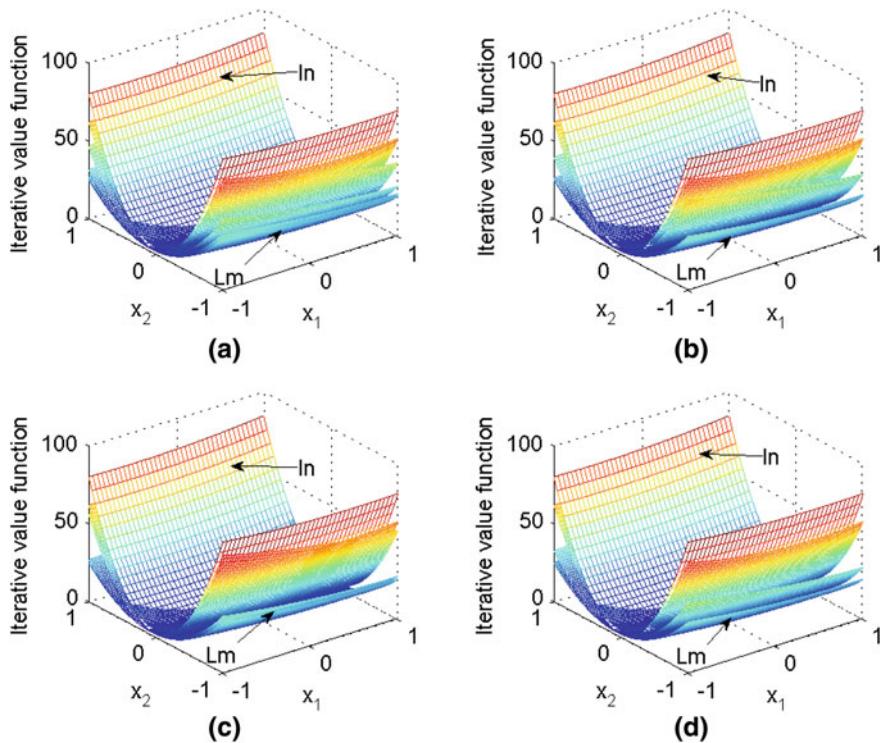


Fig. 5.2 Iterative value functions $V_i(x_k)$, for $i = 0, 1, \dots, 10$. **a** $V_i(x_k)$ for $\{N_i^1\}$. **b** $V_i(x_k)$ for $\{N_i^2\}$. **c** $V_i(x_k)$ for $\{N_i^3\}$. **d** $V_i(x_k)$ for $\{N_i^4\}$

From the above simulation results, we can see that for $i = 0, 1, \dots$, the iterative control laws $v_i(x_k)$ are admissible. For linear system (5.2.36), the optimal cost function is quadratic and in this case given by $J^*(x_k) = x_k^T P^* x_k$. According to the discrete algebraic Riccati equation (DARE) [7], we obtain

$$P^* = \begin{bmatrix} 26.61 & 1.81 \\ 1.81 & 1.90 \end{bmatrix}$$

and the effectiveness of the present algorithm can be verified for linear systems.

On the other hand, the structure of the neural networks is important for its approximation performance. To show the influence of the neural network structure, we change the structures of the critic and action networks to 2-4-1 and 2-4-1, respectively, and other parameters of the neural networks are kept unchanged. Define the two neural networks as group “NN2”. Choose the same $\{N_i^2\}$ for the j -iteration. Implement the present algorithm for $i = 10$ iterations. The iterative value functions by NN1 and NN2 are shown in Fig. 5.5a. We can see that if the number of hidden layer is reduced, the neural network approximate accuracy for value function may

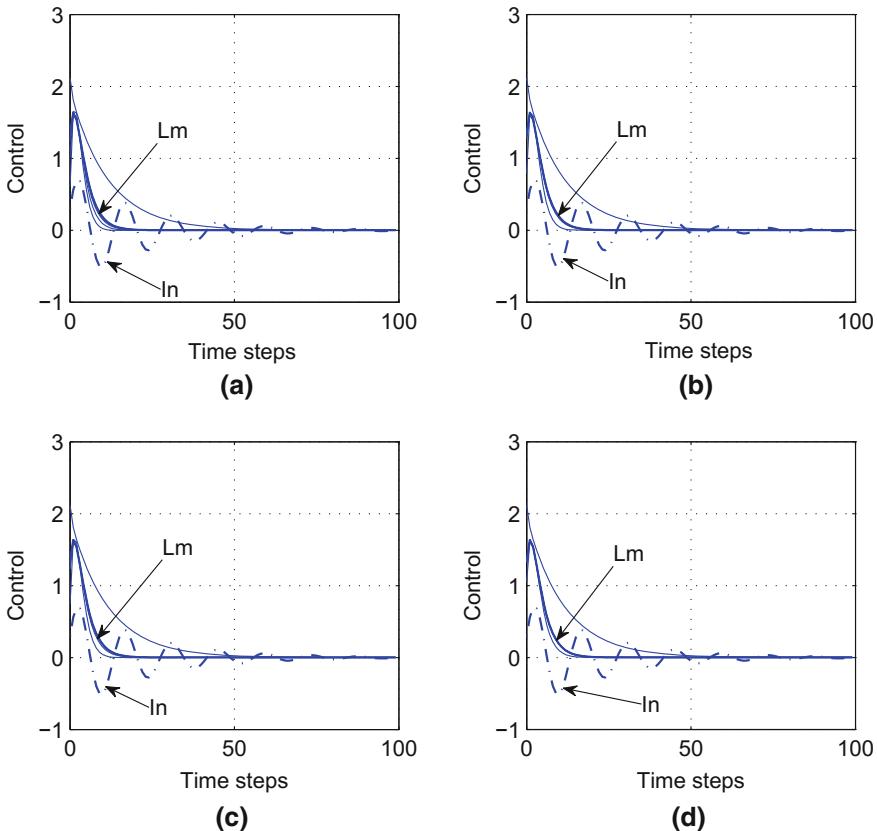


Fig. 5.3 Trajectories of iterative control law $v_i(x_k)$, $i = 1, 2, \dots, 10$. **a** $v_i(x_k)$ for $\{N_i^1\}$. **b** $v_i(x_k)$ for $\{N_i^2\}$. **c** $v_i(x_k)$ for $\{N_i^3\}$. **d** $v_i(x_k)$ for $\{N_i^4\}$

decrease. The plot of $V_i(x_k)$ is shown in Fig. 5.5b. The corresponding trajectories of states and control are shown in Fig. 5.5c, d, respectively. We can see that if the structure of neural networks is not chosen appropriately, the performance of the control system will be poor.

Example 5.2.2 We now examine the performance of the present algorithm in a torsional pendulum system [10, 13] with modifications. The dynamics of the pendulum is given by

$$\begin{cases} \frac{d\theta}{dt} = \omega, \\ J \frac{d\omega}{dt} = u - Mgl \sin \theta - f_d \frac{d\theta}{dt}, \end{cases}$$

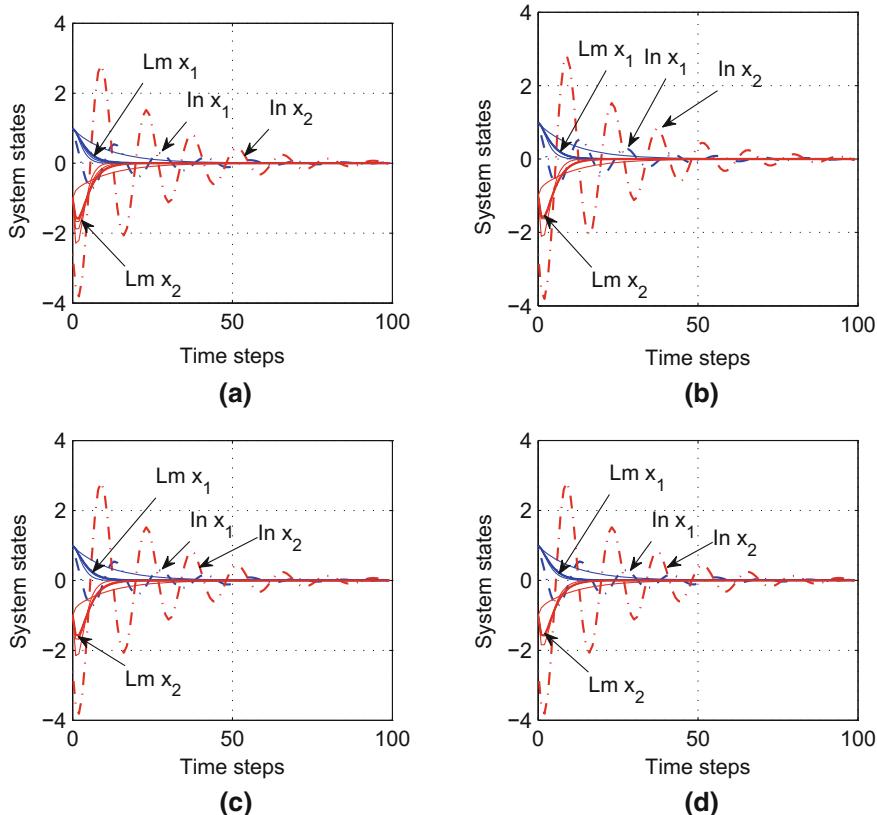


Fig. 5.4 Trajectories of system states. **a** State trajectories for $\{N_i^1\}$. **b** State trajectories for $\{N_i^2\}$. **c** State trajectory for $\{N_i^3\}$. **d** State trajectories for $\{N_i^4\}$

where $M = 1/3 \text{ kg}$ and $l = 2/3 \text{ m}$ are the mass and length of the pendulum bar, respectively. Let $J = 4/3 Ml^2$ and $f_d = 0.2$ be the rotary inertia and frictional factor, respectively. Let $x_1 = \theta$ and $x_2 = \omega$. Let $g = 9.8 \text{ m/s}^2$ be the gravity and the sampling time interval $\Delta t = 0.1 \text{ s}$. Then, the discretized system can be expressed by

$$\begin{bmatrix} x_{1(k+1)} \\ x_{2(k+1)} \end{bmatrix} = \begin{bmatrix} 0.1x_{2k} + x_{1k} \\ -0.49 \sin(x_{1k}) - 0.1f_d x_{2k} + x_{2k} \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u_k. \quad (5.2.37)$$

Let the initial state be $x_0 = [1, -1]^T$ and let the utility function be the same as the one in Example 5.2.1.

Neural networks are also used to implement the generalized policy iteration algorithm, where the structures of the critic network and the action network are the same as the ones in Example 5.2.1. We choose $p = 10000$ states in Ω to implement the generalized policy iteration algorithm. For nonlinear system (5.2.37), the initial

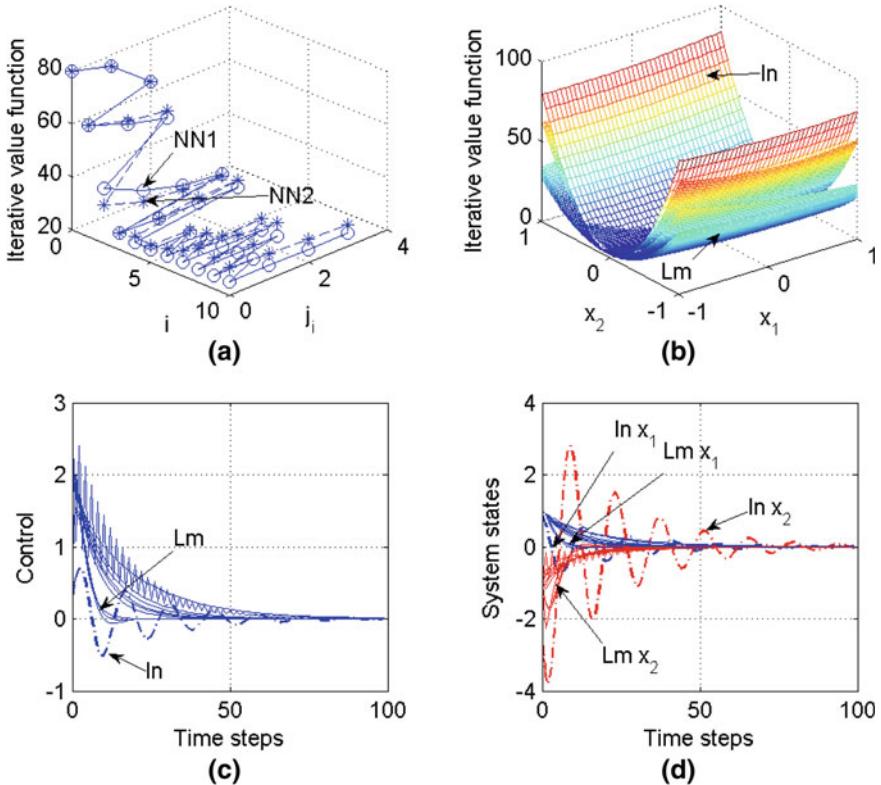


Fig. 5.5 Simulation results for $i = 0, 1, \dots, 10$ and $\{N_i^2\}$. **a** Value function at $x = x_0$ for NN1 and NN2. **b** $V_i(x_k)$ by NN2. **c** Iterative control law by NN2. **d** System states by NN2

admissible control law is difficult to obtain. Thus, we implement Algorithm 5.2.2 for policy improvement, and we obtain the initial value function $\bar{\Psi}^{s_0}(x_k) = x_k^T \bar{P}_0 x_k$, where

$$\bar{P}_0 = \begin{bmatrix} 145.31 & 8.43 \\ 8.43 & 28.42 \end{bmatrix}.$$

Let iteration index $i = 30$. To illustrate the effectiveness of the algorithm, we choose four different iteration sequences $\{N_i^\gamma\}$, $\gamma = 1, 2, 3, 4$. For $\gamma = 1$ and for $i = 0, 1, \dots, 30$, we let $N_i^1 = 1$. For $\gamma = 2$, let $N_i^2, i = 1, 2, \dots, 30$, be arbitrary nonnegative integer such that $0 < N_i^2 \leq 4$. For $\gamma = 3$, let $N_i^3, i = 1, 2, \dots, 30$, be arbitrary nonnegative integer such that $0 < N_i^3 \leq 10$. For $\gamma = 4$ and for $i = 0, 1, \dots, 30$, let $N_i^4 = 20$. Train the critic and the action networks under the learning rate 0.01 and set the threshold of neural network training error as 10^{-6} . Under the iteration indices i and j_i , the trajectories of iterative value functions $V_{i,j_i}(x_k)$ for $x = x_0$ are shown in Fig. 5.6. The curves of the iterative value functions $V_i(x_k)$ are shown in Fig. 5.7.

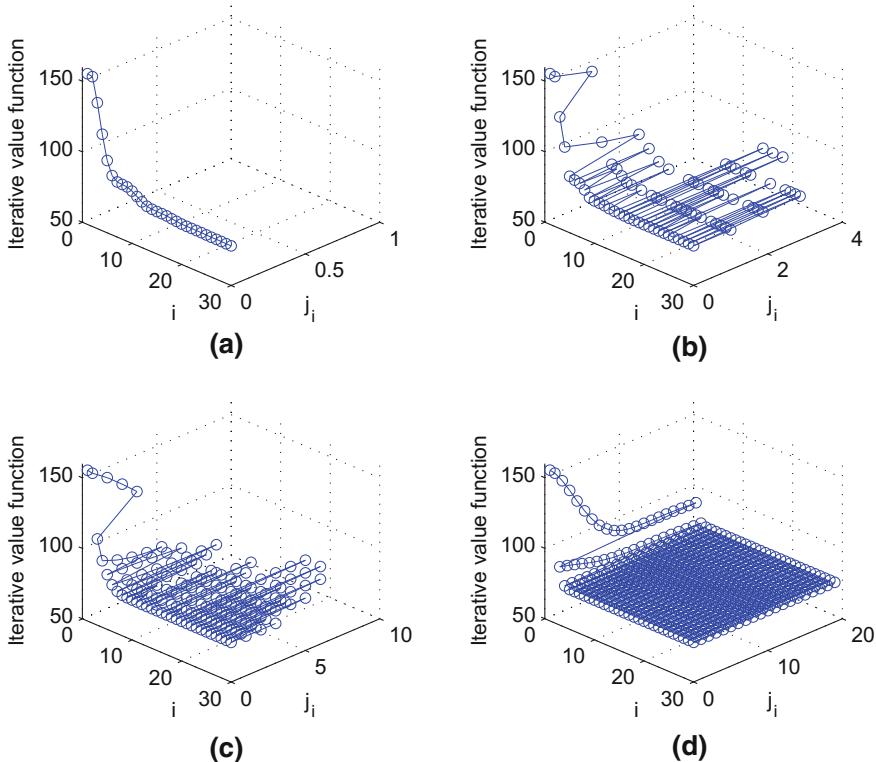


Fig. 5.6 Iterative value functions $V_{i,j_i}(x_k)$ for $i = 0, 1, \dots, 30$ and $x_k = x_0$. **a** $V_{i,j_i}(x_k)$ for $\{N_i^1\}$. **b** $V_{i,j_i}(x_k)$ for $\{N_i^2\}$ **c** $V_{i,j_i}(x_k)$ for $\{N_i^3\}$. **d** $V_{i,j_i}(x_k)$ for $\{N_i^4\}$

From Figs. 5.6 and 5.7, we can see that given an arbitrary nonnegative integer sequence $\{N_i\}$, $i = 0, 1, \dots$, the iterative value function $V_{i,j_i}(x_k)$ is monotonically nonincreasing and converges to the approximate optimum using the present generalized policy iteration algorithm. The convergence property of the generalized policy iteration algorithm for nonlinear systems can be verified. The convergence properties of value and policy iteration algorithms can also be verified by the present algorithm. The stability property of system (5.2.37) under the iterative control law $v_i(x_k)$ is shown in Figs. 5.8 and 5.9, respectively.

We can see that for $i = 0, 1, \dots$, the iterative control law $v_i(x_k)$ is admissible, and hence, the effectiveness of the present algorithm can be verified for nonlinear systems.

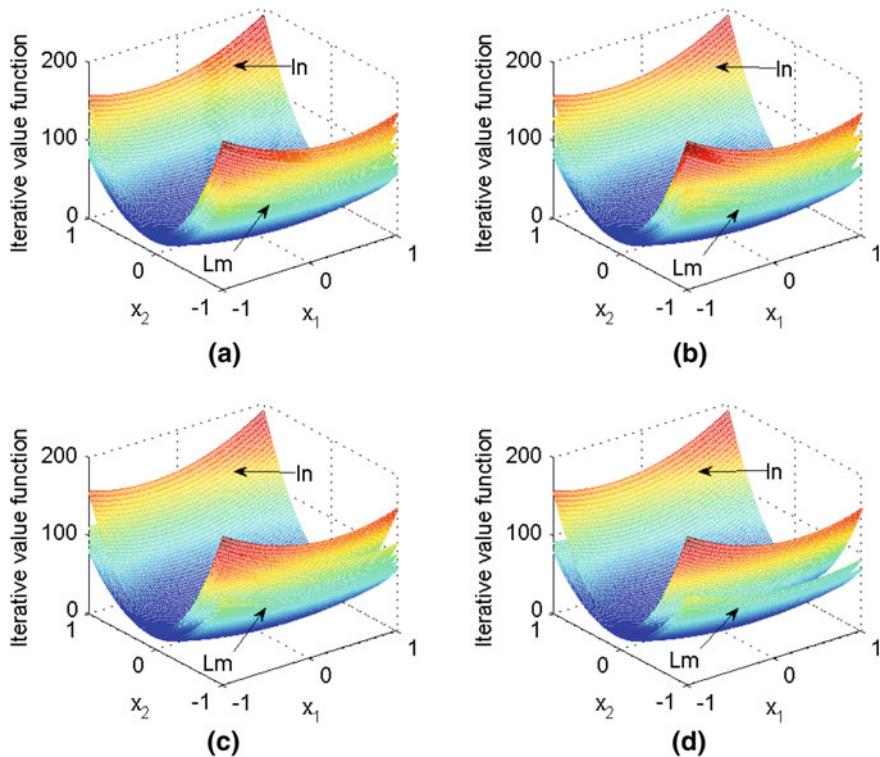


Fig. 5.7 Iterative value functions $V_i(x_k)$, $i = 0, 1, \dots, 30$. **a** $V_i(x_k)$ for $\{N_i^1\}$. **b** $V_i(x_k)$ for $\{N_i^2\}$. **c** $V_i(x_k)$ for $\{N_i^3\}$. **d** $V_i(x_k)$ for $\{N_i^4\}$

5.3 Discrete-Time GPI with General Initial Value Functions

In this section, a novel discrete-time GPI-based optimal control algorithm is developed for nonlinear systems. The developed GPI algorithm permits an arbitrary positive semidefinite function to initialize the algorithm, with iteration indices iterating between policy evaluation and policy improvement, respectively. The convergence, admissibility, and optimality properties of GPI algorithm for discrete-time nonlinear systems are analyzed [18].

5.3.1 Derivation and Properties of the GPI Algorithm

A. Derivation of the GPI Algorithm

The present generalized policy iteration algorithm contains two iteration procedures, which are named the i -iteration and the j -iteration, respectively. Both of the

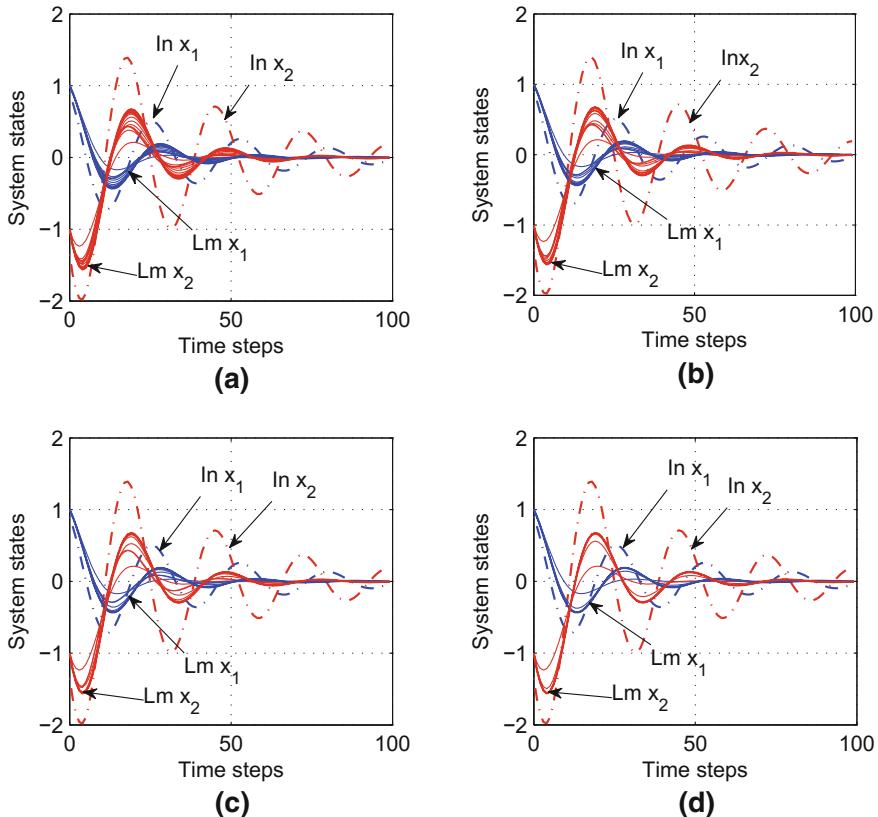


Fig. 5.8 Trajectories of system state. **a** State trajectories for $\{N_i^1\}$. **b** State trajectories for $\{N_i^2\}$. **c** State trajectories for $\{N_i^3\}$. **d** State trajectories for $\{N_i^4\}$

iteration indices increase from 0. The detailed generalized policy iteration algorithm is described as follows.

Let $\Psi(x_k)$ be a positive semidefinite function. Let

$$V_0(x_k) = \Psi(x_k) \quad (5.3.1)$$

be the initial value function. Let $\{N_1, N_2, \dots\}$ be a sequence, where $N_i > 0$, $i = 1, 2, \dots$, are positive integers. Then, for $i = 0, 1, \dots$, the iterative control laws and the iterative value functions are obtained as in (5.2.6)–(5.2.9). Details are referred to (5.2.6)–(5.2.9) since the only difference of the algorithm here is the initial condition.

Remark 5.3.1 From the generalized policy iteration algorithm (5.2.6)–(5.2.9) with initial condition (5.3.1), for $i = 1, 2, \dots$, if we let $N_i \equiv 1$, then the generalized policy iteration algorithm is reduced to a value iteration algorithm [2, 6, 9] (see Chaps. 2 and 3). For $i = 1, 2, \dots$, if we let $N_i \rightarrow \infty$, then the generalized policy iteration

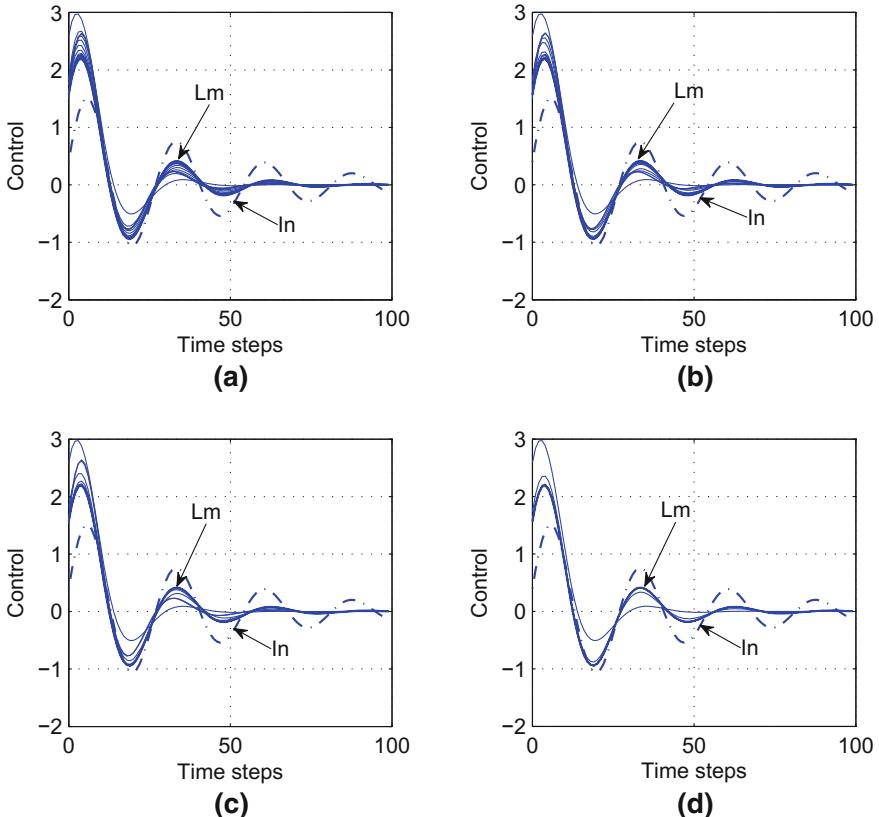


Fig. 5.9 Trajectories of iterative control law $v_i(x_k)$. **a** $v_i(x_k)$ for $\{N_i^1\}$. **b** $v_i(x_k)$ for $\{N_i^2\}$. **c** $v_i(x_k)$ for $\{N_i^3\}$. **d** $v_i(x_k)$ for $\{N_i^4\}$

algorithm becomes a policy iteration algorithm [10] (see Chap. 4). Hence, the value and policy iteration algorithms are special cases of the present generalized policy iteration algorithm. On the other hand, we can see that the generalized policy iteration algorithm (5.2.6)–(5.2.9) is inherently different from policy and value iteration algorithms. Analysis results for (5.2.5)–(5.2.9) have been established in Sect. 5.2. In this section, further analysis results will be established for (5.2.6)–(5.2.9) with initial condition (5.3.1) using the approach due to Rantzer et al. [9, 12].

B. Properties of the GPI Algorithm

In this section, the properties of the GPI algorithm are analyzed. First, for $i = 1, 2, \dots$, the convergence property of the iterative value function in j -iteration (local convergence property) is analyzed. The local convergence criterion is obtained. Second, the convergence property of the iterative value function for $i \rightarrow \infty$ (global convergence property) is developed and the corresponding convergence criterion is obtained. The admissibility and optimality analysis are also presented in this section.

Theorem 5.3.1 For $i = 1, 2, \dots$, let $V_{i,j_i}(x_k)$ and $v_i(x_k)$ be obtained by (5.2.6)–(5.2.9) with initial condition (5.3.1). Let $0 < \gamma_i < \infty$ and $1 \leq \sigma_i < \infty$ be constants such that

$$\gamma_i U(x_k, u_k) \geq V_i(x_k), \quad (5.3.2)$$

and

$$V_{i,0}(x_k) \leq \sigma_i V_{i-1}(x_k). \quad (5.3.3)$$

Then, for $i = 1, 2, \dots$ and $j_i = 0, 1, \dots, N_i$, we have

$$V_{i,j_i}(x_k) \leq \left[1 + \sum_{\rho=1}^{j_i} \frac{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1)}{(1 + \gamma_i)^\rho} \right] V_{i,0}(x_k), \quad (5.3.4)$$

where we define $\sum_{\rho=1}^{j_i} (\cdot) = 0$ for $j_i < 1$.

Proof The theorem can be proven by mathematical induction. Obviously, inequality (5.3.4) holds for $j_i = 0$. For $j_i = 1$, we have

$$\begin{aligned} V_{i,1}(x_k) &= U(x_k, v_{i-1}(x_k)) + V_{i,0}(x_{k+1}) \\ &\leq U(x_k, v_{i-1}(x_k)) + \sigma_i V_{i-1}(x_{k+1}) \\ &\leq \left(1 + \gamma_i \frac{\sigma_i - 1}{1 + \gamma_i} \right) U(x_k, v_{i-1}(x_k)) + \left(\sigma_i - \frac{\sigma_i - 1}{1 + \gamma_i} \right) V_{i-1}(x_{k+1}) \\ &\leq \left(1 + \gamma_i \frac{\sigma_i - 1}{1 + \gamma_i} \right) V_{i,0}(x_k). \end{aligned}$$

Thus, (5.3.4) holds for $j_i = 1$. Assume that the conclusion holds for $j_i = l - 1$, $l = 1, 2, \dots, N_i$. Then, for $j_i = l$, we have

$$\begin{aligned} V_{i,l}(x_k) &= U(x_k, v_{i-1}(x_k)) + V_{i,l-1}(x_{k+1}) \\ &\leq U(x_k, v_{i-1}(x_k)) + \left[1 + \sum_{\rho=1}^{l-1} \frac{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1)}{(1 + \gamma_i)^\rho} \right] V_{i,0}(x_k) \\ &\leq \left[1 + \frac{\gamma_i}{1 + \gamma_i} \left(\sigma_i - 1 + \sum_{\rho=1}^{l-1} \frac{\gamma_i^\rho \sigma_i^\rho (\sigma_i - 1)}{(1 + \gamma_i)^\rho} \right) \right] U(x_k, v_i(x_k)) \\ &\quad + \left[\sigma_i + \sum_{\rho=1}^{l-1} \frac{\gamma_i^\rho \sigma_i^\rho (\sigma_i - 1)}{(1 + \gamma_i)^\rho} - \left(\frac{\sigma_i - 1}{1 + \gamma_i} + \sum_{\rho=1}^{l-1} \frac{\gamma_i^\rho \sigma_i^\rho (\sigma_i - 1)}{(1 + \gamma_i)^{\rho+1}} \right) \right] V_{i-1}(x_{k+1}) \\ &= \left[1 + \sum_{\rho=1}^l \frac{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1)}{(1 + \gamma_i)^\rho} \right] V_{i,0}(x_k). \end{aligned}$$

Hence, (5.3.4) holds for $j = l$. The mathematical induction is complete. This completes the proof of the theorem.

Theorem 5.3.2 (Local convergence criterion) *For $i = 0, 1, \dots$ and $j_i = 1, 2, \dots, N_i$, let $V_{i,j_i}(x_k)$ and $v_i(x_k)$ be obtained by (5.2.6)–(5.2.9) with initial condition (5.3.1). Let $0 < \gamma_i < \infty$ and $1 \leq \sigma_i < \infty$ be constants that satisfy (5.3.2) and (5.3.3), respectively. If for $i = 1, 2, \dots$, the constant σ_i satisfies*

$$\sigma_i < \frac{1 + \gamma_i}{\gamma_i}, \quad (5.3.5)$$

then, the iterative value function $V_{i,j_i}(x_k)$ is convergent as $j_i \rightarrow \infty$, i.e.,

$$\lim_{j_i \rightarrow \infty} V_{i,j_i}(x_k) \leq \frac{1}{1 + \gamma_i - \gamma_i \sigma_i} V_{i,0}(x_k). \quad (5.3.6)$$

Proof According to (5.3.4), we can see the sequence $\{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1) / (1 + \gamma_i)^\rho\}$ is geometrical for $j_i = 1, 2, \dots$. Then, (5.3.4) can be written as

$$V_{i,j_i}(x_k) \leq \left[1 + \frac{\frac{\gamma_i(\sigma_i - 1)}{\gamma_i + 1} \left(1 - \left(\frac{\gamma_i \sigma_i}{\gamma_i + 1} \right)^j \right)}{1 - \frac{\gamma_i \sigma_i}{\gamma_i + 1}} \right] V_{i,0}(x_k).$$

For

$$1 \leq \sigma_i < \frac{\gamma_i + 1}{\gamma_i},$$

letting $j_i \rightarrow \infty$, we can obtain (5.3.6). The proof is complete.

For optimal control problems, the present control scheme must not only stabilize the control systems, but also guarantee the cost function to be finite, i.e., the control law must be admissible [2]. Next, the admissibility property of the iterative control law $v_i(x_k)$ will be analyzed.

Theorem 5.3.3 (Admissibility property) *For $i = 1, 2, \dots$ and $j_i = 0, 1, \dots, N_i$, let $V_{i,j_i}(x_k)$ and $v_i(x_k)$ be obtained by (5.2.6)–(5.2.9) with initial condition (5.3.1). Let $0 < \gamma_i < \infty$ and $1 \leq \sigma_i < \infty$ be constants that satisfy (5.3.2) and (5.3.3), respectively. If for $i = 1, 2, \dots$, the constant σ_i satisfies (5.3.5), and the iterative value function $V_i(x_k)$ is finite, then $v_i(x_k)$ is admissible, i.e., $v_i(x_k) \in \mathcal{A}(\Omega)$.*

Proof Let $\bar{j}_i = 0, 1, \dots$. We construct a value function $\mathcal{V}_{i,\bar{j}_i}(x_k)$ as

$$\mathcal{V}_{i,\bar{j}_i}(x_k) = U(x_k, v_{i-1}(x_k)) + \mathcal{V}_{i,\bar{j}_i-1}(F(x_k, v_{i-1}(x_k))), \quad (5.3.7)$$

where

$$\mathcal{V}_{i,0}(x_k) = V_{i-1}(x_k). \quad (5.3.8)$$

From (5.3.7) and (5.3.8), we have $\mathcal{V}_{i,\bar{j}_i}(x_k) = V_{i,\bar{j}_i}(x_k)$ for $\bar{j}_i \leq N_i$. For $\bar{j}_i = 0, 1, \dots$, according to (5.3.7), we can obtain

$$\mathcal{V}_{i,\bar{j}_i}(x_k) = \sum_{l=0}^{\bar{j}_i-1} U(x_{k+l}, v_{i-1}(x_{k+l})) + V_{i-1}(x_{k+\bar{j}_i}).$$

Let σ_i satisfy (5.3.5). According to Theorem 5.3.2, letting $\bar{j}_i \rightarrow \infty$, we can obtain

$$\begin{aligned} \lim_{\bar{j}_i \rightarrow \infty} \mathcal{V}_{i,\bar{j}_i}(x_k) &= \lim_{\bar{j}_i \rightarrow \infty} \sum_{l=0}^{\bar{j}_i-1} U(x_{k+l}, v_{i-1}(x_{k+l})) + \lim_{\bar{j}_i \rightarrow \infty} V_{i-1}(x_{k+\bar{j}_i}) \\ &\leq \frac{\sigma_i}{1 + \gamma_i - \gamma_i \sigma_i} V_{i-1}(x_k). \end{aligned}$$

Since $V_{i-1}(x_k)$ is finite for $i = 1, 2, \dots$, $\sum_{l=0}^{\infty} U(x_{k+l}, v_i(x_{k+l}))$ is also finite. Define $\mathcal{V}_{i,\infty}(x_k) = \lim_{\bar{j}_i \rightarrow \infty} \mathcal{V}_{i,\bar{j}_i}(x_k)$. If σ_i satisfies (5.3.5), letting $\bar{j}_i \rightarrow \infty$, the iterative control law $v_i(x_k)$ satisfies the following generalized Bellman equation

$$\mathcal{V}_{i,\infty}(x_k) = U(x_k, v_{i-1}(x_k)) + \mathcal{V}_{i,\infty}(F(x_k, v_{i-1}(x_k))). \quad (5.3.9)$$

According to Lemma 5.2.1, we can derive that $\mathcal{V}_{i,\infty}(x_k)$ is a positive-definite function. From (5.3.9), we get

$$\mathcal{V}_{i,\infty}(F(x_k, v_{i-1}(x_k))) - \mathcal{V}_{i,\infty}(x_k) \leq 0,$$

which means that $\mathcal{V}_{i,\infty}(x_k)$ is a Lyapunov function. Thus, $x_k \rightarrow 0$ as $k \rightarrow \infty$, which means that the iterative control law $v_i(x_k) \in \mathcal{Q}_u$. The proof is complete.

Theorem 5.3.4 For $i = 0, 1, \dots$, let $V_{i,j_i}(x_k)$ and $v_i(x_k)$ be obtained by (5.2.6)–(5.2.9) with initial condition (5.3.1). Let $0 < \gamma < \infty$ and $1 \leq \sigma < \infty$ be constants such that $J^*(F(x_k, u_k)) \leq \gamma U(x_k, u_k)$ and $V_0(x_k) \leq \sigma J^*(x_k)$. For $i = 0, 1, \dots$, if σ_i satisfies (5.3.5), then the iterative value function $V_{i,j_i}(x_k)$ satisfies

$$V_{i,j_i}(x_k) \leq \left(1 + \sum_{\rho=1}^{j_i} \frac{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1)}{(1 + \gamma_i)^\rho} \right)$$

$$\begin{aligned} & \times \left[1 + \sum_{l=1}^{i-1} \frac{\gamma^{i-l} \gamma_l (\sigma_l - 1)}{(1 + \gamma)^{i-l} \prod_{\eta=l}^{i-1} (1 - \gamma_\eta (\sigma_\eta - 1))} \right. \\ & \left. + \frac{\gamma^i (\sigma - 1)}{(1 + \gamma)^i \prod_{\eta=1}^{i-1} (1 - \gamma_\eta (\sigma_\eta - 1))} \right] J^*(x_k), \end{aligned} \quad (5.3.10)$$

where we define $\sum_k^l (\cdot) = 0$ and $\prod_k^l (\cdot) = 1$, for $k > l$.

Proof The statement can be proven by mathematical induction. First, the conclusion is obviously true for $i = 0$. Let $i = 1$. For $j_i = 1, 2, \dots$, we have from (5.3.4),

$$\begin{aligned} V_{1,j_1}(x_k) & \leq \left(1 + \sum_{\rho=1}^{j_1} \frac{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1)}{(1 + \gamma_i)^\rho} \right) V_{1,0}(x_k) \\ & \leq \left(1 + \sum_{\rho=1}^{j_1} \frac{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1)}{(1 + \gamma_i)^\rho} \right) \min_{u_k} \{ U(x_k, u_k) + \sigma J^*(x_{k+1}) \} \\ & \leq \left(1 + \sum_{\rho=1}^{j_1} \frac{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1)}{(1 + \gamma_i)^\rho} \right) \min_{u_k} \left\{ \left(1 + \gamma \frac{\sigma - 1}{1 + \gamma} \right) U(x_k, u_k) \right. \\ & \quad \left. + \left(\sigma - \frac{\sigma - 1}{1 + \gamma} \right) J^*(x_{k+1}) \right\} \\ & \leq \left(1 + \sum_{\rho=1}^{j_1} \frac{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1)}{(1 + \gamma_i)^\rho} \right) \left(1 + \frac{\gamma(\sigma - 1)}{1 + \gamma} \right) J^*(x_k), \end{aligned}$$

which proves inequality (5.3.10) for $i = 1$. Assume that the conclusion holds for $i = \vartheta - 1$, $\vartheta = 1, 2, \dots$. As σ_i satisfies (5.3.5), we can get

$$\begin{aligned} V_{\vartheta-1,j_{\vartheta-1}}(x_k) & \leq \frac{1}{1 - \gamma_{\vartheta-1} (\sigma_{\vartheta-1} - 1)} \\ & \times \left[1 + \sum_{l=1}^{\vartheta-2} \frac{\gamma^{i-\vartheta-1} \gamma_l (\sigma_l - 1)}{(1 + \gamma)^{i-\vartheta-1} \prod_{\eta=l}^{\vartheta-2} (1 - \gamma_\eta (\sigma_\eta - 1))} \right. \\ & \left. + \frac{\gamma^{\vartheta-1} (\sigma - 1)}{(1 + \gamma)^{\vartheta-1} \prod_{\eta=1}^{\vartheta-2} (1 - \gamma_\eta (\sigma_\eta - 1))} \right] J^*(x_k). \end{aligned}$$

For $i = \vartheta$, we can get

$$\begin{aligned}
 V_{\vartheta, j_\vartheta}(x_k) &\leq \left(1 + \sum_{\rho=1}^{j_\vartheta} \frac{\gamma_\vartheta^\rho \sigma_\vartheta^{\rho-1} (\sigma_\vartheta - 1)}{(1 + \gamma_\vartheta)^\rho} \right) V_{\vartheta, 0}(x_{k+1}) \\
 &\leq \left(1 + \sum_{\rho=1}^{j_\vartheta} \frac{\gamma_\vartheta^\rho \sigma_\vartheta^{\rho-1} (\sigma_\vartheta - 1)}{(1 + \gamma_\vartheta)^\rho} \right) \min_{u_k} \{ U(x_k, u_k) + V_{\vartheta-1}(x_{k+1}) \} \\
 &\leq \left(1 + \sum_{\rho=1}^{j_\vartheta} \frac{\gamma_\vartheta^\rho \sigma_\vartheta^{\rho-1} (\sigma_\vartheta - 1)}{(1 + \gamma_\vartheta)^\rho} \right) \min_{u_k} \left\{ U(x_k, u_k) + \frac{1}{1 - \gamma_{\vartheta-1}(\sigma_{\vartheta-1} - 1)} \right. \\
 &\quad \times \left[1 + \sum_{l=1}^{\vartheta-2} \frac{\gamma^{\vartheta-l-1} \gamma_l (\sigma_l - 1)}{(1 + \gamma)^{\vartheta-l-1} \prod_{\eta=l}^{\vartheta-2} (1 - \gamma_\eta (\sigma_\eta - 1))} \right. \\
 &\quad \left. + \frac{\gamma^{\vartheta-1} (\sigma - 1)}{(1 + \gamma)^{\vartheta-1} \prod_{\eta=l}^{\vartheta-2} (1 - \gamma_\eta (\sigma_\eta - 1))} \right] J^*(x_{k+1}) \Big\} \\
 &\leq \left(1 + \sum_{\rho=1}^{j_\vartheta} \frac{\gamma_\vartheta^\rho \sigma_\vartheta^{\rho-1} (\sigma_\vartheta - 1)}{(1 + \gamma_\vartheta)^\rho} \right) \min_{u_k} \left\{ \left[\sum_{l=1}^{\vartheta-1} \frac{\gamma^{\vartheta-l} \gamma_l (\sigma_l - 1)}{(1 + \gamma)^{\vartheta-l} \prod_{\eta=l}^{\vartheta-1} (1 - \gamma_\eta (\sigma_\eta - 1))} \right. \right. \\
 &\quad \left. + \frac{\gamma^\vartheta (\sigma - 1)}{(1 + \gamma)^\vartheta \prod_{\eta=l}^{\vartheta-1} (1 - \gamma_\eta (\sigma_\eta - 1))} + 1 \right] U(x_k, u_k) \\
 &\quad + \left[\frac{1}{1 - \gamma_{\vartheta-1}(\sigma_{\vartheta-1} - 1)} \left(1 + \sum_{l=1}^{\vartheta-2} \frac{\gamma^{\vartheta-l-1} \gamma_l (\sigma_l - 1)}{(1 + \gamma)^{\vartheta-l-1} \prod_{\eta=l}^{\vartheta-2} (1 - \gamma_\eta (\sigma_\eta - 1))} \right. \right. \\
 &\quad \left. + \frac{\gamma^{\vartheta-1} (\sigma - 1)}{(1 + \gamma)^{\vartheta-1} \prod_{\eta=l}^{\vartheta-2} (1 - \gamma_\eta (\sigma_\eta - 1))} \right) \\
 &\quad - \left(\sum_{l=1}^{\vartheta-1} \frac{\gamma^{\vartheta-l-1} \gamma_l (\sigma_l - 1)}{(1 + \gamma)^{\vartheta-l} \prod_{\eta=l}^{\vartheta-1} (1 - \gamma_\eta (\sigma_\eta - 1))} \right)
 \end{aligned}$$

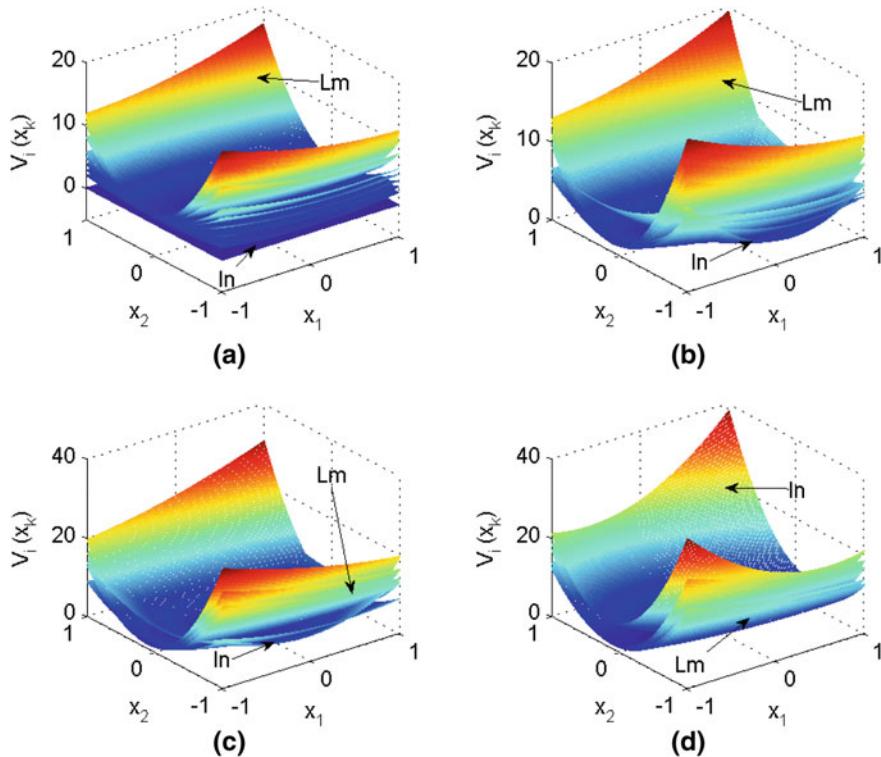


Fig. 5.10 Iterative value function $V_i(x_k)$ for different $\Psi(x_k)$'s. **a** $V_i(x_k)$ for $\Psi^1(x_k)$. **b** $V_i(x_k)$ for $\Psi^2(x_k)$. **c** $V_i(x_k)$ for $\Psi^3(x_k)$. **d** $V_i(x_k)$ for $\Psi^4(x_k)$

$$\begin{aligned}
& + \frac{\gamma^{\vartheta-1}(\sigma-1)}{(1+\gamma)^{\vartheta} \prod_{\eta=l}^{\vartheta-1} (1-\gamma_{\eta}(\sigma_{\eta}-1))} \Bigg] J^*(x_{k+1}) \Bigg\} \\
& = \left(1 + \sum_{\rho=1}^{j_{\vartheta}} \frac{\gamma_{\vartheta}^{\rho} \sigma_{\vartheta}^{\rho-1} (\sigma_{\vartheta}-1)}{(1+\gamma_{\vartheta})^{\rho}} \right) \left[1 + \sum_{l=1}^{\vartheta-1} \frac{\gamma^{i-\vartheta} \gamma_l (\sigma_l-1)}{(1+\gamma)^{i-\vartheta} \prod_{\eta=l}^{\vartheta-1} (1-\gamma_{\eta}(\sigma_{\eta}-1))} \right. \\
& \quad \left. + \frac{\gamma^{\vartheta}(\sigma-1)}{(1+\gamma)^{\vartheta} \prod_{\eta=l}^{\vartheta-1} (1-\gamma_{\eta}(\sigma_{\eta}-1))} \right] J^*(x_k).
\end{aligned}$$

The mathematical induction is complete for inequality (5.3.10). This completes the proof of the theorem.

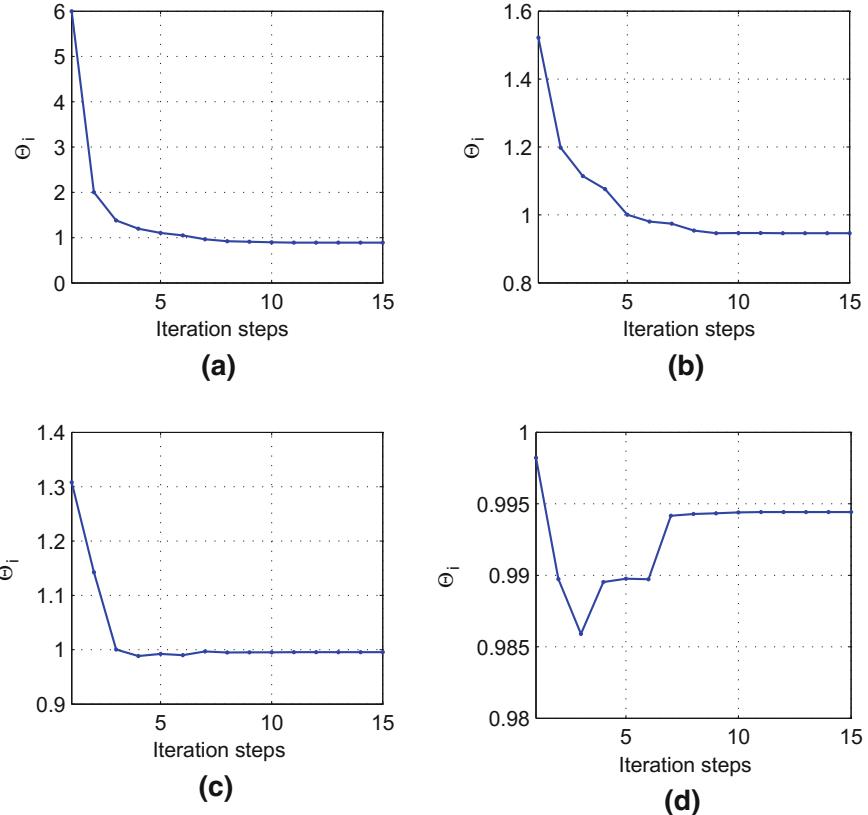


Fig. 5.11 The trajectories of Θ_i for different $\Psi(x_k)$'s. **a** Θ_i for $\Psi^1(x_k)$. **b** Θ_i for $\Psi^2(x_k)$. **c** Θ_i for $\Psi^3(x_k)$. **d** Θ_i for $\Psi^4(x_k)$

Theorem 5.3.5 (Global convergence criterion) *For $i = 0, 1, \dots$, let $V_{i,j_i}(x_k)$ and $v_i(x_k)$ be obtained by (5.2.6)–(5.2.9) with initial condition (5.3.1). If for $i = 0, 1, \dots$, σ_i satisfies*

$$\sigma_i < q_i \frac{1}{\gamma_i(1 + \gamma)} + 1, \quad (5.3.11)$$

where $0 < q_i < 1$ is a constant, then for $j_i = 0, 1, \dots, N_i$, the iterative value function $V_{i,j_i}(x_k)$ is convergent as $i \rightarrow \infty$.

Proof Define $\Omega_{\text{IN}} = \{i : i = 1, 2, \dots\}$ as an iteration index set. For $i \in \Omega_{\text{IN}}$, if we let

$$q = \max_{i \in \Omega_{\text{IN}}} \{q_i\},$$

then $0 < q < 1$. According to (5.3.10), we can get

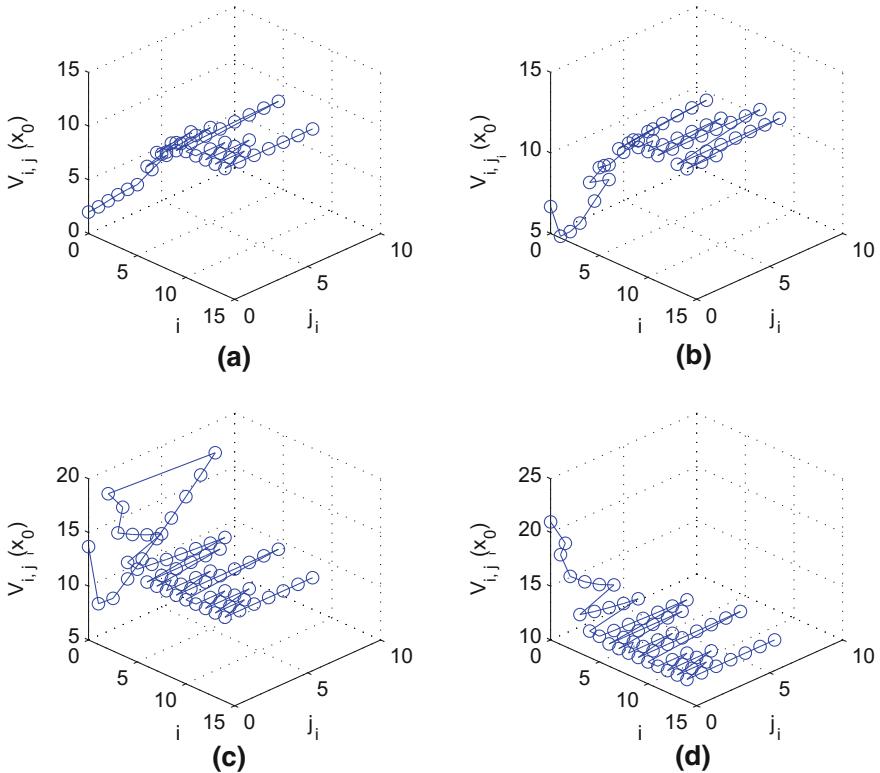


Fig. 5.12 Iterative value function $V_{i,j_i}(x_k)$ for $x_k = x_0$ and different $\Psi(x_k)$'s. **a** $V_{i,j_i}(x_k)$ for $\Psi^1(x_k)$. **b** $V_{i,j_i}(x_k)$ for $\Psi^2(x_k)$. **c** $V_{i,j_i}(x_k)$ for $\Psi^3(x_k)$. **d** $V_{i,j_i}(x_k)$ for $\Psi^4(x_k)$

$$\lim_{i \rightarrow \infty} \sum_{l=1}^{i-1} \frac{\gamma^{i-l} \gamma_l (\sigma_l - 1)}{(1 + \gamma)^{i-l} \prod_{\eta=l}^{i-1} (1 - \gamma_\eta (\sigma_\eta - 1))} < \lim_{i \rightarrow \infty} \sum_{l=1}^{i-1} \frac{\gamma^{i-l} \frac{q}{1 + \gamma}}{(1 + \gamma)^{i-l} \left(1 - \frac{q}{1 + \gamma}\right)^{i-l}} = \frac{q \gamma}{(1 - q)(1 + \gamma)}. \quad (5.3.12)$$

If σ_i satisfies (5.3.11), then by the necessity of convergent series, we can get

$$\lim_{i \rightarrow \infty} \frac{\gamma^i (\sigma - 1)}{(1 + \gamma)^i \prod_{\eta=1}^{i-1} (1 - \gamma_\eta (\sigma_\eta - 1))} = 0. \quad (5.3.13)$$

Following the fact that

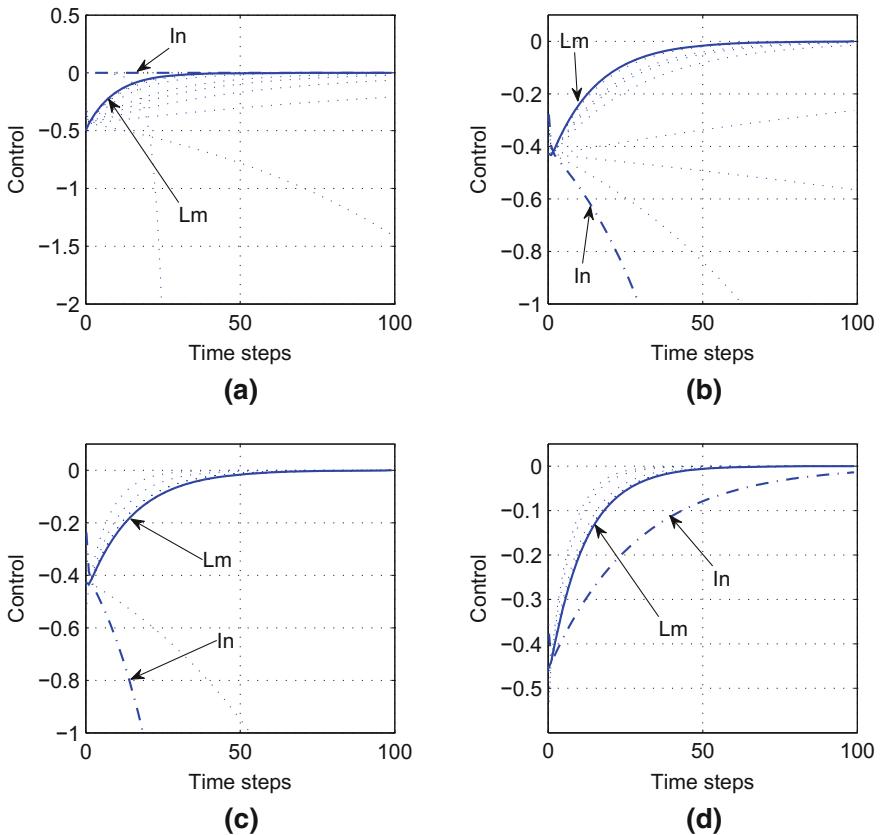


Fig. 5.13 Trajectories of iterative control law $v_i(x_k)$ for different $\Psi(x_k)$'s. **a** $v_i(x_k)$ for $\Psi^1(x_k)$. **b** $v_i(x_k)$ for $\Psi^2(x_k)$. **c** $v_i(x_k)$ for $\Psi^3(x_k)$. **d** $v_i(x_k)$ for $\Psi^4(x_k)$

$$q_i \frac{1}{\gamma_i(1+\gamma)} + 1 < \frac{1+\gamma_i}{\gamma_i}, \quad (5.3.14)$$

we can obtain

$$\sum_{\rho=1}^{j_i} \frac{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1)}{(1 + \gamma_i)^\rho} < \lim_{j_i \rightarrow \infty} \sum_{\rho=1}^{j_i} \frac{\gamma_i^\rho \sigma_i^{\rho-1} (\sigma_i - 1)}{(1 + \gamma_i)^\rho} < \frac{q}{\gamma + 1 - q}. \quad (5.3.15)$$

According to (5.3.12)–(5.3.15), we can obtain

$$\lim_{i \rightarrow \infty} V_{i, j_i}(x_k) < \left(1 + \frac{q}{\gamma + 1 - q}\right) \left(1 + \frac{q\gamma}{(1 - q)(1 + \gamma)}\right) J^*(x_k).$$

The proof is complete.

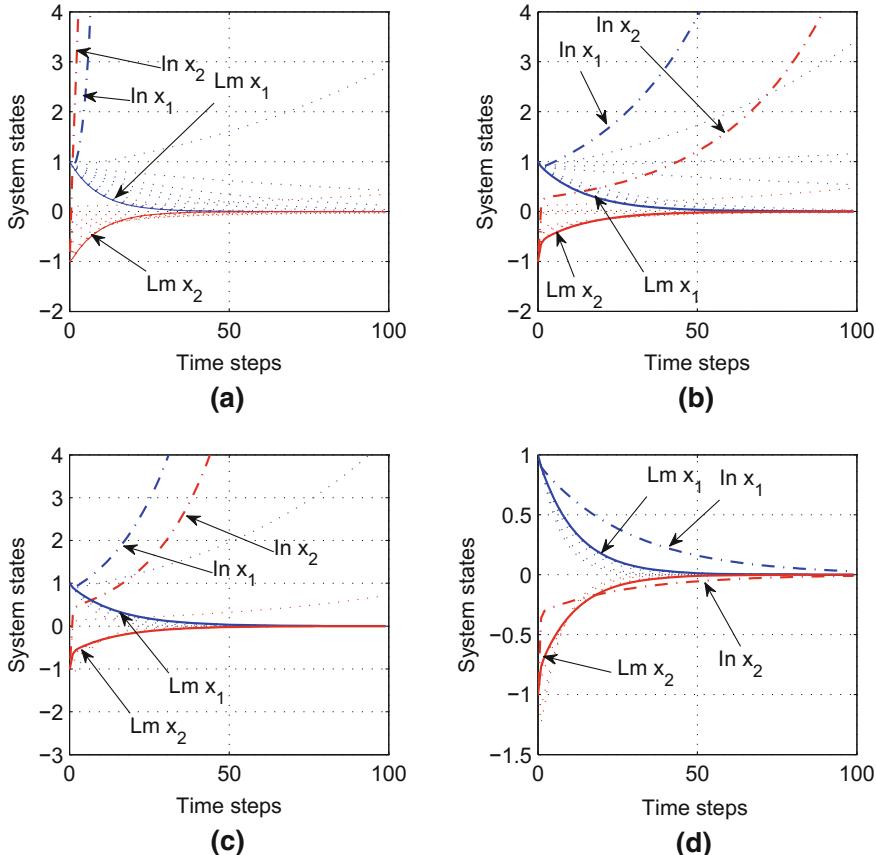


Fig. 5.14 Trajectories of system states for different $\Psi(x_k)$'s. **a** State trajectories for $\Psi^1(x_k)$. **b** State trajectories for $\Psi^2(x_k)$. **c** State trajectories for $\Psi^3(x_k)$. **d** State trajectories for $\Psi^4(x_k)$

5.3.2 Relaxations of the Convergence Criterion and Summary of the GPI Algorithm

From Theorem 5.3.5, we can see that the convergence criterion (5.3.11) is independent of the parameter σ . For $i = 1, 2, \dots, \sigma_i$ and γ_i can be obtained by (5.3.2) and (5.3.3), respectively. If the parameter γ can be estimated, then the convergence criterion (5.3.11) can be implemented. As the optimal cost function $J^*(x_k)$ is unknown, the parameter γ cannot be obtained directly. In this section, relaxations of the convergence criterion are discussed. First, we define a new set Ω_γ as

$$\Omega_\gamma = \{\gamma : \gamma U(x_k, u_k) \geq J^*(F(x_k, u_k))\}.$$

Then, we can obtain the following statements.

Lemma 5.3.1 *Let $P_i(x_k)$ be the iterative value function that satisfies*

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

where $P_0(x_k)$ is an arbitrary positive semidefinite function and $\mu(x_k)$ is an arbitrary admissible control law. Let $P_\infty(x_k) = \lim_{i \rightarrow \infty} P_i(x_k)$. If there exists a constant $\tilde{\gamma}$ such that

$$\tilde{\gamma} U(x_k, u_k) \geq P_\infty(F(x_k, u_k)), \quad (5.3.16)$$

then $\tilde{\gamma} \in \Omega_\gamma$.

Proof As $\mu(x_k)$ is an admissible control law, according to Theorem 5.2.3, we have $P_\infty(x_k) \geq V_\infty(x_k)$. If $\tilde{\gamma}$ satisfies (5.3.16), then we can get

$$\tilde{\gamma} U(x_k, u_k) \geq P_\infty(F(x_k, u_k)) \geq V_\infty(F(x_k, u_k)) = J^*(F(x_k, u_k)).$$

The proof is complete.

Theorem 5.3.6 *Let $\Phi_0(x_k)$ be a positive-definite function and let $\tilde{\gamma}$ be a constant such that*

$$\tilde{\gamma} U(x_k, u_k) \geq \Phi_0(F(x_k, u_k)). \quad (5.3.17)$$

For $l = 0, 1, \dots$, let $\Phi_l(x_k)$ be the iterative value function such that

$$\Phi_{l+1}(x_k) = \min_{u_k} \{U(x_k, u_k) + \Phi_l(x_{k+1})\}.$$

If $\Phi_1(x_k) \leq \Phi_0(x_k)$, then $\tilde{\gamma} \in \Omega_\gamma$.

Proof The statement can be proven in two steps. First, for $l = 0, 1, \dots$, we prove the inequality

$$\Phi_l(x_k) \geq \Phi_{l+1}(x_k), \quad \forall x_k. \quad (5.3.18)$$

Obviously, (5.3.18) holds for $l = 0$. For $l = 1$, we have

$$\begin{aligned} \Phi_2(x_k) &= \min_{u_k} \{U(x_k, u_k) + \Phi_1(x_{k+1})\} \\ &\leq \min_{u_k} \{U(x_k, u_k) + \Phi_0(x_{k+1})\} \\ &= \Phi_1(x_k). \end{aligned}$$

According to mathematical induction, for $l = 0, 1, \dots$, we have

$$\Phi_l(x_k) \geq \Phi_{l+1}(x_k) \geq \Phi_{l+2}(x_k) \geq \dots.$$

Let $l \rightarrow \infty$. According to Corollary 5.2.2, we can obtain

$$\Phi_l(x_k) \geq \lim_{l \rightarrow \infty} \Phi_l(x_k) = J^*(x_k).$$

Hence, we have

$$\Phi_0(x_k) \geq J^*(x_k).$$

If $\bar{\gamma}$ satisfies (5.3.17), then we can obtain $\bar{\gamma} \in \Omega_\gamma$. The proof is complete.

Corollary 5.3.1 *Let $\Phi_0(x_k)$ be a positive-definite function and let $\bar{\gamma}$ and $\bar{\sigma}$ be constants that satisfy (5.3.17) and*

$$\Phi_1(x_k) \leq \bar{\sigma} \Phi_0(x_k), \quad (5.3.19)$$

respectively. For $l = 1, 2, \dots$, let the value function $\mathcal{P}_l(x_k)$ satisfy

$$\mathcal{P}_l(x_k) = U(x_k, v(x_k)) + \mathcal{P}_{l-1}(x_{k+1}), \quad (5.3.20)$$

where $\mathcal{P}_0(x_k) = \Phi_0(x_k)$ and

$$v(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + \mathcal{P}_0(x_{k+1})\}. \quad (5.3.21)$$

If $\bar{\gamma}$ and $\bar{\sigma}$ satisfy

Algorithm 5.3.1 Estimation of the parameter γ

Initialization:

Give the computation precision ε .

Iteration:

Step 1. Choose a positive semidefinite function $\Phi_0(x_k) \geq 0$.

Step 2. Obtain the iterative control law and the iterative value function by (5.3.21) and

$$\Phi_1(x_k) = U(x_k, v(x_k)) + \Phi_0(x_{k+1}).$$

Step 3. If $\Phi_1(x_k) \leq \Phi_0(x_k)$, then goto Step 7; else, goto next step.

Step 4. Choose two parameters $0 < \bar{\gamma} < \infty$ and $1 \leq \bar{\sigma} < \infty$ that satisfy (5.3.17) and (5.3.19).

Step 5. If $\bar{\gamma}$ and $\bar{\sigma}$ satisfy (5.3.22), then for $l = 1, 2, \dots$, implement (5.3.20) until the computation precision is achieved, i.e.,

$$|\mathcal{P}_l(x_k) - \mathcal{P}_{l-1}(x_k)| < \varepsilon,$$

where $\mathcal{P}_0(x_k) = \Phi_0(x_k)$, and goto next step; else goto Step 1.

Step 6. Let $\Phi_0(x_k) = \mathcal{P}_l(x_k)$.

Step 7. Choose γ such that $\Phi_0(F(x_k, u_k)) \leq \gamma U(x_k, u_k)$.

Step 8. Return γ .

$$\bar{\sigma} < \frac{1 + \bar{\gamma}}{\bar{\gamma}}, \quad (5.3.22)$$

and there exists a constant $\hat{\gamma}$ such that $\hat{\gamma} U(x_k, u_k) \geq \lim_{l \rightarrow \infty} \mathcal{P}_l(x_k)$, then $\hat{\gamma} \in \Omega_\gamma$.

Proof If $\bar{\gamma}$ and $\bar{\sigma}$ satisfy (5.3.22), according to Theorem 5.3.3, the iterative control law $v(x_k)$ in (5.3.21) is admissible. Then, the limit of iterative value function $\mathcal{P}_l(x_k)$ exists, as $l \rightarrow \infty$. According to Lemma 5.3.1, we can obtain $\hat{\gamma} \in \Omega_\gamma$. This completes the proof of the corollary.

According to Theorem 5.3.6 and Corollary 5.3.1, we can establish an effective iteration algorithm to estimate γ by experiments. The detailed implementation of the iteration algorithm is provided in Algorithm 5.3.1.

Based on the above preparations, we summarize the generalized policy iteration-based iterative ADP algorithm in Algorithm 5.3.2.

Algorithm 5.3.2 The generalized policy iteration algorithm

Initialization:

Choose randomly an array of initial states x_0 . Choose a computation precision ε . Give a positive semidefinite function $\Psi(x_k)$. Construct a sequence $\{N_i\}$, where $N_i > 0, i = 1, 2, \dots$, are positive integers. Estimate γ by Algorithm 5.3.1.

Iteration:

Step 1. Let the iteration index $i = 0$ and $V_0(x_k) = \Psi(x_k)$.

Step 2. Let $i = i + 1$.

Step 3. Let $V_{i,0}(x_k) = V_{i-1}(x_k)$. Obtain γ_i and σ_i by (5.3.2) and (5.3.3), respectively.

Step 4. If σ_i satisfies the convergence criterion (5.3.11), then for $j_i = 0, 1, \dots, N_i$, do **Policy Evaluation**

u**t****o****n**

$$V_{i,j_i}(x_k) = U(x_k, v_{i-1}(x_k)) + V_{i,j_i-1}(F(x_k, v_{i-1}(x_k)));$$

else, let

$$V_i(x_k) = U(x_k, v_{i-1}(x_k)) + V_{i-1}(F(x_k, v_{i-1}(x_k))),$$

and goto Step 2.

Step 5. Do **Policy Improvement**

$$v_i(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + V_i(x_{k+1})\}.$$

Step 6. Let $V_i(x_k) = V_{i,N_i}(x_k)$.

Step 7. If $|V_i(x_k) - V_{i-1}(x_k)| < \varepsilon$, then the optimal cost function and optimal control law are obtained, and goto Step 8; else goto Step 2.

Step 8. Return $V_i(x_k)$ and $v_i(x_k)$.

5.3.3 *Simulation Studies*

To evaluate the performance of our generalized policy iteration algorithm, we choose two examples for numerical experiments.

Example 5.3.1 The first example we choose is an inverted pendulum system [4]. The dynamics of the pendulum is expressed as

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ \frac{g}{\ell} \sin(x_1) - \kappa \ell x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{m\ell^2} \end{pmatrix} u,$$

where $m = 1/2$ kg and $\ell = 1/3$ m are the mass and length of the pendulum bar, respectively. Let $\kappa = 0.2$ and $g = 9.8$ m/s² be the frictional factor and the gravitational acceleration, respectively. Discretization of the system function with the sampling interval $\Delta t = 0.1$ s leads to

$$\begin{bmatrix} x_{1(k+1)} \\ x_{2(k+1)} \end{bmatrix} = \begin{bmatrix} x_{1k} + \Delta t x_{2k} \\ \frac{g}{\ell} \Delta t \sin(x_{1k}) + (1 - \kappa \ell \Delta t) x_{2k} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\Delta t}{m\ell^2} \end{bmatrix} u_k.$$

Let the initial state be $x_0 = [1, -1]^T$. Let the state space be $\mathcal{E} = \{x : -1 \leq x_1 \leq 1, -1 \leq x_2 \leq 1\}$. NNs are used to implement the present generalized policy iteration algorithm. The critic network and the action network are chosen as three-layer BP NNs with the structures of 2–8–1 and 2–8–1, respectively. We choose $p = 5000$ states in \mathcal{E} to train the action and critic networks. To illustrate the effectiveness of the algorithm, four different initial value functions are chosen which are expressed by $\Psi^\varsigma(x_k) = x_k^\top P_\varsigma x_k$, $\varsigma = 1, 2, 3, 4$. Let $P_1 = 0$. Let P_2 – P_4 be initialized by positive-definite matrices given by $P_2 = [2.98, 1.05; 1.05, 5.78]$, $P_3 = [6.47, -0.33; -0.33, 6.55]$, and $P_4 = [22.33, 4.26; 4.26, 7.18]$, respectively. For $i = 0, 1, \dots$, let $q_i = 0.9999$. First, implement Algorithm 5.3.1 and it returns $\gamma = 5.40$. Let the iteration sequence be $\{N_i^\varsigma\}$, where $N_i^\varsigma \in [1, 10]$ be a random non-negative integer. Then, initialized by $\Psi^\varsigma(x_k)$, $\varsigma = 1, 2, 3, 4$, the generalized policy iteration algorithm in Algorithm 5.3.2 is implemented for $i = 15$ iterations. Train the critic and action networks under the learning rate of 0.01 and set the NN training errors as 10^{-6} . The curves of the iterative value functions $V_i(x_k)$ are shown in Fig. 5.10, where we let “In” denote “initial iteration” and let “Lm” denote “limiting iteration”.

For $i = 1, 2, \dots, 15$, define the function Θ_i as

$$\Theta_i = \frac{\sigma_i \gamma_i (1 + \gamma)}{q_i + \gamma_i (1 + \gamma)}. \quad (5.3.23)$$

It can easily be shown that $\Theta_i < 1$ implies σ_i satisfies (5.3.11). The trajectories of the function Θ_i are shown in Fig. 5.11. Under the iteration indices i and j_i , the trajectories of iterative value functions $V_{i,j_i}(x_k)$ for $x_k = x_0$ are shown in Fig. 5.12. From

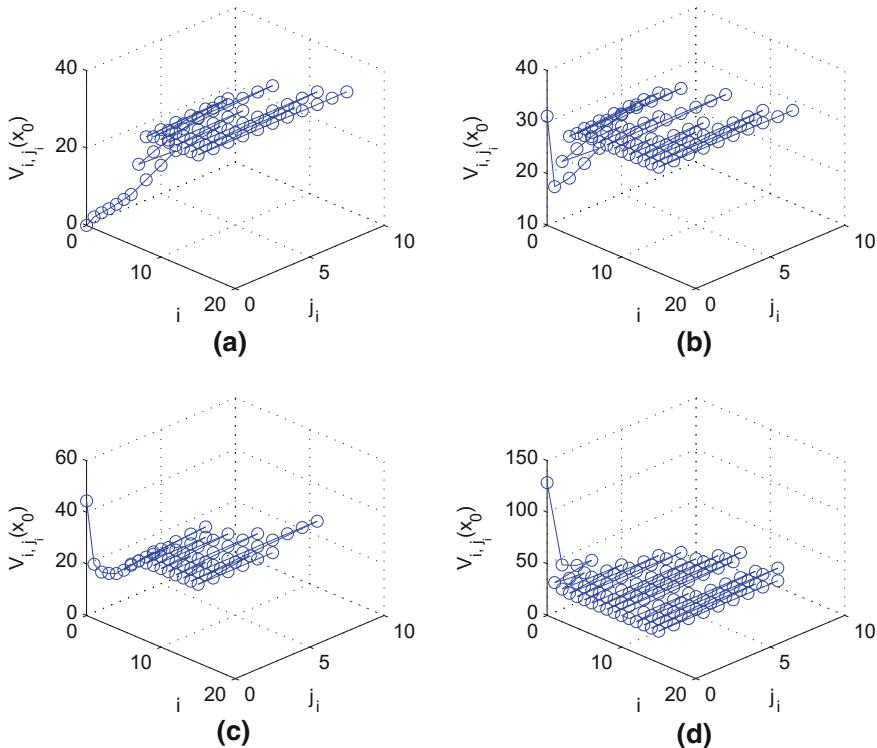


Fig. 5.15 Iterative value function $V_{i,j_i}(x_k)$ for $x_k = x_0$ and different $\bar{\Psi}(x_k)$'s. **a** $V_{i,j_i}(x_k)$ for $\bar{\Psi}^1(x_k)$. **b** $V_{i,j_i}(x_k)$ for $\bar{\Psi}^2(x_k)$. **c** $V_{i,j_i}(x_k)$ for $\bar{\Psi}^3(x_k)$. **d** $V_{i,j_i}(x_k)$ for $\bar{\Psi}^4(x_k)$

Figs. 5.10 and 5.12, we can see that given an arbitrary positive semidefinite function, the iterative value function $V_{i,j_i}(x_k)$ converges to the optimum using the present generalized policy iteration algorithm. The convergence property of the present generalized policy iteration algorithm for nonlinear systems can be verified. From Figs. 5.11 and 5.12, we can see that when $\Theta_i < 1$, the GPI algorithm is convergent and both the policy evaluation and improvement procedures can be implemented. If $\Theta_i < 1$, the iterative control law $v_i(x_k)$ is admissible. Let the execution time T_f be 100 time steps. The trajectories of the iterative control laws and system states are shown in Figs. 5.13 and 5.14, respectively, where the effectiveness of the present generalized policy iteration algorithm for nonlinear systems can be verified.

Example 5.3.2 Our second example is chosen as a nonaffine nonlinear system in [1] with modifications. The system is expressed by

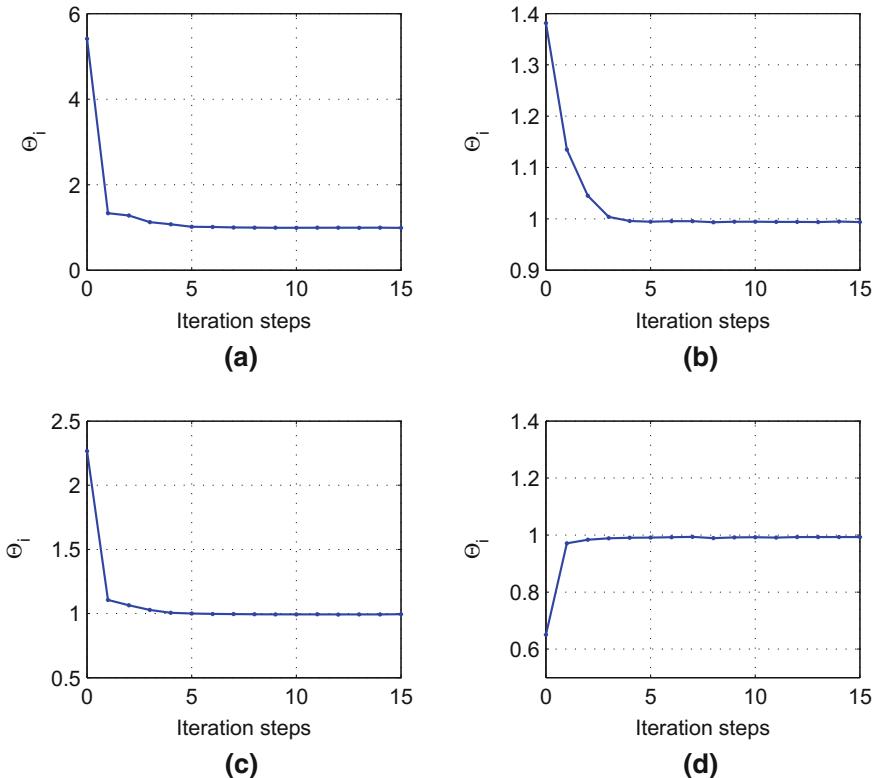


Fig. 5.16 The trajectories of Θ_i for different $\bar{\Psi}(x_k)$'s. **a** Θ_i for $\bar{\Psi}^1(x_k)$. **b** Θ_i for $\bar{\Psi}^2(x_k)$. **c** Θ_i for $\bar{\Psi}^3(x_k)$. **d** Θ_i for $\bar{\Psi}^4(x_k)$

$$\dot{x}_1 = 2 \sin x_1 + x_2 + x_3,$$

$$\dot{x}_2 = x_1 + \sin(-x_2 + u_2),$$

$$\dot{x}_3 = x_3 + \sin u_1.$$

Let $x = [x_1, x_2, x_3]^\top$ be the system state and let $u = [u_1, u_2]^\top$ be the system control input. Let the initial state be $x_0 = [1, -1, 1]^\top$. Let the sampling interval be $\Delta t = 0.1$ s. Let the cost function be expressed by $J(x_0) = \sum_{k=0}^{\infty} (x_k^\top Q x_k + \mathcal{R}(u_k))$, where

$$\mathcal{R}(u_k) = \int_0^{u_k} (\Phi^{-1}(v))^\top R dv.$$

Let Q be an identity matrix with suitable dimension. Let $\Phi(\cdot)$ be a sigmoid function, i.e., $\Phi(\cdot) = \tanh(\cdot)$. Let the state space be $\bar{\mathcal{S}} = \{x: -1 \leq x_1 \leq 1, -1 \leq x_2 \leq 1, -1 \leq x_3 \leq 1\}$. NNs are used to implement the present generalized policy

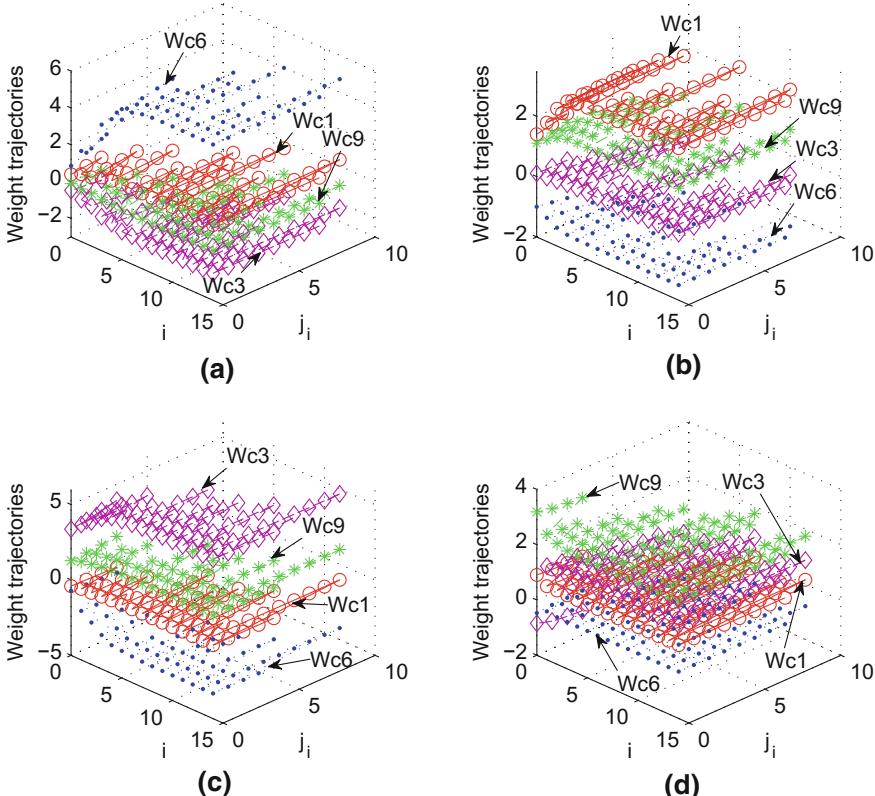


Fig. 5.17 Trajectories of weights for the critic network for different $\bar{\Psi}(x_k)$'s. **a** $W_{cl}, l = 1, 3, 6, 9$, for $\bar{\Psi}^1(x_k)$. **b** $W_{cl}, l = 1, 3, 6, 9$, for $\bar{\Psi}^2(x_k)$. **c** $W_{cl}, l = 1, 3, 6, 9$, for $\bar{\Psi}^3(x_k)$. **d** $W_{cl}, l = 1, 3, 6, 9$, for $\bar{\Psi}^4(x_k)$

iteration algorithm. The critic and action networks are chosen as three-layer BP NNs with the structures of 3–10–1 and 3–10–2, respectively. We choose randomly $p = 10000$ states in $\bar{\mathcal{S}}$ to train the action and critic networks. To illustrate the effectiveness of the algorithm, we also choose four different initial value functions which are expressed by $\bar{\Psi}^\varsigma(x_k) = x_k^T \bar{P}_\varsigma x_k$, $\varsigma = 1, \dots, 4$. Let \bar{P}_1 – \bar{P}_4 be positive-definite matrices given by $\bar{P}_1 = 0.01 \times [0.11, 0, -0.18; 0, 0.12, 0.04; -0.18, 0.04, 0.19]$, $\bar{P}_2 = [9.09, -2.06, 1.77; -2.06, 2.93, -1.89; 1.77, -1.89, 7.69]$, $\bar{P}_3 = [6.89, -8.49, -0.46; -8.49, 15.60, 0.44; -0.46, 0.45, 6.81]$, and $\bar{P}_4 = [51.39, -18.53, -4.64; -18.53, 36.72, 6.22; -4.64, 6.22, 25.34]$, respectively. For $i = 0, 1, \dots$, let $q_i = 0.9999$. First, implement Algorithm 5.3.1 and it returns $\gamma = 6.3941$. Let the iteration sequence be $\{N_i^\varsigma\}$, where $N_i^\varsigma \in [1, 10]$ be a random nonnegative integer. Then, initialized by $\bar{\Psi}^\varsigma(x_k)$, $\varsigma = 1, \dots, 4$, the generalized policy iteration algorithm (Algorithm 5.3.2) is implemented for a total of $i = 20$ iterations. Train the critic and action networks under the learning rate of 0.01 and set the NN training error threshold

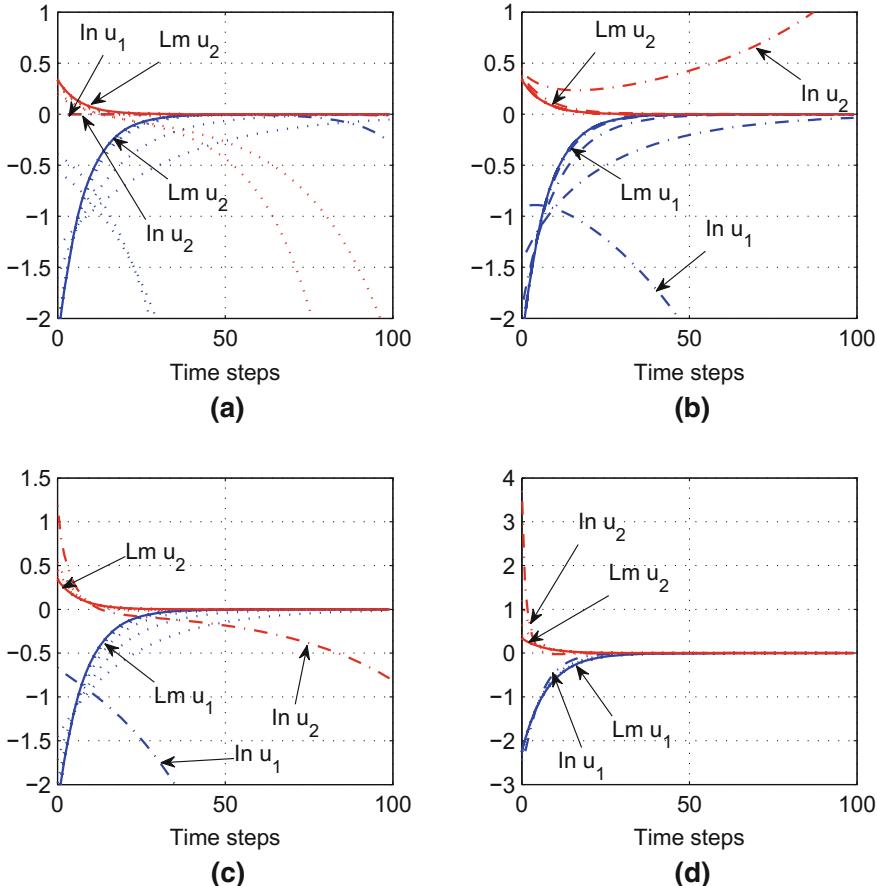


Fig. 5.18 Trajectories of iterative control law $v_i(x_k)$, for different $\bar{\Psi}(x_k)$'s. **a** $v_i(x_k)$ for $\bar{\Psi}^1(x_k)$. **b** $v_i(x_k)$ for $\bar{\Psi}^2(x_k)$. **c** $v_i(x_k)$ for $\bar{\Psi}^3(x_k)$. **d** $v_i(x_k)$ for $\bar{\Psi}^4(x_k)$

as 10^{-6} . Under the iteration indices i and j_i , the trajectories of iterative value functions $V_{i,j_i}(x_k)$ for $x_k = x_0$ are shown in Fig. 5.15, where we can see that initialized by an arbitrary positive semidefinite function, the iterative value function $V_{i,j_i}(x_k)$ converges to the optimum using the present generalized policy iteration algorithm.

For $i = 1, 2, \dots, 15$, define the function Θ_i as (5.3.23) and the trajectories of the function Θ_i are shown in Fig. 5.16.

Let $W_{cl}, l = 1, 2, \dots, 10$, be the first column of the hidden-output weight matrix for the critic network. The convergence trajectories of weights $W_{c1}, W_{c3}, W_{c6}, W_{c9}$ for the critic network are shown in Fig. 5.17, where we can see that the weights for critic network are convergent to the optimum. Other weights of the NNs are omitted here.

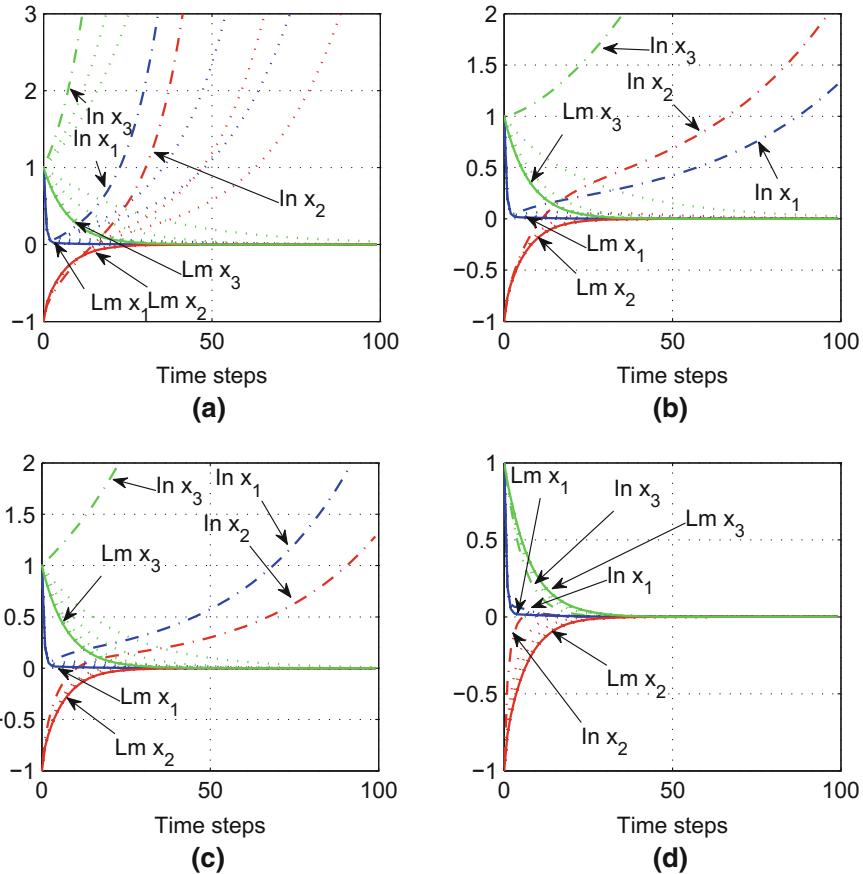


Fig. 5.19 Trajectories of system states for different $\bar{\Psi}(x_k)$'s. **a** State trajectories for $\bar{\Psi}^1(x_k)$. **b** State trajectories for $\bar{\Psi}^2(x_k)$. **c** State trajectories for $\bar{\Psi}^3(x_k)$. **d** State trajectories for $\bar{\Psi}^4(x_k)$

From Figs. 5.15 and 5.16, the convergence property of the present generalized policy iteration algorithm for nonlinear systems can be verified. We can see that for $\Theta_i < 1$, the GPI algorithm is convergent and both the policy evaluation and improvement procedures can be implemented. Let the execution time T_f be 100 time steps. The trajectories of the iterative control laws and system states are shown in Figs. 5.18 and 5.19, respectively. From Fig. 5.16d, we can see that for $\Psi^4(x_k)$, $\Theta_i < 1$ for $i = 1, 2, \dots, 15$. In this case, the iterative control law $v_i(x_k)$ is admissible. This property can be verified from results shown in Figs. 5.18d and 5.19d, respectively. From these simulation results, the effectiveness of the present generalized policy iteration algorithm for nonlinear systems can be verified.

5.4 Conclusions

In this chapter, an effective generalized policy iteration algorithm is developed to solve infinite-horizon optimal control problems for discrete-time nonlinear systems. The iterative ADP algorithm can be implemented by an arbitrary positive semidefinite function. It has been proven that the iterative value function for the generalized policy iteration algorithm converges to the optimum. Convergence criteria of the algorithm are obtained. Admissibility of the iterative control laws is analyzed. Effective methods are presented to estimate the parameter γ of the present algorithm. NNs are used to implement the generalized policy iteration algorithm. Finally, two numerical examples are given to illustrate the effectiveness of the present algorithm.

References

1. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791
2. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2008) Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):943–949
3. Apostol TM (1974) Mathematical analysis, 2nd edn. Addison-Wesley, Boston
4. Beard RW (1995) Improving the closed-loop performance of nonlinear systems. Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY
5. Bertsekas DP (2007) Dynamic programming and optimal control, 3rd edn. Athena Scientific, Belmont
6. Bertsekas DP, Tsitsiklis JN (1996) Neuro-dynamic programming. Athena Scientific, Belmont
7. Dorf RC, Bishop RH (2011) Modern control systems, 12th edn. Prentice-Hall, Upper Saddle River
8. Lewis FL, Vrabie D, Vamvoudakis KG (2012) Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst Mag* 32(6):76–105
9. Lincoln B, Rantzer A (2006) Relaxing dynamic programming. *IEEE Trans Autom Control* 51(8):1249–1260
10. Liu D, Wei Q (2014) Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Trans Neural Network Learn Syst* 25(3):621–634
11. Liu D, Wei Q, Yan P (2015) Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems. *IEEE Trans Syst Man Cybern Syst* 45(12):1577–1591
12. Rantzer A (2006) Relaxed dynamic programming in switching systems. *IEE Proc Control Theory Appl* 153(5):567–574
13. Si J, Wang YT (2001) Online learning control by association and reinforcement. *IEEE Trans Neural Network* 12(2):264–276
14. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
15. Vrabie D, Vamvoudakis KG, Lewis FL (2013) Optimal adaptive control and differential games by reinforcement learning principles. IET, London
16. Wei Q, Liu D (2014) A novel iterative θ -adaptive dynamic programming for discrete-time nonlinear systems. *IEEE Trans Autom Sci Eng* 11(4):1176–1190
17. Wei Q, Liu D, Lin H (2016) Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems. *IEEE Trans Cybern* 46(3):840–853
18. Wei Q, Liu D, Yang X (2015) Infinite horizon self-learning optimal control of nonaffine discrete-time nonlinear systems. *IEEE Trans Neural Network Learn Syst* 26(4):866–879

Chapter 6

Error Bounds of Adaptive Dynamic Programming Algorithms

6.1 Introduction

Value iteration and policy iteration are two basic classes of ADP algorithms which can solve optimal control problems for nonlinear dynamical systems with continuous state and action spaces. Value iteration algorithms can solve the optimal control problems for nonlinear systems without requiring an initial stabilizing control policy. Value iteration algorithms iterate between value function update and policy improvement until the iterative value functions converge to the optimal one. Value iteration-based ADP algorithms have been used for solving the Bellman equation [3, 8, 17, 19, 30, 31], the near-optimal control of discrete-time affine nonlinear systems with control constraints [20, 34], the finite-horizon optimal control problem [10, 28], the optimal tracking control problem [33, 35], and the optimal control of unknown nonaffine nonlinear discrete-time systems with discount factor in the cost function [19, 29]. In contrast to value iteration algorithms, policy iteration algorithms [1, 11, 14, 18, 27] always require an initial stabilizing control policy. The policy iteration is built to iterate between policy evaluation and policy improvement until the policy converges to the optimal one. The policy iteration-based ADP approaches were also developed for optimal control of discrete-time nonlinear dynamical systems [7, 18]. For all the iterative ADP algorithms mentioned above, it is assumed that the value function and control policy update equations can be exactly solved at each iteration. Furthermore, optimistic policy iteration [25] represents a spectrum of iterative algorithms which includes value iteration and policy iteration algorithms, and it is also known as generalized policy iteration [24, 26] or modified policy iteration [22]. The optimistic policy iteration algorithm has not been widely applied to ADP for solving optimal control problems of nonlinear dynamical systems.

On the other hand, an inequality version of the Bellman equation was used to derive bounds on the optimal cost function [12]. For undiscounted optimal control problems of discrete-time nonlinear systems, a relaxed value iteration scheme [23] based on the inequality version of Bellman equation [12] was introduced to derive the upper

and lower bounds of the optimal cost function, where the distance from optimal values can be kept within prescribed bounds. In [16], the relaxed value iteration scheme was used to solve the optimal switching between linear systems, the optimal control of a linear system with piecewise linear cost, and a partially observable Markov decision problem. In [9], the relaxed value iteration scheme was applied to receding horizon control schemes for discrete-time nonlinear systems. Based on [23], a new expression of approximation errors at each iteration was introduced to establish convergence analysis results for the approximate value iteration algorithm [17].

For optimal control problems with continuous state and action spaces, ADP methods use a critic neural network to approximate the value function, and use an action neural network to approximate the control policy. Iterating on these approximate schemes will inevitably give rise to approximation errors. Therefore, we establish error bounds for the approximate value iteration, approximate policy iteration, and approximate optimistic policy iteration in this chapter [21]. A new assumption is utilized instead of the contraction assumption in discounted optimal control problems [16]. It is shown that the iterative approximate value function can converge to a finite neighborhood of the optimal value function under some mild conditions. To implement the present algorithms, two multilayer feedforward neural networks are used to approximate the value function and control policy. A simulation example is given to demonstrate the present results. Furthermore, we also present error bound analysis results of Q-function for the action-dependent ADP to solve discounted optimal control problems of unknown discrete-time nonlinear systems [32]. It is shown that the approximate Q-function will converge to a finite neighborhood of the optimal Q-function. Results in this chapter will complement the analysis results developed in Chaps. 2–5 by introducing various error conditions for different VI and PI algorithms [17–19, 28–30].

6.2 Error Bounds of ADP Algorithms for Undiscounted Optimal Control Problems

6.2.1 Problem Formulation

Consider the discrete-time deterministic nonlinear dynamical systems described by

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \dots, \quad (6.2.1)$$

where $x_k \in \mathbb{R}^n$ is the system state at time k and $u_k \in \mathbb{R}^m$ is the control input. Let x_0 be the initial state. Assume that the system function $F(x_k, u_k)$ is Lipschitz continuous on a compact set $\Omega \subset \mathbb{R}^n$ containing the origin, and $F(0, 0) = 0$. Hence, $x = 0$ is an equilibrium state of the system (6.2.1) under the control $u = 0$. Assume that system (6.2.1) is stabilizable on the compact set Ω [3].

Define the undiscounted infinite-horizon cost function as

$$J(x_0, u) = \sum_{k=0}^{\infty} U(x_k, u_k), \quad (6.2.2)$$

where U is a positive-definite utility function. In this chapter, the utility function is chosen as the quadratic form $U(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k$, where Q and R are positive-definite matrices with suitable dimensions. Our goal is to find a control policy $u_k = u(x_k)$ which can minimize the cost function (6.2.2) for every initial state $x_0 \in \Omega$. For optimal control problems, the designed feedback control must not only stabilize the system (6.2.1) on Ω but also guarantee that the cost function (6.2.2) is finite, i.e., the control must be admissible (cf. Definition 2.2.1 or [3]).

For any admissible control policy $u = \mu(x) \in \mathcal{A}(\Omega)$, the mapping from any state x to the value of (6.2.2) is the value function defined by $V^\mu(x) \triangleq J(x, \mu(x))$, where $\mathcal{A}(\Omega)$ is the set of admissible control policies associated with the controllable set Ω of states. Then, we define the optimal value function as

$$V^*(x) = \inf_{\mu} \{V^\mu(x)\}. \quad (6.2.3)$$

Note that the optimal value function defined here is the same as the optimal cost function, i.e.,

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

According to Bellman's principle of optimality [4, 13], the optimal value function $V^*(x)$ satisfies the Bellman equation

$$V^*(x) = \inf_{\mu} \{U(x, \mu) + V^*(F(x, \mu))\}. \quad (6.2.4)$$

If it can be solved for V^* , the optimal control policy $\mu^*(x)$ can be obtained by

$$\mu^*(x) = \arg \inf_{\mu} \{U(x, \mu) + V^*(F(x, \mu))\}.$$

Similar to [6], we define the Hamiltonian as

$$H(x, u, V) = U(x, u) + V(F(x, u)).$$

Then, we define an operator \mathcal{T}_μ as

$$(\mathcal{T}_\mu V)(x) = H(x, \mu(x), V),$$

and define an operator \mathcal{T} as

$$(\mathcal{T} V)(x) = \min_{\mu} H(x, \mu(x), V).$$

For convenience, \mathcal{T}_μ^k denotes the k time composition of the operator \mathcal{T}_μ , i.e.,

$$(\mathcal{T}_\mu^k V)(x) = (\mathcal{T}_\mu(\mathcal{T}_\mu^{k-1} V))(x). \quad (6.2.5)$$

Similarly, the mapping \mathcal{T}^k is defined by

$$(\mathcal{T}^k V)(x) = (\mathcal{T}(\mathcal{T}^{k-1} V))(x).$$

Therefore, the Bellman equation (6.2.4) can be written compactly as

$$V^* = \mathcal{T}V^*,$$

i.e., V^* is the fixed point of \mathcal{T} .

In this chapter, we assume the following monotonicity property holds, which was also used in [6].

Assumption 6.2.1 *If $V \leq V'$, then $H(x, u, V) \leq H(x, u, V')$, $\forall x, u$.*

Besides the above monotonicity assumption, the contraction assumption in [5] is often required for discounted optimal control problems. However, for undiscounted optimal control problems, we utilize the following assumption from [16] instead of the contraction assumption.

Assumption 6.2.2 Suppose the condition $0 \leq V^*(F(x, u)) \leq \lambda U(x, u)$ holds uniformly for some $0 < \lambda < \infty$.

The positive constant λ is usually larger than 1 and gives a measure on how “contractive” the optimally controlled system is, i.e., how close the value function is to the cost of a single step [16]. A smaller λ implies a faster convergence.

The Bellman equation (6.2.4) reduces to the Riccati equation in the case of linear quadratic regulator, which can be efficiently solved. In the case of general nonlinear systems, it is very difficult to solve the Bellman equation analytically. ADP methods using function approximation structures are proposed to learn the near-optimal control policy and value function associated with the Bellman equation. Because of the approximation errors, the value function and control policy are generally impossible to obtain accurately at each iteration. Therefore, in this chapter, we analyze the convergence and establish the error bounds for ADP algorithms considering function approximation errors. We note that the analysis conducted in this chapter uses a different notation. Most notably, the use of “operators” \mathcal{T} and \mathcal{T}_μ can significantly simplify our presentation.

6.2.2 Approximate Value Iteration

We first present the algorithm for exact value iteration and then the algorithm for approximate value iteration with analysis results for error bounds.

A. Value Iteration

For the value iteration algorithm, it starts with any initial positive-definite value function $V_0(x)$ or $V_0(\cdot) = 0$. Then, the control policy $v_0(x)$ can be obtained by

$$v_0(x) = \arg \min_u \{U(x, u) + V_0(F(x, u))\}. \quad (6.2.6)$$

For $i = 1, 2, \dots$, the value iteration algorithm iterates between the value function update

$$\begin{aligned} V_i(x) &= \mathcal{T}V_{i-1}(x) \\ &= \min_u \{U(x, u) + V_{i-1}(F(x, u))\} \\ &= U(x, v_{i-1}(x)) + V_{i-1}(F(x, v_{i-1}(x))), \end{aligned} \quad (6.2.7)$$

and policy improvement

$$v_i(x) = \arg \min_u \{U(x, u) + V_i(F(x, u))\}. \quad (6.2.8)$$

It is required that $V_i(0) = 0$ and $v_i(0) = 0$, $\forall i \geq 1$. The value iteration algorithm does not require an initial stabilizing control policy. Note that the algorithm described in (6.2.6)–(6.2.8) is the same as the one given earlier in (2.2.10)–(2.2.12) or (2.2.22)–(2.2.24).

We list some key expressions in Table 6.1 which can be used in this section for establishing analysis results.

According to Assumption 6.2.1, it is easy to give the following lemma.

Lemma 6.2.1 *If $V_0 \geq \mathcal{T}V_0$, the value function sequence $\{V_i\}$ is a monotonically nonincreasing sequence, i.e., $V_i \geq V_{i+1}$, $\forall i \geq 0$. If $V_0 \leq \mathcal{T}V_0$, the value function sequence $\{V_i\}$ is a monotonically nondecreasing sequence, i.e., $V_i \leq V_{i+1}$, $\forall i \geq 0$.*

The lemma can be proved by considering the key expressions in Table 6.1.

For undiscounted optimal control problems, the convergence of value iteration algorithm has been given in the following theorem [23] (cf. Theorem 2.2.2).

Theorem 6.2.1 *Let Assumptions 6.2.1 and 6.2.2 hold. Suppose that $0 \leq \alpha V^* \leq V_0 \leq \beta V^*$, $0 \leq \alpha \leq 1$, and $1 \leq \beta < \infty$. The value function V_i and the control policy v_{i+1} are iteratively updated by (6.2.7) and (6.2.8). Then, the value function sequence $\{V_i\}$ approaches V^* according to the inequalities*

Table 6.1 Key expressions used in Sect. 6.2.2

Key expression	Location
$V \leq V' \Rightarrow \mathcal{T}_u V \leq \mathcal{T}_u V'$	Assumption 6.2.1
$V_i = \mathcal{T}V_{i-1}$	(6.2.7)

$$\left[1 - \frac{1-\alpha}{(1+\lambda^{-1})^i}\right]V^* \leq V_i \leq \left[1 + \frac{\beta-1}{(1+\lambda^{-1})^i}\right]V^*, \quad \forall i \geq 1.$$

Moreover, the value function V_i converges to V^* uniformly on Ω as $i \rightarrow \infty$.

B. Error Bounds for Approximate Value Iteration

For the approximate value iteration algorithm, function approximation structures like neural networks are usually used to approximate the value function V_i and the control policy v_i . In general, the value function and control policy update equations (6.2.7) and (6.2.8) cannot be solved accurately, because we only have some samples from the state space and there exist approximation errors for function approximation structures. Here, we use \hat{V}_i and \hat{v}_i , $i = 0, 1, \dots$, to stand for the approximate expressions of V_i and v_i , respectively, where $\hat{V}_0 = V_0$.

We assume that there exist finite positive constants $\underline{\delta} \leq 1$ and $\bar{\delta} \geq 1$ such that

$$\underline{\delta} \mathcal{T}_{\hat{v}_i} \hat{V}_i \leq \hat{V}_{i+1} \leq \bar{\delta} \mathcal{T}_{\hat{v}_i} \hat{V}_i, \quad \forall i = 0, 1, \dots, \quad (6.2.9)$$

hold uniformly. Similarly, we also assume that there exist finite positive constants $\underline{\zeta} \leq 1$ and $\bar{\zeta} \geq 1$ such that

$$\underline{\zeta} \mathcal{T} \hat{V}_i \leq \mathcal{T}_{\hat{v}_i} \hat{V}_i \leq \bar{\zeta} \mathcal{T} \hat{V}_i, \quad \forall i = 0, 1, \dots, \quad (6.2.10)$$

hold uniformly. Combining (6.2.9) and (6.2.10), we can get

$$\underline{\zeta} \underline{\delta} \mathcal{T} \hat{V}_i \leq \hat{V}_{i+1} \leq \bar{\zeta} \bar{\delta} \mathcal{T} \hat{V}_i. \quad (6.2.11)$$

For simplicity, (6.2.11) can be written as

$$\underline{\eta} \mathcal{T} \hat{V}_i \leq \hat{V}_{i+1} \leq \bar{\eta} \mathcal{T} \hat{V}_i, \quad \forall i = 0, 1, \dots, \quad (6.2.12)$$

by denoting

$$\underline{\eta} \triangleq \underline{\zeta} \underline{\delta} \leq 1, \quad \bar{\eta} \triangleq \bar{\zeta} \bar{\delta} \geq 1. \quad (6.2.13)$$

Based on Assumptions 6.2.1 and 6.2.2, we can establish the error bounds for the approximate value iteration by the following theorem.

Theorem 6.2.2 *Let Assumptions 6.2.1 and 6.2.2 hold. Suppose that $0 \leq \alpha V^* \leq V_0 \leq \beta V^*$, $0 \leq \alpha \leq 1$ and $1 \leq \beta < \infty$. The approximate value function \hat{V}_i satisfies the iterative error condition (6.2.12). Then, the value function sequence $\{\hat{V}_i\}$ approaches V^* according to the following inequalities*

$$\begin{aligned} & \eta \left[1 - \sum_{j=1}^i \frac{\lambda^j \underline{\eta}^{j-1} (1 - \underline{\eta})}{(\lambda + 1)^j} - \frac{\lambda^i \underline{\eta}^i (1 - \alpha)}{(\lambda + 1)^i} \right] V^* \\ & \leq \hat{V}_{i+1} \leq \bar{\eta} \left[1 + \sum_{j=1}^i \frac{\lambda^j \bar{\eta}^{j-1} (\bar{\eta} - 1)}{(\lambda + 1)^j} + \frac{\lambda^i \bar{\eta}^i (\beta - 1)}{(\lambda + 1)^i} \right] V^*, \quad i \geq 0, \quad (6.2.14) \end{aligned}$$

where we define $\sum_{j=1}^i (\cdot) = 0$ for $i < 1$. Moreover, the value function sequence $\{\hat{V}_i\}$ converges to a finite neighborhood of V^* uniformly on Ω as $i \rightarrow \infty$, i.e.,

$$\frac{\underline{\eta}}{1 + \lambda - \lambda \underline{\eta}} V^* \leq \lim_{i \rightarrow \infty} \hat{V}_i \leq \frac{\bar{\eta}}{1 + \lambda - \lambda \bar{\eta}} V^*, \quad (6.2.15)$$

under the condition $\bar{\eta} < \frac{1}{\lambda} + 1$.

Proof First, we prove the lower bound of the approximate value function \hat{V}_{i+1} by mathematical induction. Letting $i = 0$ in (6.2.12), we can obtain $\hat{V}_1 \geq \underline{\eta} \mathcal{T} \hat{V}_0 = \underline{\eta} \mathcal{T} V_0$. Considering $\alpha V^* \leq V_0$, we can get

$$\hat{V}_1 \geq \underline{\eta} \mathcal{T} V_0 \geq \alpha \underline{\eta} \mathcal{T} V^* = \alpha \underline{\eta} V^*.$$

Thus, the lower bound of \hat{V}_{i+1} holds for $i = 0$. According to (6.2.12), Assumptions 6.2.1 and 6.2.2, we can get

$$\begin{aligned} \hat{V}_2 & \geq \underline{\eta} \mathcal{T} \hat{V}_1 \\ & = \underline{\eta} \min_u \{U(x, u) + \hat{V}_1(F(x, u))\} \\ & \geq \underline{\eta} \min_u \{U(x, u) + \alpha \underline{\eta} V^*(F(x, u))\} \\ & \geq \underline{\eta} \min_u \left\{ \left(1 + \lambda \frac{\alpha \underline{\eta} - 1}{\lambda + 1}\right) U(x, u) + \left(\alpha \underline{\eta} - \frac{\alpha \underline{\eta} - 1}{\lambda + 1}\right) V^*(F(x, u)) \right\} \\ & = \underline{\eta} \left(1 + \lambda \frac{\alpha \underline{\eta} - 1}{\lambda + 1}\right) \min_u \{U(x, u) + V^*(F(x, u))\} \\ & = \underline{\eta} \left(1 - \frac{\lambda(1 - \underline{\eta})}{\lambda + 1} - \frac{\lambda \underline{\eta}(1 - \alpha)}{\lambda + 1}\right) V^*. \end{aligned}$$

Hence, the lower bound of \hat{V}_{i+1} holds for $i = 1$. Assume the lower bound in (6.2.14) holds for $i = l$, i.e.,

$$\hat{V}_{l+1} \geq \underline{\eta} \left[1 - \sum_{j=1}^l \frac{\lambda^j \underline{\eta}^{j-1} (1 - \underline{\eta})}{(\lambda + 1)^j} - \frac{\lambda^l \underline{\eta}^l (1 - \alpha)}{(\lambda + 1)^l} \right] V^* \triangleq \Theta V^*.$$

When $i = l + 1$, according to (6.2.12) and the above inequality, we have

$$\begin{aligned}\hat{V}_{l+2} &\geq \underline{\eta} \mathcal{X} \hat{V}_{l+1} \\ &= \underline{\eta} \min_u \{U(x, u) + \hat{V}_{l+1}(F(x, u))\} \\ &\geq \underline{\eta} \min_u \{U(x, u) + \Theta V^*(F(x, u))\}.\end{aligned}$$

Let

$$\Xi \triangleq - \sum_{j=1}^{l+1} \frac{\lambda^{j-1} \underline{\eta}^{j-1} (1 - \underline{\eta})}{(\lambda + 1)^j} - \frac{\lambda^l \underline{\eta}^{l+1} (1 - \alpha)}{(\lambda + 1)^{l+1}}. \quad (6.2.16)$$

According to the definitions of Θ and Ξ , we have

$$\begin{aligned}1 + \lambda \Xi + \Xi &= 1 - (\lambda + 1) \left[\frac{1 - \underline{\eta}}{\lambda + 1} + \sum_{j=2}^{l+1} \frac{\lambda^{j-1} \underline{\eta}^{j-1} (1 - \underline{\eta})}{(\lambda + 1)^j} - \frac{\lambda^l \underline{\eta}^{l+1} (1 - \alpha)}{(\lambda + 1)^{l+1}} \right] \\ &= \underline{\eta} - \sum_{j=1}^l \frac{\lambda^j \underline{\eta}^j (1 - \underline{\eta})}{(\lambda + 1)^j} - \frac{\lambda^l \underline{\eta}^{l+1} (1 - \alpha)}{(\lambda + 1)^l} \\ &= \Theta.\end{aligned}$$

According to Assumption 6.2.2 and $\Xi \leq 0$, we have

$$\lambda \Xi U(x, u) - \Xi V^*(F(x, u)) \leq 0. \quad (6.2.17)$$

Therefore, noticing $1 + \lambda \Xi = \Theta - \Xi$, we have

$$\begin{aligned}\hat{V}_{l+2} &\geq \underline{\eta} \min_u \{(1 + \lambda \Xi) U(x, u) + (\Theta - \Xi) V^*(F(x, u))\} \\ &= \underline{\eta} (1 + \lambda \Xi) \min_u \{U(x, u) + V^*(F(x, u))\} \\ &= \underline{\eta} \left[1 - \sum_{j=1}^{l+1} \frac{\lambda^j \underline{\eta}^{j-1} (1 - \underline{\eta})}{(\lambda + 1)^j} - \frac{\lambda^{l+1} \underline{\eta}^{l+1} (1 - \alpha)}{(\lambda + 1)^{l+1}} \right] V^*.\end{aligned}$$

Also, the upper bound can be proved similarly. Therefore, the lower and upper bounds of \hat{V}_{l+1} in (6.2.14) have been proved.

Finally, we prove that the value function sequence $\{\hat{V}_i\}$ converges to a finite neighborhood of V^* uniformly on Ω as $i \rightarrow \infty$ if $\bar{\eta} < 1/\lambda + 1$. Since the sequence $\left\{ \frac{\lambda^j \underline{\eta}^{j-1} (1 - \underline{\eta})}{(\lambda + 1)^j} \right\}$ is geometric, we have

$$\sum_{j=1}^i \frac{\lambda^j \underline{\eta}^{j-1} (1 - \underline{\eta})}{(\lambda + 1)^j} = \frac{\frac{\lambda(1 - \underline{\eta})}{\lambda + 1} \left(1 - \left(\frac{\lambda \underline{\eta}}{\lambda + 1}\right)^i\right)}{1 - \frac{\lambda \underline{\eta}}{\lambda + 1}}.$$

Considering $\frac{\lambda \underline{\eta}}{\lambda + 1} \leq \frac{\lambda \bar{\eta}}{\lambda + 1} < 1$, we have

$$\lim_{i \rightarrow \infty} \hat{V}_i \geq \frac{\underline{\eta}}{1 + \lambda - \lambda \underline{\eta}} V^*.$$

For the upper bound, if $\lambda \bar{\eta}/(\lambda + 1) < 1$, i.e., $\bar{\eta} < 1/\lambda + 1$, we can show

$$\lim_{i \rightarrow \infty} \hat{V}_i \leq \frac{\bar{\eta}}{1 + \lambda - \lambda \bar{\eta}} V^*.$$

Thus, we complete the proof.

Remark 6.2.1 We can find that the lower and upper bounds in (6.2.15) are both monotonically increasing functions of $\underline{\eta}$ and $\bar{\eta}$, respectively. The condition $\bar{\eta} < 1/\lambda + 1$ should be satisfied such that the upper bound in (6.2.15) is finite and positive. Since $\underline{\eta} \leq 1$, the lower bound in (6.2.15) is always positive. The values of $\underline{\eta}$ and $\bar{\eta}$ may gradually be refined during the iterative process similar to [2], in which a crude initial operator approximation is gradually refined with new iterations. We can also derive that a larger λ will lead to a slower convergence rate and a larger error bound. A larger λ also requires more accurate iteration to converge. When $\underline{\eta} = \bar{\eta} = 1$, Theorem 6.2.2 reduces to Theorem 6.2.1 and the value function sequence $\{\hat{V}_i\}$ converges to V^* uniformly on Ω as $i \rightarrow \infty$.

6.2.3 Approximate Policy Iteration

We first present and analyze the exact policy iteration and then establish the error bounds for approximate policy iteration.

A. Policy Iteration

For the policy iteration algorithm, an initial stabilizing control policy is usually required. In this section, we start the policy iteration from an initial value function V_0 such that $V_0 \geq \mathcal{T}V_0$. We can see that the control policy $\pi_0(x)$ obtained by

$$\pi_0(x) = \arg \min_u \{U(x, u) + V_0(F(x, u))\} \quad (6.2.18)$$

is an asymptotically stable control law for the system (6.2.1), because

Table 6.2 The iterative process of the PI algorithm in (6.2.18)–(6.2.20)

(empty)	$\pi_0 \rightarrow V_1^\pi$ (6.2.19) evaluation	$\pi_1 \rightarrow V_2^\pi$ (6.2.19) evaluation	...
$V_0 \rightarrow \pi_0$ (6.2.18) minimization	$V_1^\pi \rightarrow \pi_1$ (6.2.20) minimization	$V_2^\pi \rightarrow \pi_2$ (6.2.20) minimization	...
$i = 0$	$i = 1$	$i = 2$...

$$\begin{aligned} V_0(F(x, \pi_0(x))) - V_0(x) &\leq V_0(F(x, \pi_0(x))) - (\mathcal{T}V_0)(x) \\ &= -U(x, \pi_0(x)) \leq 0, \end{aligned}$$

is negative definite. For $i = 1, 2, \dots$, the policy iteration algorithm iterates between policy evaluation

$$V_i^\pi(x) = U(x, \pi_{i-1}(x)) + V_i^\pi(F(x, \pi_{i-1}(x))), \quad (6.2.19)$$

and policy improvement

$$\pi_i(x) = \arg \min_u \{U(x, u) + V_i^\pi(F(x, u))\}. \quad (6.2.20)$$

Note that a short expression for (6.2.19) is given by $V_i^\pi = \mathcal{T}_{\pi_{i-1}} V_i^\pi$.

The flow of iterations of the PI algorithm in (6.2.18)–(6.2.20) is illustrated in Table 6.2, where the iteration flows as in Fig. 2.1. The algorithm starts with a special initial value function so that an admissible control law $\pi_0(x_k)$ is obtained from (6.2.18). Comparing between Tables 6.2 and 4.1, we note that the two descriptions are the same except the initialization step. The algorithm in (6.2.18)–(6.2.20) starts at $i = 0$ using an initial value function V_0 satisfying $V_0 \geq \mathcal{T}V_0$, whereas the PI algorithm in (4.2.5)–(4.2.6) starts with an admissible control law $v_0(x_k)$. For a particular problem, the one with initial condition easier to obtain will usually be chosen.

When the policy evaluation equation (6.2.19) cannot be solved directly, the following iterative process can be used to solve the value function at the policy evaluation step

$$V_i^{\pi(j)}(x) = U(x, \pi_{i-1}(x)) + V_i^{\pi(j-1)}(F(x, \pi_{i-1}(x))), \quad j > 0, \quad (6.2.21)$$

with $V_i^{\pi(0)} = V_{i-1}^\pi$, $\forall i \geq 1$ and $V_1^{\pi(0)} = V_0^\pi = V_0$. The solution of (6.2.21) for the value function can be obtained after infinite number of iterations as $V_i^\pi = V_i^{\pi(\infty)}$ (cf. Lemma 6.2.2 presented next). If a finite number of iterations is employed in the calculation of (6.2.21), it gives approximations to the evaluation in (6.2.19) and the

Table 6.3 Key expressions used in Sect. 6.2.3

Key expression	Location
$V \leq V' \Rightarrow \mathcal{T}_u V \leq \mathcal{T}_u V'$	Assumption 6.2.1
$V_i^\pi = \mathcal{T}_{\pi_{i-1}} V_i^\pi$	(6.2.19)
$\mathcal{T} V_i^\pi = \mathcal{T}_{\pi_i} V_i^\pi$	(6.2.20)
$V_i^{\pi(j)} = \mathcal{T}_{\pi_{i-1}} V_i^{\pi(j-1)}$	(6.2.21)
$V_i^{\pi(0)} = V_{i-1}^\pi,$	(6.2.21)
$V_1^{\pi(0)} = V_0^\pi = V_0$	

algorithm becomes the optimistic policy iteration algorithm to be studied later in Sect. 6.2.4.

We list in Table 6.3 some key expressions which will be used in this section for establishing analysis results.

Now, using the expressions in Table 6.3, we can prove the following lemma.

Lemma 6.2.2 *Let Assumption 6.2.1 hold. Suppose that $V_0 \geq \mathcal{T} V_0$. Let π_{i+1} and $V_i^{\pi(j)}$ be updated by (6.2.20) and (6.2.21). Then, the sequence $\{V_i^{\pi(j)}\}$ is a monotonically nonincreasing sequence, i.e., $V_i^{\pi(j)} \leq V_i^{\pi(j-1)}$, $\forall i \geq 1$. Moreover, as $j \rightarrow \infty$, the limit of $V_i^{\pi(j)}$ denoted by $V_i^{\pi(\infty)}$ exists, and it is equal to V_i^π , $\forall i \geq 1$.*

Proof We prove the lemma by mathematical induction. Letting $i = 1$ and $j = 1$ in (6.2.21), we can obtain

$$\begin{aligned}
 V_1^{\pi(1)}(x) &= \mathcal{T}_{\pi_0} V_1^{\pi(0)}(x) \\
 &= U(x, \pi_0(x)) + V_1^{\pi(0)}(F(x, \pi_0(x))) \\
 &= U(x, \pi_0(x)) + V_0(F(x, \pi_0(x))) \\
 &= \mathcal{T} V_0 \leq V_0 = V_1^{\pi(0)}(x).
 \end{aligned} \tag{6.2.22}$$

For $j = 2$, according to (6.2.21) and Assumption 6.2.1, we have

$$V_1^{\pi(2)}(x) = \mathcal{T}_{\pi_0} V_1^{\pi(1)}(x) \leq \mathcal{T}_{\pi_0} V_1^{\pi(0)}(x) = V_1^{\pi(1)}(x).$$

Assume that $V_1^{\pi(j)} \leq V_1^{\pi(j-1)}$ is true for $j = l$, $l = 1, 2, \dots$, i.e., $V_1^{\pi(l)} \leq V_1^{\pi(l-1)}$. For $j = l + 1$, we have

$$V_1^{\pi(l+1)} = \mathcal{T}_{\pi_0} V_1^{\pi(l)} \leq \mathcal{T}_{\pi_0} V_1^{\pi(l-1)} = V_1^{\pi(l)}.$$

Therefore, we obtain $V_i^{\pi(j)} \leq V_i^{\pi(j-1)}$ for $i = 1$ by induction. Since the sequence $\{V_i^{\pi(j)}\}$ is a monotonically nonincreasing sequence and $V_i^{\pi(j)} \geq 0$, the limit of $V_i^{\pi(j)}$ exists, which is denoted by $V_i^{\pi(\infty)}$ and satisfies $V_i^{\pi(\infty)} \leq V_i^{\pi(j)}$. Considering

$$V_1^{\pi(j+1)}(x) = U(x, \pi_0(x)) + V_1^{\pi(j)}(F(x, \pi_0(x))), \quad j \geq 0,$$

we have

$$V_1^{\pi(j+1)}(x) \geq U(x, \pi_0(x)) + V_1^{\pi(\infty)}(F(x, \pi_0(x))), \quad j \geq 0. \quad (6.2.23)$$

Letting $j \rightarrow \infty$ in (6.2.23), we have

$$V_1^{\pi(\infty)}(x) \geq U(x, \pi_0(x)) + V_1^{\pi(\infty)}(F(x, \pi_0(x))). \quad (6.2.24)$$

Similarly, we obtain

$$V_1^{\pi(\infty)}(x) \leq V_1^{\pi(j+1)}(x) = U(x, \pi_0(x)) + V_1^{\pi(j)}(F(x, \pi_0(x))), \quad j \geq 0. \quad (6.2.25)$$

Letting $j \rightarrow \infty$ in (6.2.25), we can obtain

$$V_1^{\pi(\infty)}(x) \leq U(x, \pi_0(x)) + V_1^{\pi(\infty)}(F(x, \pi_0(x))). \quad (6.2.26)$$

Combining (6.2.24) and (6.2.26), we get

$$V_1^{\pi(\infty)}(x) = U(x, \pi_0(x)) + V_1^{\pi(\infty)}(F(x, \pi_0(x))).$$

Considering (6.2.19), we can obtain that $V_i^{\pi(\infty)}(x) = V_i^\pi(x)$ holds for $i = 1$.

We assume that $V_i^{\pi(j)} \geq V_i^{\pi(j+1)}$ holds and $V_i^{\pi(\infty)}(x) = V_i^\pi(x)$, $\forall i \geq 1$. Then, considering (6.2.19)–(6.2.21), for $j = 0$, we can get

$$V_{i+1}^{\pi(1)} = \mathcal{T}_{\pi_i} V_{i+1}^{\pi(0)} = \mathcal{T}_{\pi_i} V_i^\pi = \mathcal{T} V_i^\pi \leq V_i^\pi = V_{i+1}^{\pi(0)}.$$

According to (6.2.21) and Assumption 6.2.1, for $j = 1$, we have

$$V_{i+1}^{\pi(2)} = \mathcal{T}_{\pi_i} V_{i+1}^{\pi(1)} \leq \mathcal{T}_{\pi_i} V_{i+1}^{\pi(0)} = V_{i+1}^{\pi(1)}.$$

Similarly, we can obtain that $V_{i+1}^{\pi(j+1)} \leq V_{i+1}^{\pi(j)}$ holds for $j = 0, 1, \dots$, by induction, and $V_{i+1}^{\pi(\infty)}(x) = V_{i+1}^\pi(x)$. We have now shown $V_i^{\pi(j)} \leq V_i^{\pi(j-1)}$ for all $i = 1, 2, \dots$, and all $j = 0, 1, \dots$, and $V_i^{\pi(\infty)} = V_i^\pi$ for all $i = 1, 2, \dots$. Therefore, the proof is complete.

Note that Lemma 6.2.2 presented the same result as Theorems 5.2.1(i) and 5.2.2. We state and prove it here using our new notation.

Lemma 6.2.3 (cf. Lemma 5.2.3) *Let Assumption 6.2.1 hold. Suppose that $V_0 \geq \mathcal{T}V_0$. Let π_i and $V_i^{\pi(j)}$ be updated by (6.2.20) and (6.2.21). Then, the sequence $\{V_i^\pi\}$ is a monotonically nonincreasing sequence, i.e., $V_i^\pi \geq V_{i+1}^\pi, \forall i \geq 0$.*

Proof According to Lemma 6.2.2, we can get

$$V_{i+1}^{\pi(0)} \geq V_{i+1}^{\pi(\infty)} = V_{i+1}^{\pi}.$$

Then, considering

$$V_i^{\pi} = \mathcal{T}_{\pi_{i-1}} V_i^{\pi} \geq \mathcal{T} V_i^{\pi} = \mathcal{T}_{\pi_i} V_i^{\pi} = \mathcal{T}_{\pi_i} V_{i+1}^{\pi(0)},$$

and Assumption 6.2.1, we can obtain

$$V_i^{\pi} \geq \mathcal{T}_{\pi_i} V_{i+1}^{\pi(0)} \geq \mathcal{T}_{\pi_i} V_{i+1}^{\pi} = V_{i+1}^{\pi}.$$

Therefore, the sequence $\{V_i^{\pi}\}$, $i \geq 0$, is a monotonically nonincreasing sequence. This completes the proof of the lemma.

Based on the lemmas above, an extended result of Proposition 5 in [23] is given in the following theorem.

Theorem 6.2.3 *Let Assumptions 6.2.1 and 6.2.2 hold. Suppose that $V^* \leq V_0 \leq \beta V^*$, $1 \leq \beta < \infty$, and that $V_0 \geq \mathcal{T} V_0$. Let π_i and $V_i^{\pi(j)}$ be updated by (6.2.20) and (6.2.21). Then, the value function sequence $\{V_i^{\pi}\}$ approaches V^* according to the following inequalities*

$$V^* \leq V_i^{\pi} \leq \left[1 + \frac{\beta - 1}{(1 + \lambda^{-1})^i} \right] V^*, \quad i \geq 1.$$

Proof First, we prove that $V_i^{\pi} \leq V_i$, $\forall i \geq 1$, holds by induction, where V_i is defined in (6.2.7). According to Lemma 6.2.2 and (6.2.22), we have

$$V_1^{\pi} = \mathcal{T}_{\pi_0} V_1^{\pi} = \mathcal{T}_{\pi_0} V_1^{\pi(\infty)} \leq \mathcal{T}_{\pi_0} V_1^{\pi(0)} = \mathcal{T}_{\pi_0} V_0 = \mathcal{T} V_0 = V_1.$$

Assume $V_i^{\pi} \leq V_i$ holds, $\forall i \geq 1$. Considering Assumption 6.2.1 and Lemma 6.2.2, we have

$$V_{i+1}^{\pi} = \mathcal{T}_{\pi_i} V_{i+1}^{\pi} = \mathcal{T}_{\pi_i} V_{i+1}^{\pi(\infty)} \leq \mathcal{T}_{\pi_i} V_{i+1}^{\pi(0)} \leq \mathcal{T} V_{i+1}^{\pi(0)} = \mathcal{T} V_{i+1}^{\pi} \leq \mathcal{T} V_i = V_{i+1}.$$

Therefore, considering (6.2.3) and Theorem 6.2.1, we can obtain the conclusion. This completes the proof of the theorem.

B. Error Bounds for Approximate Policy Iteration

Similar to Sect. 6.2.2B, we use function approximation structures to approximate the value function and control policy. Here, we use $\hat{V}_i^{\hat{\pi}}$ and $\hat{\pi}_i$ to stand for the approximate expressions of V_{π_i} and π_i , respectively. We assume that there exist finite positive constants $\underline{\delta} \leq 1$ and $\bar{\delta} \geq 1$ such that

$$\underline{\delta} V_i^{\hat{\pi}} \leq \hat{V}_i^{\hat{\pi}} \leq \bar{\delta} V_i^{\hat{\pi}}, \quad \forall i = 1, 2, \dots, \quad (6.2.27)$$

hold uniformly, where $V_i^{\hat{\pi}}$ is the exact value function associated with $\hat{\pi}_i$. Considering Lemma 6.2.2, we have

$$\hat{V}_i^{\hat{\pi}} \leq \bar{\delta} V_i^{\hat{\pi}} \leq \bar{\delta} V_i^{\hat{\pi}(1)} = \bar{\delta} \mathcal{T}_{\hat{\pi}_i} V_i^{\hat{\pi}(0)} = \bar{\delta} \mathcal{T}_{\hat{\pi}_i} \hat{V}_{i-1}^{\hat{\pi}}. \quad (6.2.28)$$

Similarly, we assume that there exist finite positive constants $\underline{\zeta} \leq 1$ and $\bar{\zeta} \geq 1$ such that

$$\underline{\zeta} \mathcal{T} \hat{V}_{i-1}^{\hat{\pi}} \leq \mathcal{T}_{\hat{\pi}_i} \hat{V}_{i-1}^{\hat{\pi}} \leq \bar{\zeta} \mathcal{T} \hat{V}_{i-1}^{\hat{\pi}}, \quad \forall i = 1, 2, \dots, \quad (6.2.29)$$

hold uniformly. Combining (6.2.28) and (6.2.29), we can get

$$\hat{V}_i^{\hat{\pi}} \leq \bar{\zeta} \bar{\delta} \mathcal{T} \hat{V}_{i-1}^{\hat{\pi}}.$$

On the other hand, considering (6.2.27), (6.2.29), and Assumption 6.2.1, we can obtain

$$\begin{aligned} \hat{V}_i^{\hat{\pi}} &\geq \underline{\delta} V_i^{\hat{\pi}} \\ &= \underline{\delta} (\mathcal{T}_{\hat{\pi}_i} \cdots \mathcal{T}_{\hat{\pi}_1} \mathcal{T}_{\hat{\pi}_i}) \hat{V}_{i-1}^{\hat{\pi}} \\ &\geq \underline{\delta} (\mathcal{T} \cdots \mathcal{T} \mathcal{T}_{\hat{\pi}_i}) \hat{V}_{i-1}^{\hat{\pi}} \\ &\geq \underline{\zeta} \underline{\delta} (\mathcal{T} \cdots \mathcal{T} \mathcal{T}) \hat{V}_{i-1}^{\hat{\pi}} \\ &\geq \underline{\zeta} \underline{\delta} V^*. \end{aligned}$$

Therefore, the whole approximation errors in the value function and control policy update equations can be expressed by

$$\underline{\zeta} \underline{\delta} V^* \leq \hat{V}_i^{\hat{\pi}} \leq \bar{\zeta} \bar{\delta} \mathcal{T} \hat{V}_{i-1}^{\hat{\pi}}. \quad (6.2.30)$$

Using the notation in (6.2.13), (6.2.30) can be written as

$$\underline{\eta} V^* \leq \hat{V}_i^{\hat{\pi}} \leq \bar{\eta} \mathcal{T} \hat{V}_{i-1}^{\hat{\pi}}. \quad (6.2.31)$$

Similar to Sect. 6.2.2B, we can establish the error bounds for approximate policy iteration by the following theorem.

Theorem 6.2.4 *Let Assumptions 6.2.1 and 6.2.2 hold. Suppose that $V^* \leq V_0 \leq \beta V^*$, $1 \leq \beta < \infty$, and that $V_0 \geq \mathcal{T} V_0$. The approximate value function $\hat{V}_i^{\hat{\pi}}$ satisfies the iterative error condition (6.2.31). Then, the value function sequence $\{\hat{V}_i^{\hat{\pi}}\}$ approaches V^* according to the following inequalities*

$$\underline{\eta} V^* \leq \hat{V}_{i+1}^{\hat{\pi}} \leq \bar{\eta} \left[1 + \sum_{j=1}^i \frac{\lambda^j \bar{\eta}^{j-1} (\bar{\eta} - 1)}{(\lambda + 1)^j} + \frac{\lambda^i \bar{\eta}^i (\beta - 1)}{(\lambda + 1)^i} \right] V^*, \quad i \geq 0.$$

Moreover, the approximate value function sequence $\{\hat{V}_i^\pi\}$ converges to a finite neighborhood of V^* uniformly on Ω as $i \rightarrow \infty$, i.e.,

$$\underline{\eta} V^* \leq \lim_{i \rightarrow \infty} \hat{V}_i^{\hat{\pi}} \leq \frac{\bar{\eta}}{1 + \lambda - \lambda \bar{\eta}} V^*, \quad (6.2.32)$$

under the condition $\bar{\eta} < \frac{1}{\lambda} + 1$.

6.2.4 Approximate Optimistic Policy Iteration

We will prove the convergence of exact optimistic policy iteration and establish the error bounds for approximate optimistic policy iteration.

A. Optimistic Policy Iteration

According to Lemma 6.2.2, we can see that the policy evaluation can be obtained as $j \rightarrow \infty$ in (6.2.21). However, this process will usually take a long time to converge. To avoid this problem, the optimistic policy iteration algorithm only performs finite number of iterations in (6.2.21).

For the optimistic policy iteration algorithm, we start the iteration from an initial value function V_0 such that $V_0 \geq \mathcal{T}V_0$. The initial control policy $\mu_0 = \pi_0$ is obtained by (6.2.18). For $i = 1, 2, \dots$, and for some positive integer N_i , the optimistic policy iteration algorithm updates the value function by

$$V_i^\mu(x) = V_i^{\mu(N_i)}(x), \quad (6.2.33)$$

where

$$V_i^{\mu(j)}(x) = U(x, \mu_{i-1}(x)) + V_i^{\mu(j-1)}(F(x, \mu_{i-1}(x))), \quad 0 < j \leq N_i, \quad (6.2.34)$$

$V_i^{\mu(0)} = V_{i-1}^\mu$, $\forall i \geq 1$, and $V_1^{\mu(0)} = V_0^\mu = V_0$. Using the definition in (6.2.5), the value function $V_i^\mu(x)$ can be expressed by $V_i^\mu(x) = \mathcal{T}_{\mu_{i-1}}^{N_i} V_{i-1}^\mu(x)$. The optimistic policy iteration algorithm updates the control policy by

$$\mu_i(x) = \arg \min_u \{U(x, u) + V_i^\mu(F(x, u))\}. \quad (6.2.35)$$

In this case, the optimistic policy iteration algorithm becomes the generalized policy iteration algorithm studied in Chap. 4. The above algorithm becomes value iteration as $N_i = 1$ and becomes the policy iteration as $N_i \rightarrow \infty$. For policy iteration, it solves the value function associated with the current control policy at each iteration, while it takes only one iteration toward that value function for value

Table 6.4 Key expressions used in Sect. 6.2.4

Key expression	Location
$V \leq V' \Rightarrow \mathcal{T}_u V \leq \mathcal{T}_u V'$	Assumption 6.2.1
$V_i^\mu(x) = V_i^{\mu(N_i)}(x) = \mathcal{T}_{\mu_{i-1}} V_i^{\mu(N_i-1)}$	(6.2.33)
$V_i^{\mu(j)}(x) = \mathcal{T}_{\mu_{i-1}} V_i^{\mu(j-1)}$	(6.2.34)
$V_i^{\mu(0)} = V_{i-1}^\mu, V_1^{\mu(0)} = V_0^\mu = V_0$	(6.2.34)
$\mathcal{T}V_i^\mu = \mathcal{T}_{\mu_i} V_i^\mu$	(6.2.35)

iteration. However, the value function update in (6.2.34) has to stop before $j \rightarrow \infty$ in practical implementations.

We list some key expressions in Table 6.4 which will be used in this section for establishing analysis results.

Next, we will show the monotonicity property of value function which is given in [6] and then establish the convergence property of optimistic policy iteration.

Lemma 6.2.4 *Let Assumption 6.2.1 hold. Suppose that $V_0 \geq \mathcal{T}V_0$. Let V_i^μ and μ_i be updated by (6.2.34) and (6.2.35). Then, the value function sequence $\{V_i^\mu\}$ is a monotonically nonincreasing sequence, i.e., $V_i^\mu \geq V_{i+1}^\mu, \forall i \geq 0$.*

Proof According to Assumption 6.2.1 and Lemma 6.2.2, for $i = 0$, we have

$$V_0^\mu = V_0 \geq \mathcal{T}V_0 = \mathcal{T}_{\mu_0} V_0 = \mathcal{T}_{\mu_0} V_1^{\mu(0)} \geq \mathcal{T}_{\mu_0} V_1^{\mu(N_1-1)} = V_1^\mu.$$

Thus, it holds for $i = 0$. Similarly, for $i = 1$, we can get

$$V_1^\mu = \mathcal{T}_{\mu_0} V_1^{\mu(N_1-1)} \geq \mathcal{T}_{\mu_0} V_1^{\mu(N_1)} = \mathcal{T}_{\mu_0} V_1^\mu \geq \mathcal{T}V_1^\mu \quad (6.2.36)$$

and

$$\mathcal{T}V_1^\mu = \mathcal{T}_{\mu_1} V_1^\mu = \mathcal{T}_{\mu_1} V_2^{\mu(0)} \geq \mathcal{T}_{\mu_1} V_2^\mu = V_2^\mu.$$

Therefore, the conclusion can be proved by induction.

Theorem 6.2.5 *Let Assumptions 6.2.1 and 6.2.2 hold. Suppose that $V^* \leq V_0 \leq \beta V^*$, $1 \leq \beta < \infty$, and that $V_0 \geq \mathcal{T}V_0$. The value function V_i^μ and the control policy μ_i are updated by (6.2.34) and (6.2.35). Then, the value function sequence $\{V_i^\mu\}$ approaches V^* according to the inequalities*

$$V^* \leq V_i^\mu \leq \left[1 + \frac{\beta - 1}{(1 + \lambda^{-1})^i} \right] V^*, i \geq 1. \quad (6.2.37)$$

Moreover, the value function V_i^μ converges to V^* uniformly on Ω .

Proof First, we prove that $V_i^\mu \leq V_i, \forall i \geq 1$, holds by mathematical induction, where V_i is defined in (6.2.7). According to Lemma 6.2.2, we have

$$V_1^\mu = V_1^{\mu(N_1)} \leq V_i^{\mu(1)} = \mathcal{T}_{\mu_0} V_0 = \mathcal{T} V_0 = V_1.$$

Thus, (6.2.37) holds for $i = 1$. Assume that it holds for $i \geq 1$, i.e., $V_i^\mu \leq V_i$. According to Lemma 6.2.2, we have

$$V_{i+1}^\mu = V_{i+1}^{\mu(N_i)} \leq V_{i+1}^{\mu(1)} = \mathcal{T}_{\mu_i} V_{i+1}^{\mu(0)} = \mathcal{T}_{\mu_i} V_i^\mu = \mathcal{T} V_i^\mu.$$

Considering Assumption 6.2.1, we can get

$$\mathcal{T} V_i^\mu \leq \mathcal{T} V_i = V_{i+1}.$$

Thus, we can obtain $V_{i+1}^\mu \leq V_{i+1}$. Then, it can also be proved that $V_i^\mu \geq V^*, \forall i \geq 1$, by mathematical induction. Therefore, considering Theorem 6.2.1, we can obtain (6.2.37). As $i \rightarrow \infty$, the value function V_i^μ converges to V^* uniformly on Ω . The proof is complete.

B. Error Bounds for Approximate Optimistic Policy Iteration

Here, we use $\hat{V}_i^{\hat{\mu}}$ and $\hat{\mu}_i$ to stand for the approximate expressions of V_i^μ and μ_i , respectively. Without loss of generality, we let N_i be a constant integer K for all iteration steps.

We assume that there exist finite positive constants $\underline{\delta} \leq 1$ and $\bar{\delta} \geq 1$ such that

$$\underline{\delta} \mathcal{T}_{\hat{\mu}_i}^K \hat{V}_{i-1}^{\hat{\mu}} \leq \hat{V}_i^{\hat{\mu}} \leq \bar{\delta} \mathcal{T}_{\hat{\mu}_i}^K \hat{V}_{i-1}^{\hat{\mu}}, \quad \forall i = 1, 2, \dots, \quad (6.2.38)$$

hold uniformly. Considering Lemma 6.2.2, we have

$$\hat{V}_i^{\hat{\mu}} \leq \bar{\delta} \mathcal{T}_{\hat{\mu}_i}^K \hat{V}_{i-1}^{\hat{\mu}} \leq \bar{\delta} \mathcal{T}_{\hat{\mu}_i} \hat{V}_{i-1}^{\hat{\mu}}. \quad (6.2.39)$$

Similarly, we assume that there exist finite positive constants $\underline{\zeta} \leq 1$ and $\bar{\zeta} \geq 1$ such that

$$\underline{\zeta} \mathcal{T} \hat{V}_{i-1}^{\hat{\mu}} \leq \mathcal{T}_{\hat{\mu}_i} \hat{V}_{i-1}^{\hat{\mu}} \leq \bar{\zeta} \mathcal{T} \hat{V}_{i-1}^{\hat{\mu}}, \quad \forall i = 1, 2, \dots, \quad (6.2.40)$$

hold uniformly. Combining (6.2.39) and (6.2.40), we can get

$$\hat{V}_i^{\hat{\mu}} \leq \bar{\zeta} \bar{\delta} \mathcal{T} \hat{V}_{i-1}^{\hat{\mu}}.$$

On the other hand, considering (6.2.38), (6.2.40), and Assumption 6.2.1, we get

$$\hat{V}_i^{\hat{\mu}} \geq \underline{\delta} \mathcal{T}_{\hat{\mu}_i}^K \hat{V}_{i-1}^{\hat{\mu}} \geq \underline{\delta} \mathcal{T}(\mathcal{T}_{\hat{\mu}_i}^{(K-1)} \hat{V}_{i-1}^{\hat{\mu}}) \geq \cdots \geq \underline{\delta} \mathcal{T}^{(K-1)}(\mathcal{T}_{\hat{\mu}_i} \hat{V}_{i-1}^{\hat{\mu}}) \geq \underline{\zeta} \underline{\delta} \mathcal{T}^K \hat{V}_{i-1}^{\hat{\mu}}.$$

Therefore, the whole approximation errors in the value function and control policy update equations can be expressed by

$$\underline{\zeta} \underline{\delta} \mathcal{T}^K \hat{V}_{i-1}^{\hat{\mu}} \leq \hat{V}_i^{\hat{\mu}} \leq \bar{\zeta} \bar{\delta} \mathcal{T} \hat{V}_{i-1}^{\hat{\mu}}. \quad (6.2.41)$$

Using the notation in (6.2.13), (6.2.41) can be written as

$$\underline{\eta} \mathcal{T}^K \hat{V}_{i-1}^{\hat{\mu}} \leq \hat{V}_i^{\hat{\mu}} \leq \bar{\eta} \mathcal{T} \hat{V}_{i-1}^{\hat{\mu}}. \quad (6.2.42)$$

Next, we can establish the error bounds for the approximate optimistic policy iteration by the following theorem.

Theorem 6.2.6 *Let Assumptions 6.2.1 and 6.2.2 hold. Suppose that $V^* \leq V_0 \leq \beta V^*$, $1 \leq \beta < \infty$, and that $V_0 \geq \mathcal{T}V_0$. The approximate value function $\hat{V}_i^{\hat{\mu}}$ satisfies the iterative error condition (6.2.42). Then, the value function sequence $\{\hat{V}_i^{\hat{\mu}}\}$ approaches V^* according to the following inequalities*

$$\begin{aligned} & \underline{\eta} \left[1 - \sum_{j=1}^i \frac{\lambda^j \underline{\eta}^{j-1} (1 - \underline{\eta})}{(\lambda + 1)^j} \right] V^* \\ & \leq \hat{V}_{i+1}^{\hat{\mu}} \\ & \leq \bar{\eta} \left[1 + \sum_{j=1}^i \frac{\lambda^j \bar{\eta}^{j-1} (\bar{\eta} - 1)}{(\lambda + 1)^j} + \frac{\lambda^i \bar{\eta}^i (\beta - 1)}{(\lambda + 1)^i} \right] V^*, \quad i \geq 0. \end{aligned} \quad (6.2.43)$$

Moreover, the approximate value function sequence $\{\hat{V}_i^{\hat{\mu}}\}$ converges to a finite neighborhood of V^* uniformly on Ω as $i \rightarrow \infty$, i.e.,

$$\frac{\underline{\eta}}{1 + \lambda - \lambda \underline{\eta}} V^* \leq \lim_{i \rightarrow \infty} \hat{V}_i^{\hat{\mu}} \leq \frac{\bar{\eta}}{1 + \lambda - \lambda \bar{\eta}} V^*, \quad (6.2.44)$$

under the condition $\bar{\eta} < \frac{1}{\lambda} + 1$.

Proof First, we prove the lower bound in (6.2.43). According to (6.2.42) and Assumption 6.2.1, we have

$$\hat{V}_1^{\hat{\mu}} \geq \underline{\eta} \mathcal{T}^K V_0 \geq \underline{\eta} \mathcal{T}^K V^* = \underline{\eta} V^*,$$

i.e., it holds for $i = 0$. Similarly, we can get

$$\begin{aligned}
\hat{V}_2^{\hat{\mu}} &\geq \underline{\eta} \mathcal{P}^K \hat{V}_1^{\hat{\mu}} \\
&= \underline{\eta} \min_u \{U(x, u) + \mathcal{P}^{K-1} \hat{V}_1^{\hat{\mu}}(F(x, u))\} \\
&\geq \underline{\eta} \min_u \{U(x, u) + \underline{\eta} \mathcal{P}^{K-1} V^*(F(x, u))\} \\
&= \underline{\eta} \min_u \{U(x, u) + \underline{\eta} V^*(F(x, u))\} \\
&\geq \underline{\eta} \min_u \left\{ \left(1 - \lambda \frac{1 - \underline{\eta}}{\lambda + 1}\right) U(x, u) + \left(\underline{\eta} + \frac{1 - \underline{\eta}}{\lambda + 1}\right) V^*(F(x, u)) \right\} \\
&= \underline{\eta} \left(1 - \frac{\lambda(1 - \underline{\eta})}{\lambda + 1}\right) V^*,
\end{aligned}$$

i.e., it holds for $i = 1$. Therefore, we can obtain the lower bound by continuing the above process. Similar to Theorem 6.2.2, we can obtain the upper bound in (6.2.43) and the conclusion in (6.2.44). The proof is complete.

6.2.5 Neural Network Implementation

We have just proven that the approximate value iteration, approximate policy iteration, and approximate optimistic policy iteration algorithms can converge to a finite neighborhood of the optimal value function associated with the Bellman equation. It should be mentioned that we consider approximation errors in both value function and control policy update equations at each iteration. This makes it feasible to use neural network approximation for solving undiscounted optimal control problems of nonlinear systems. Since the optimistic policy iteration contains the value iteration and policy iteration, we only present a detailed implementation of the approximate optimistic policy iteration using neural networks in this section. The neural network implementation of approximate value iteration has been discussed in previous chapters [15, 17].

The whole structural diagram of the approximate optimistic policy iteration is shown in Fig. 6.1 (cf. (6.2.34)), where two multilayer feedforward neural networks are used. The critic neural network is used to approximate the value function, and the action neural network is used to approximate the control policy.

A neural network can be used to approximate some smooth function on a prescribed compact set. The value function $V_i^{\mu(j)}(x_k)$ in (6.2.34) is approximated by the critic neural network

$$\hat{V}_i^{\hat{\mu}(j)}(x_k) = (W_{c(i)}^j)^\top \phi((Y_{c(i)}^j)^\top x_k),$$

where the activation functions are selected as $\tanh(\cdot)$. The target function of the critic neural network training is given by

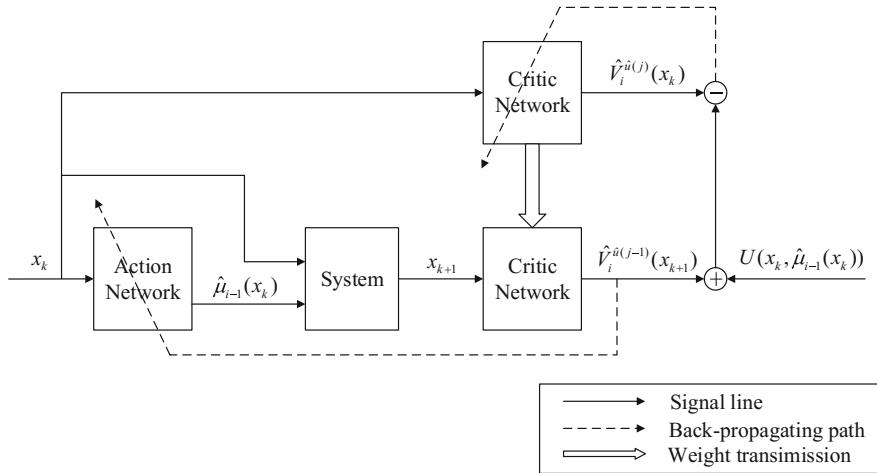


Fig. 6.1 Structural diagram of approximate optimistic policy iteration

$$V_i^{\hat{\mu}(j)}(x_k) = U(x_k, \hat{\mu}_{i-1}(x_k)) + \hat{V}_i^{\hat{\mu}(j-1)}(x_{k+1}),$$

where

$$x_{k+1} = F(x_k, \hat{\mu}_{i-1}(x_k)).$$

Then, the error function for training critic neural network is defined by

$$e_{c(i)}^j(x_k) = V_i^{\hat{\mu}(j)}(x_k) - \hat{V}_i^{\hat{\mu}(j)}(x_k),$$

and the performance function to be minimized is defined by

$$E_{c(i)}^j(x_k) = \frac{1}{2}(e_{c(i)}^j(x_k))^2. \quad (6.2.45)$$

The control policy $\mu_i(x_k)$ in (6.2.35) is approximated by the action neural network

$$\hat{\mu}_i(x_k) = W_{a(i)}^T \phi(Y_{a(i)}^T x_k).$$

The target function of the action neural network training is defined by

$$d_i(x_k) = \arg \min_{u_k} \{U(x_k, u_k) + \hat{V}_i^{\hat{\mu}}(x_{k+1})\}.$$

Then, the error function for training the action neural network is given by

$$e_{a(i)}(x_k) = d_i(x_k) - \hat{\mu}_i(x_k).$$

The weights of the action neural network are updated to minimize the following performance function

$$E_{a(i)}(x_k) = \frac{1}{2} (e_{a(i)}(x_k))^T e_{a(i)}(x_k). \quad (6.2.46)$$

We use the gradient descent method to tune the weights of neural networks on a training set constructed from the compact set Ω . The details of this tuning method can be found in [15]. Some other tuning methods can also be used, such as Newton's method and the Levenberg–Marquardt method, in order to increase the convergence rate of neural network training.

A detailed process of the approximate optimistic policy iteration is given in Algorithm 6.2.1, where the approximate value iteration can be regarded as a special case. If we have an initial stabilizing control policy, the algorithm can iterate between Steps 4 and 5 directly. It should be mentioned that Algorithm 6.2.1 runs in an off-line manner. Note that it can also be implemented online but a persistence of excitation condition is usually required.

Algorithm 6.2.1 Approximate optimistic policy iteration

Step 1. Initialization:

 Initialize critic and action neural networks.

 Select an initial value function V_0 satisfying $V_0 \geq \mathcal{V}V_0$.

 Set policy evaluation steps K and maximum number of iteration steps i_{\max} .

Step 2. Update the control policy $\hat{\mu}_0(x_k)$ by minimizing (6.2.46) on a training set $\{x_k\}$ randomly selected from the compact set Ω .

Step 3. Set $i = 1$.

Step 4. For $j = 0, 1, \dots, K - 1$, update the value function $\hat{V}_i^{\hat{\mu}(j)}(x_k)$ by minimizing (6.2.45) on a training set $\{x_k\}$ randomly selected from the compact set Ω . After convergence, set $\hat{V}_i^{\hat{\mu}}(x_k) = \hat{V}_i^{\hat{\mu}(K)}(x_k)$.

Step 5. Update the control policy $\hat{\mu}_i(x_k)$ by minimizing (6.2.46) on a training set $\{x_k\}$ randomly selected from the compact set Ω .

Step 6. Set $i \leftarrow i + 1$.

Step 7. Repeat Steps 4–6 until the convergence conditions are met or $i > i_{\max}$.

Step 8. Obtain the approximate optimal control policy $\hat{\mu}_i(x_k)$ or the optimal control policy is not obtained if $i > i_{\max}$.

6.2.6 Simulation Study

In this section, we provide a simulation example to demonstrate the effectiveness of the present algorithms. Consider the following discrete-time nonlinear system $x_{k+1} = h(x_k) + g(x_k)u_k$, where

$$h(x_k) = \begin{bmatrix} 0.9x_{1k} + 0.1x_{2k} \\ -0.05(x_{1k} + x_{2k}(1 - (\cos(2x_{1k}) + 2)^2)) + x_{2k} \end{bmatrix},$$

$$g(x_k) = \begin{bmatrix} 0 \\ 0.1 \cos(2x_{1k}) + 0.2 \end{bmatrix}, \quad (6.2.47)$$

$x_k = [x_{1k}, x_{2k}]^\top \in \mathbb{R}^2$, and $u_k \in \mathbb{R}$, $k = 0, 1, \dots$. Define the cost function as

$$J(x_0, u) = \sum_{k=0}^{\infty} (x_k^\top Q x_k + u_k^\top R u_k), \quad (6.2.48)$$

where $Q = [0.1, 0; 0, 0.1]$ and $R = 0.1$.

To implement the present algorithms, we choose three-layer feedforward neural networks as function approximation structures. The structures of the critic and action neural networks are both chosen as 2–8–1. The initial weights of the critic and action neural networks are chosen randomly in $[-0.1, 0.1]$. The maximum number of iteration steps is selected as $i_{\max} = 20$. The compact set Ω or the operation region of the system is selected as $-1 \leq x_1 \leq 1$ and $-1 \leq x_2 \leq 1$. The training set $\{x_k\}$ is constructed by randomly choosing 1000 samples from the compact set Ω at each iteration.

The initial value function is selected as $V_0 = 2x_{1k}^2 + 2x_{2k}^2$. According to Fig. 6.2, it can be seen that $V_0 \geq V_1$ holds for all states in the compact set Ω .

We implement Algorithm 6.2.1 by letting $K = 1$, $K = 3$, and $K = 10$, respectively. For the initial state $x_0 = [1, -1]^\top$, the convergence curve of the value function sequence $\{\hat{V}_{\hat{\mu}_1}^j\}$ is shown in Fig. 6.3. We can see that $\{\hat{V}_{\hat{\mu}_1}^j\}$ is a monotonically non-increasing sequence, and it is convergent at $j = 10$. Thus, the algorithm for $K = 10$ can be regarded as the approximate policy iteration. The algorithms for $K = 1$ and $K = 3$ are the approximate value iteration and approximate optimistic policy iter-

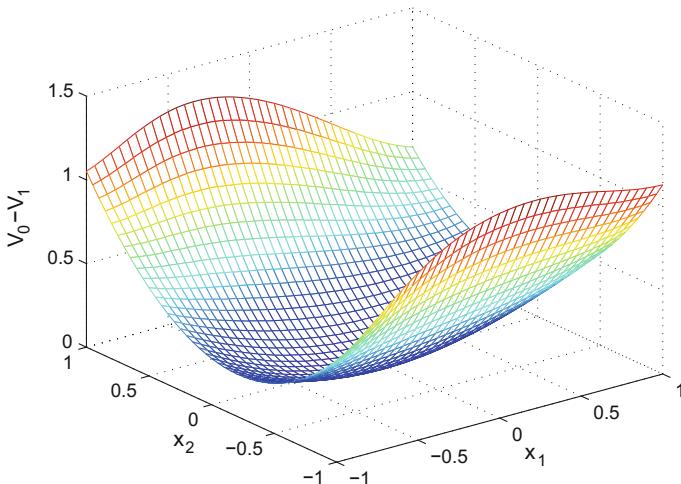


Fig. 6.2 3-D plot of $V_0 - V_1$ in the compact set Ω

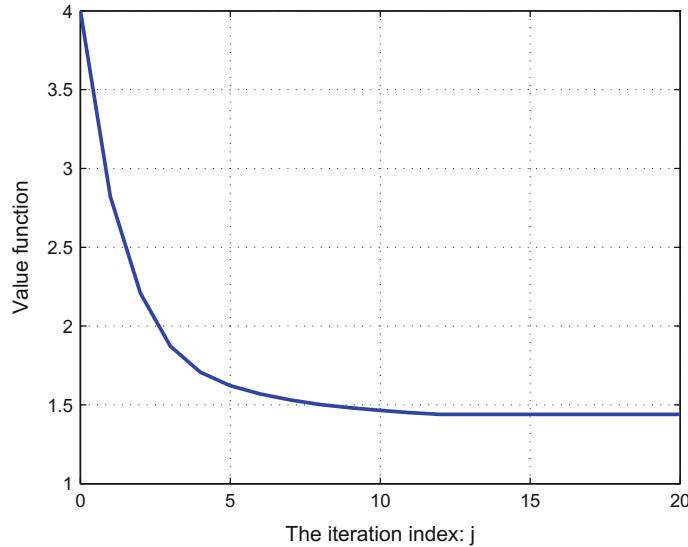


Fig. 6.3 The convergence curve of the value function $\hat{V}_1^{\hat{\mu}(j)}$ at x_0

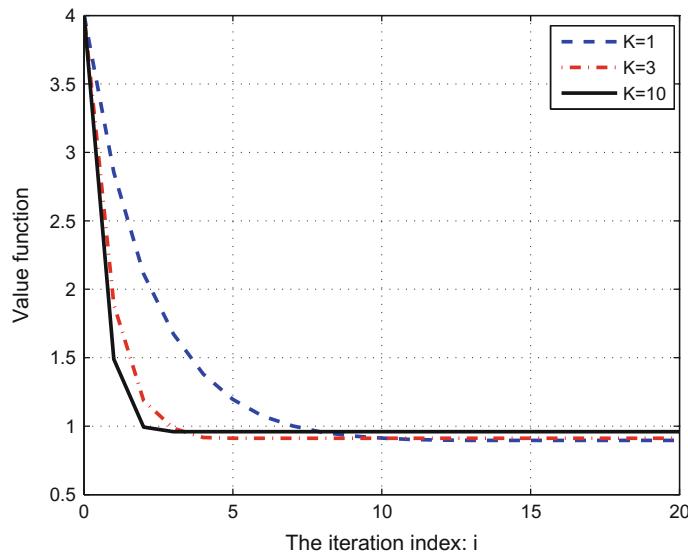


Fig. 6.4 The convergence curves of the value function $\hat{V}_1^{\hat{\mu}}$ at x_0 when $K = 1$, $K = 3$, and $K = 10$

tion, respectively. After implementing the algorithms for $i_{\max} = 20$, the convergence curves of the value functions $\hat{V}_1^{\hat{\mu}}$ at x_0 are shown in Fig. 6.4. It can be seen that all the value functions are basically convergent with the iteration index $i > 10$, and the obtained approximate optimal value functions at $i = 20$ are quite close. Although

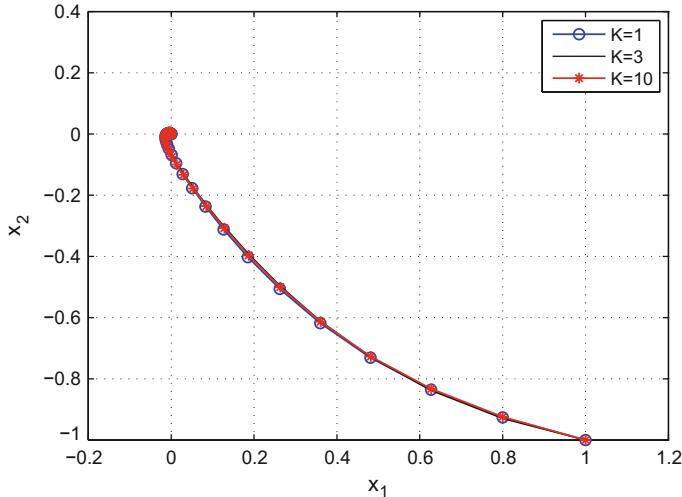


Fig. 6.5 The state trajectories

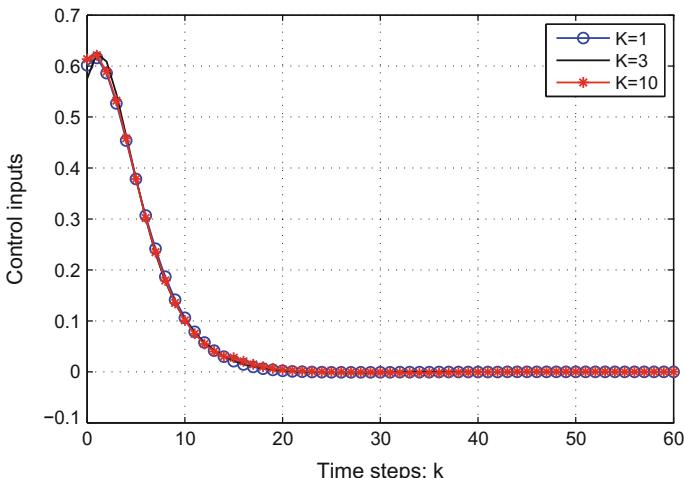


Fig. 6.6 The control inputs

there exist approximation errors in both value function and control policy update steps, the approximate value function can converge to a finite neighborhood of the optimal value function.

Finally, we apply the approximate optimal control policies obtained above to the system (6.2.47) for 60 time steps. The corresponding state trajectories are displayed in Fig. 6.5, and the control inputs are displayed in Fig. 6.6. Examining the results,

it is observed that all the control policies obtain very good performance, and the differences between the three trajectories are quite small.

6.3 Error Bounds of Q-Function for Discounted Optimal Control Problems

6.3.1 Problem Formulation

In this section, we consider the following deterministic discrete-time nonlinear dynamical systems

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \dots, \quad (6.3.1)$$

where $x_k \in \mathbb{R}^n$ is the state vector, and $u_k \in \mathbb{R}^m$ is the control input. The system (6.3.1) is assumed to be controllable which implies that there exists a continuous control policy on a compact set $\mathcal{Q} \subseteq \mathbb{R}^n$ that stabilizes the system asymptotically. We assume that $x_k = 0$ is an equilibrium state of the system (6.3.1) and $F(0, 0) = 0$. The system function $F(x_k, u_k)$ is Lipschitz continuous with respect to x_k and u_k . The infinite-horizon cost function with discount factor for any initial state x_0 is given by

$$J(x_0, \underline{u}_0) = \sum_{k=0}^{\infty} \gamma^k U(x_k, u_k), \quad (6.3.2)$$

where \underline{u}_0 is the control sequence defined as $\underline{u}_0 = (u_0, u_1, u_2, \dots)$, U is the utility function which is continuous and positive definite, and γ is the discount factor such that $0 < \gamma \leq 1$. For any feedback control policy $\mu(\cdot)$, the value function $V^\mu(x_k)$, i.e., the mapping from any state x_k to the value of (6.3.2), is defined as

$$V^\mu(x_k) = J(x_k, \mu(x_k)) = J(x_k, \underline{u}_k), \quad (6.3.3)$$

where $\underline{u}_k = (\mu(x_k), \mu(x_{k+1}), \dots)$. A feedback control policy $u = \mu(x)$ is admissible with respect to system (6.3.1), if $\mu(0) = 0$, $\mu(x)$ is continuous and stabilizes the system, and the value function in (6.3.3) is finite.

Our goal is to find an admissible control which can minimize the value function such that

$$V^*(x_k) = \inf_{\mu} \{V^\mu(x_k)\}, \quad \forall x_k. \quad (6.3.4)$$

According to Bellman's principle of optimality, the optimal value function satisfies the Bellman equation

$$V^*(x_k) = \min_{u_k} \{U(x_k, u_k) + \gamma V^*(F(x_k, u_k))\}, \quad (6.3.5)$$

and the optimal control policy $\mu^*(\cdot)$ is given by

$$\mu^*(x_k) = u_k^* = \arg \min_{u_k} \{U(x_k, u_k) + \gamma V^*(F(x_k, u_k))\}. \quad (6.3.6)$$

In the case of action-dependent ADP schemes, which will be studied in this section, Q-function (also known as the state-action value function) will be used and it is defined as

$$Q^\mu(x, u) = U(x, u) + \gamma Q^\mu(x^+, \mu(x^+)), \quad (6.3.7)$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and x^+ is the state of the next moment, i.e., $x^+ = F(x, u)$. According to (6.3.3), the relationship between value function and Q-function is

$$\begin{aligned} Q^\mu(x, \mu(x)) &= U(x, \mu(x)) + \gamma Q^\mu(x^+, \mu(x^+)) \\ &= J(x, \mu(x)) \\ &= V^\mu(x). \end{aligned}$$

Then, the optimal Q-function is defined as

$$Q^*(x, u) = \inf_{\mu} Q^\mu(x, u),$$

and the optimal control policy $\mu^*(\cdot)$ can be obtained by

$$\mu^*(x_k) = \arg \min_{u_k} Q^*(x_k, u_k), \quad \forall x_k. \quad (6.3.8)$$

The optimal Q-function satisfies the following Bellman optimality equation

$$Q^*(x_k, u_k) = U(x_k, u_k) + \gamma \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}), \quad (6.3.9)$$

where u_{k+1} is the action of the next moment. The connection between the optimal value function and the optimal Q-function is

$$V^*(x) = \min_u Q^*(x, u).$$

In most situations, the optimal control problem for nonlinear systems has no analytical solution and the traditional dynamic programming faces the “curse of dimensionality.” In this section, we develop an iterative ADP algorithm by using Q-function which depends on the state and action to solve the nonlinear optimal control problem. Similar to most ADP methods, function approximation structures such as neural networks are used to approximate the Q-function (or state-action value function) and the control policy. The approximation errors may increase along with the iteration processes. Therefore, it is necessary to establish the error bounds of

Q-function for the present iteration algorithm considering the function approximation errors.

6.3.2 Policy Iteration Under Ideal Conditions

We assume that the nonlinear dynamical system (6.3.1) is unknown, and only an off-line data set $\{x_k, u_k, x_{k+1}\}_N$ is available, where x_{k+1} is the next state given x_k and u_k , and N is the number of samples in the data set. x_{k+1} and x_k stand for the dynamic behavior of one-shot data and do not necessarily mean that the data set has to take samples from one trajectory. In general, the data set contains a variety of trajectories and scattered data.

For the policy iteration of action-dependent ADP algorithms, it starts with an initial admissible control μ_0 . For $i = 0, 1, \dots$, the policy iteration algorithm contains policy evaluation phase and policy improvement phase given as follows.

Policy evaluation:

$$Q_{j+1}^{\mu_i}(x_k, u_k) = U(x_k, u_k) + \gamma Q_j^{\mu_i}(x_{k+1}, \mu_i(x_{k+1})) \quad (6.3.10)$$

Policy improvement:

$$\mu_{i+1}(x_k) = \arg \min_{u_k} Q_j^{\mu_i}(x_k, u_k) \quad (6.3.11)$$

where j is the policy evaluation index and i is the policy improvement index. $Q_j^{\mu_i}$ represents the j th evaluation for the i th control policy μ_i , and

$$Q_0^{\mu_i} = Q_{\infty}^{\mu_{i-1}}.$$

Let Q^{μ_i} denote the Q-function for μ_i . Next, we will prove that the limit of $Q_j^{\mu_i}$ as $j \rightarrow \infty$ exists and $Q_{\infty}^{\mu_i} = Q^{\mu_i}$.

Assumption 6.3.1 There exists a finite positive constant λ such that the condition

$$\min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \leq \lambda U(x_k, u_k)$$

holds uniformly on \mathcal{Q} for some $0 < \lambda < \infty$.

Remark 6.3.1 Assumption 6.3.1 is a basic assumption which ensures the convergence of ADP algorithms. For most nonlinear systems, it is easy to find a sufficiently large number λ to satisfy this assumption as $Q^*(\cdot)$ and $U(\cdot)$ are finite and positive.

Lemma 6.3.1 *Let Assumption 6.3.1 hold. Suppose that μ_0 is an admissible control policy and Q^{μ_0} is the Q-function of μ_0 . Let $Q_j^{\mu_i}$ and μ_i be updated by (6.3.10) and (6.3.11). Then we can obtain the following conclusions:*

(i) *the sequence $\{Q_j^{\mu_i}\}$ is monotonically nonincreasing, i.e., $Q_j^{\mu_i} \geq Q_{j+1}^{\mu_i}$, $\forall i \geq 1$.*

Moreover, as $j \rightarrow \infty$, the limit of $Q_j^{\mu_i}$ exists and is denoted by $Q_\infty^{\mu_i}$, and equals to Q^{μ_i} , $\forall i \geq 1$.

(ii) the sequence $\{Q^{\mu_i}\}$ is monotonically nonincreasing, i.e., $Q^{\mu_i} \geq Q^{\mu_{i+1}}$, $\forall i \geq 1$.

Proof (i) μ_0 is an admissible control policy. According to (6.3.10) and (6.3.11), we can obtain

$$\begin{aligned} Q_0^{\mu_1}(x_k, u_k) &= Q^{\mu_0}(x_k, u_k) \\ &= U(x_k, u_k) + \gamma Q^{\mu_0}(x_{k+1}, \mu_0(x_{k+1})) \\ &\geq U(x_k, u_k) + \gamma \min_{u_{k+1}} Q^{\mu_0}(x_{k+1}, u_{k+1}) \\ &= U(x_k, u_k) + \gamma Q^{\mu_0}(x_{k+1}, \mu_1(x_{k+1})) \\ &= Q_1^{\mu_1}(x_k, u_k). \end{aligned} \quad (6.3.12)$$

So $Q_j^{\mu_1} \geq Q_{j+1}^{\mu_1}$ holds for $j = 0$. If $Q_j^{\mu_1} \geq Q_{j+1}^{\mu_1}$ holds for $j = m - 1$, when $j = m$, we obtain

$$\begin{aligned} Q_m^{\mu_1}(x_k, u_k) &= U(x_k, u_k) + \gamma Q_{m-1}^{\mu_1}(x_{k+1}, \mu_1(x_{k+1})) \\ &\geq U(x_k, u_k) + \gamma Q_m^{\mu_1}(x_{k+1}, \mu_1(x_{k+1})) \\ &= Q_{m+1}^{\mu_1}(x_k, u_k). \end{aligned} \quad (6.3.13)$$

According to the mathematical induction, we can obtain that $Q_j^{\mu_i} \geq Q_{j+1}^{\mu_i}$ holds for $i = 1$. Since $\{Q_j^{\mu_1}\}$ is a monotonically nonincreasing sequence and $Q_j^{\mu_1} \geq 0$, the limit of $\{Q_j^{\mu_1}\}$ exists, which is denoted by $Q_\infty^{\mu_1}$, and $Q_j^{\mu_1} \geq Q_\infty^{\mu_1}$, $\forall j$. According to (6.3.10), we have

$$Q_{j+1}^{\mu_1}(x_k, u_k) \geq U(x_k, u_k) + \gamma Q_\infty^{\mu_1}(x_{k+1}, \mu_1(x_{k+1})), \quad j \geq 0. \quad (6.3.14)$$

Letting $j \rightarrow \infty$, we have $Q_\infty^{\mu_1}(x_k, u_k) \geq U(x_k, u_k) + \gamma Q_\infty^{\mu_1}(x_{k+1}, \mu_1(x_{k+1}))$. Similarly, we can obtain

$$Q_\infty^{\mu_1}(x_k, u_k) \leq U(x_k, u_k) + \gamma Q_j^{\mu_1}(x_{k+1}, \mu_1(x_{k+1})), \quad j \geq 0. \quad (6.3.15)$$

Letting $j \rightarrow \infty$, we have $Q_\infty^{\mu_1}(x_k, u_k) \leq U(x_k, u_k) + \gamma Q_\infty^{\mu_1}(x_{k+1}, \mu_1(x_{k+1}))$. Hence, we can obtain

$$Q_\infty^{\mu_1}(x_k, u_k) = U(x_k, u_k) + \gamma Q_\infty^{\mu_1}(x_{k+1}, \mu_1(x_{k+1})). \quad (6.3.16)$$

Thus, we can obtain that $Q_\infty^{\mu_i} = Q^{\mu_i}$ holds for $i = 1$. We assume that $Q_j^{\mu_i} \geq Q_{j+1}^{\mu_i}$ and $Q_\infty^{\mu_i} = Q^{\mu_i}$ hold for any $i = l$, $l \geq 1$. According to (6.3.10) and (6.3.11), we can obtain

$$\begin{aligned}
Q_0^{\mu_{l+1}}(x_k, u_k) &= Q^{\mu_l}(x_k, u_k) \\
&= U(x_k, u_k) + \gamma Q^{\mu_l}(x_{k+1}, \mu_l(x_{k+1})) \\
&\geq U(x_k, u_k) + \gamma \min_{u_{k+1}} Q^{\mu_l}(x_{k+1}, u_{k+1}) \\
&= U(x_k, u_k) + \gamma Q^{\mu_l}(x_{k+1}, \mu_{l+1}(x_{k+1})) \\
&= Q_1^{\mu_{l+1}}(x_k, u_k).
\end{aligned} \tag{6.3.17}$$

Considering (6.3.10) and (6.3.17), we have

$$\begin{aligned}
Q_1^{\mu_{l+1}}(x_k, u_k) &= U(x_k, u_k) + \gamma Q_0^{\mu_{l+1}}(x_{k+1}, \mu_l(x_{k+1})) \\
&\geq U(x_k, u_k) + \gamma Q_1^{\mu_{l+1}}(x_{k+1}, \mu_l(x_{k+1})) \\
&= Q_2^{\mu_{l+1}}(x_k, u_k).
\end{aligned} \tag{6.3.18}$$

Similarly, we can obtain that $Q_j^{\mu_i} \geq Q_{j+1}^{\mu_i}$ holds for $i = l + 1$ by induction and $Q_\infty^{\mu_i} = Q^{\mu_i}$. Therefore, this completes the proof by induction.

(ii) According to (i), we have

$$Q^{\mu_i} = Q_0^{\mu_{i+1}} \geq Q_\infty^{\mu_{i+1}} = Q^{\mu_{i+1}}. \tag{6.3.19}$$

Therefore, $\{Q^{\mu_i}\}$ is a monotonically nonincreasing sequence and the proof is complete.

Theorem 6.3.1 *Let Assumption 6.3.1 hold. Suppose that $Q^* \leq Q^{\mu_0} \leq \beta Q^*$, $1 \leq \beta \leq \infty$. μ_0 is an admissible control policy and Q^{μ_0} is the Q-function of μ_0 . Let $Q_j^{\mu_i}$ and μ_i be updated by (6.3.10) and (6.3.11). Then the Q-function sequence Q^{μ_i} approaches Q^* according to the inequalities*

$$Q^* \leq Q^{\mu_i} \leq \left[1 + \frac{\beta - 1}{(1 + \lambda^{-1})^i} \right] Q^*, \quad \forall i \geq 1. \tag{6.3.20}$$

Proof According to the definitions of Q^* and Q^{μ_i} , the left-hand side of the inequality (6.3.20) always holds for any $i \geq 1$. Next, we prove the right-hand side of (6.3.20) by induction. According to Assumption 6.3.1, we have

$$\gamma \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \leq \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \leq \lambda U(x_k, u_k). \tag{6.3.21}$$

Considering (6.3.10), (6.3.11), and (6.3.21), we can obtain

$$\begin{aligned}
Q_1^{\mu_1}(x_k, u_k) &= U(x_k, u_k) + \gamma \min_{u_{k+1}} Q_0^{\mu_1}(x_{k+1}, u_{k+1}) \\
&= U(x_k, u_k) + \gamma \min_{u_{k+1}} Q^{\mu_0}(x_{k+1}, u_{k+1}) \\
&\leq U(x_k, u_k) + \gamma \beta \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1})
\end{aligned}$$

$$\begin{aligned}
&\leq U(x_k, u_k) + \gamma \beta \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \\
&\quad + \frac{\beta - 1}{\lambda + 1} \left[\lambda U(x_k, u_k) - \gamma \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \right] \\
&= \left(1 + \lambda \frac{\beta - 1}{\lambda + 1} \right) U(x_k, u_k) \\
&\quad + \left(\beta + \frac{\beta - 1}{\lambda + 1} \right) \gamma \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \\
&= \left[1 + \frac{\beta - 1}{(1 + \lambda^{-1})} \right] Q^*(x_k, u_k).
\end{aligned} \tag{6.3.22}$$

According to Lemma 6.3.1, we have

$$Q^{\mu_1} = Q_1^{\mu_1} \leq Q_1^{\mu_1}(x_k, u_k) \leq \left[1 + \frac{\beta - 1}{(1 + \lambda^{-1})} \right] Q^*(x_k, u_k), \tag{6.3.23}$$

which shows that the right-hand side of (6.3.20) holds for $i = 1$. Assume that

$$Q^{\mu_i}(x_k, u_k) \leq \left[1 + \frac{\beta - 1}{(1 + \lambda^{-1})^i} \right] Q^*(x_k, u_k)$$

holds for any $i = l$, $l \geq 1$, then we can obtain

$$\begin{aligned}
Q^{\mu_{l+1}}(x_k, u_k) &\leq Q_1^{\mu_{l+1}}(x_k, u_k) \\
&= U(x_k, u_k) + \gamma \min_{u_{k+1}} Q^{\mu_l}(x_{k+1}, u_{k+1}) \\
&\leq U(x_k, u_k) + \gamma \left[1 + \frac{\beta - 1}{(1 + \lambda^{-1})^l} \right] \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \\
&\quad + \frac{\beta - 1}{(1 + \lambda^{-1})^{l+1}} [\lambda U(x_k, u_k) - \gamma \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1})] \\
&\leq \left[1 + \frac{\beta - 1}{(1 + \lambda^{-1})^{l+1}} \right] Q^*(x_k, u_k),
\end{aligned} \tag{6.3.24}$$

which shows that the right-hand side of (6.3.20) holds for $i = l + 1$. According to the mathematical induction, the right-hand side of (6.3.20) holds. The proof is complete.

In policy iteration, an initial admissible control policy is required, which is usually obtained by experience or trial. However, for most nonlinear systems, it is hard to obtain an admissible control policy, especially in action-dependent ADP for unknown systems. So we present a novel initial condition for policy iteration.

Lemma 6.3.2 *Let Assumption 6.3.1 hold. Suppose that there is a positive-definite function Q_0 satisfying $\gamma Q_0 \geq Q_1^{\mu_1}$ for any x_k, u_k . Let $Q_j^{\mu_1}$ and μ_1 be obtained by (6.3.10) and (6.3.11). Then $\mu_1(x)$ is an admissible control policy and $Q^{\mu_1} = Q_0^{\mu_1}$ is the Q -function of $\mu_1(x)$.*

Proof Considering (6.3.10) and (6.3.11), we have

$$\mu_1(x_k) = \arg \min_{u_k} Q_0(x_k, u_k), \quad (6.3.25)$$

and

$$Q_1^{\mu_1}(x_k, u_k) = U(x_k, u_k) + \gamma Q_0(x_{k+1}, \mu_1(x_{k+1})). \quad (6.3.26)$$

Using the assumption about Q_0 , we obtain

$$\begin{aligned} \gamma Q_0(x_k, \mu_1(x_k)) &\geq Q_1^{\mu_1}(x_k, \mu_1(x_k)) \\ &= U(x_k, \mu_1(x_k)) + \gamma Q_0(x_{k+1}, \mu_1(x_{k+1})). \end{aligned} \quad (6.3.27)$$

Considering $Q_0(x_k, \mu_1(x_k)) \geq 0$ and

$$Q_0(x_{k+1}, \mu_1(x_{k+1})) - Q_0(x_k, \mu_1(x_k)) \leq -(1/\gamma)U(x_k, \mu_1(x_k)), \quad (6.3.28)$$

we can conclude that the control policy μ_1 is asymptotically stable for the system (6.3.1). Then, similar to (6.3.12)–(6.3.16), we can obtain that $Q^{\mu_1} = Q_{\infty}^{\mu_1} \leq Q_0$. Thus, the value function of μ_1 satisfies

$$V^{\mu_1}(x_k) = Q^{\mu_1}(x_k, \mu_1(x_k)) \leq Q_0(x_k, \mu_1(x_k)). \quad (6.3.29)$$

Therefore, we can conclude that $\mu_1(x_k)$ is an admissible control. The proof is complete.

According to Lemma 6.3.2, we can obtain an admissible control by using an initial positive-definite function Q_0 . Thus, considering Theorem 6.3.1, we obtain the following corollary.

Corollary 6.3.1 *Let Assumption 6.3.1 hold. Suppose that there is an initial positive-definite function Q_0 such that $\gamma Q_0 \geq Q_1^{\mu_1}$ for any x_k, u_k . Let $Q_j^{\mu_1}$ and μ_1 be updated by (6.3.10) and (6.3.11). Then the Q-function sequence $\{Q^{\mu_i}\}$ approaches Q^* according to the following inequalities*

$$Q^* \leq Q^{\mu_i} \leq \left[1 + \frac{\beta - 1}{(1 + \lambda^{-1})^i} \right] Q^*, \quad \forall i \geq 1. \quad (6.3.30)$$

From Theorem 6.3.1 and Corollary 6.3.1, we can see that as $i \rightarrow \infty$, Q^{μ_i} converges to Q^* under ideal conditions, i.e., the control policy and Q-function in each iteration can be obtained accurately. They also give a convergence rate of Q^{μ_i} with policy iteration. When the discount factor $\gamma = 1$, the discounted optimal control problem turns into an undiscounted optimal control problem, and Theorem 6.3.1 and Corollary 6.3.1 still hold.

However, in practice, considering that the iteration indices i and j cannot reach infinity as the algorithm must stop in finite steps, there exist convergence errors in

the iteration process. In addition, the control policy and Q-function in each iteration are obtained by approximation structures, so there exist approximate errors between approximate and accurate values. Hence, Theorem 6.3.1 and Corollary 6.3.1 may be invalid and the policy-iteration-based action-dependent ADP may even be divergent. To overcome this difficulty, in the following section we establish new error bound analysis results for Q-function considering the convergence and approximation errors.

6.3.3 Error Bound for Approximate Policy Iteration

For the approximate policy iteration, function approximation structures are used to approximate the Q-function and the control policy. The approximate expressions of μ_i and Q^{μ_i} are $\hat{\mu}_i$ and $\hat{Q}^{\hat{\mu}_i}$, respectively. We assume that there exist two finite positive constants $\underline{\delta} \leq 1$ and $\bar{\delta} \geq 1$ such that

$$\underline{\delta} Q^{\hat{\mu}_i} \leq \hat{Q}^{\hat{\mu}_i} \leq \bar{\delta} Q^{\hat{\mu}_i} \quad (6.3.31)$$

holds uniformly, for any $i \geq 1$, where $Q^{\hat{\mu}_i}$ is the exact Q-function associated with $\hat{\mu}_i$. $\underline{\delta}$ and $\bar{\delta}$ imply the convergence error in j -iteration and the approximation error of $Q^{\hat{\mu}_i}$ in policy evaluation phase. When $\underline{\delta} = \bar{\delta} = 1$, both errors are zero. Considering Lemma 6.3.1, we obtain

$$\hat{Q}^{\hat{\mu}_i} \leq \bar{\delta} Q^{\hat{\mu}_i} \leq \bar{\delta} \hat{Q}_1^{\hat{\mu}_i}, \quad (6.3.32)$$

where

$$\hat{Q}_1^{\hat{\mu}_i}(x_k, u_k) = U(x_k, u_k) + \gamma \hat{Q}^{\hat{\mu}_{i-1}}(x_{k+1}, \hat{\mu}_i(x_{k+1})).$$

Similarly, we assume that there exist two finite positive constants $\underline{\zeta} \leq 1$ and $\bar{\zeta} \geq 1$ such that

$$\underline{\zeta} Q_1^{\hat{\mu}_i} \leq \hat{Q}_1^{\hat{\mu}_i} \leq \bar{\zeta} Q_1^{\hat{\mu}_i} \quad (6.3.33)$$

holds uniformly, $\forall i \geq 1$, where

$$\hat{Q}_1^{\hat{\mu}_i}(x_k, u_k) = U(x_k, u_k) + \gamma \hat{Q}^{\hat{\mu}_{i-1}}(x_{k+1}, \hat{\mu}_i(x_{k+1})).$$

$\underline{\zeta}$ and $\bar{\zeta}$ imply the approximation errors of $\hat{\mu}_i$ in the policy improvement phase. If the iterative control policy can be obtained accurately, then $\underline{\zeta} = \bar{\zeta} = 1$. Considering (6.3.32) and (6.3.33), we can get

$$\hat{Q}^{\hat{\mu}_i} \leq \bar{\zeta} \bar{\delta} Q_1^{\hat{\mu}_i}. \quad (6.3.34)$$

On the other hand, considering (6.3.31) and (6.3.33), we have

$$\hat{Q}^{\hat{\mu}_i} \geq \underline{\delta} Q^{\hat{\mu}_i} \geq \underline{\delta} Q^*. \quad (6.3.35)$$

Therefore, the approximation errors in the Q-function and control policy update step can be expressed by

$$\underline{\eta} Q^* \leq \hat{Q}^{\hat{\mu}_i} \leq \bar{\eta} Q_1^{\hat{\mu}_i}, \quad (6.3.36)$$

where $\underline{\eta} = \underline{\delta}$ and $\bar{\eta} = \bar{\zeta} \bar{\delta}$. We establish the error bounds for approximate policy iteration by the following theorem.

Theorem 6.3.2 *Let Assumption 6.3.1 hold. Suppose that $Q^* \leq Q^{\mu_0} \leq \beta Q^*$, $1 \leq \beta \leq \infty$. μ_0 is an admissible control policy and Q^{μ_0} is the Q-function of μ_0 . The approximate Q-function $\hat{Q}^{\hat{\mu}_i}$ satisfies the iterative error condition (6.3.36). Then, the Q-function sequence $\{\hat{Q}^{\hat{\mu}_i}\}$ approaches Q^* according to the following inequalities*

$$\underline{\eta} Q^* \leq \hat{Q}^{\hat{\mu}_{i+1}} \leq \bar{\eta} \left[1 + \sum_{j=1}^i \frac{\lambda^j \bar{\eta}^{j-1} (\bar{\eta} - 1)}{(\lambda + 1)^j} + \frac{\lambda^i \bar{\eta}^i (\beta - 1)}{(\lambda + 1)^i} \right] Q^*, \quad \forall i \geq 0. \quad (6.3.37)$$

Moreover, the approximate value function sequence $\{\hat{Q}^{\hat{\mu}_i}\}$ converges to a finite neighborhood of Q^* uniformly on Ω as $i \rightarrow \infty$, i.e.,

$$\underline{\eta} Q^* \leq \lim_{i \rightarrow \infty} \hat{Q}^{\hat{\mu}_i} \leq \frac{\bar{\eta}}{1 + \lambda - \lambda \bar{\eta}} Q^*, \quad (6.3.38)$$

under the condition $\bar{\eta} \leq \frac{1}{\lambda} + 1$.

Proof First, the left-hand side of (6.3.37) holds clearly according to (6.3.36). Next, we prove the right-hand side of (6.3.37) for $i \geq 1$. Considering (6.3.36), we obtain

$$\begin{aligned} \hat{Q}^{\hat{\mu}_1}(x_k, u_k) &\leq \bar{\eta} Q_1^{\hat{\mu}_1}(x_k, u_k) \\ &\leq \bar{\eta} Q_0^{\hat{\mu}_1}(x_k, u_k) \\ &= \bar{\eta} Q^{\mu_0}(x_k, u_k) \\ &\leq \bar{\eta} \beta Q^*(x_k, u_k), \end{aligned} \quad (6.3.39)$$

which means (6.3.37) is true for $i = 0$. Then, considering (6.3.36) and Assumption 6.3.1, we can derive

$$\begin{aligned} \hat{Q}^{\hat{\mu}_2}(x_k, u_k) &\leq \bar{\eta} Q_1^{\hat{\mu}_2}(x_k, u_k) \\ &= \bar{\eta}[U(x_k, u_k) + \gamma \hat{Q}^{\hat{\mu}_1}(x_{k+1}, \hat{\mu}_2(x_{k+1}))] \\ &= \bar{\eta}[U(x_k, u_k) + \gamma \min_{u_{k+1}} Q^{\hat{\mu}_1}(x_{k+1}, u_{k+1})] \end{aligned}$$

$$\begin{aligned}
&\leq \bar{\eta}[U(x_k, u_k) + \gamma \bar{\eta} \beta \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1})] \\
&\quad + \bar{\eta} \frac{\bar{\eta} \beta - 1}{\lambda + 1} [\lambda U(x_k, u_k) - \gamma \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1})] \\
&= \bar{\eta} \left(1 + \lambda \frac{\bar{\eta} \beta - 1}{\lambda + 1}\right) U(x_k, u_k) \\
&\quad + \bar{\eta} \left(\bar{\eta} \beta - \frac{\bar{\eta} \beta - 1}{\lambda + 1}\right) \gamma \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \\
&= \bar{\eta} \left(1 + \lambda \frac{\bar{\eta} \beta - 1}{\lambda + 1}\right) Q^*(x_k, u_k) \\
&= \bar{\eta} \left(1 + \frac{\lambda(\bar{\eta} - 1)}{\lambda + 1} + \frac{\lambda \bar{\eta}(\beta - 1)}{\lambda + 1}\right) Q^*(x_k, u_k) \tag{6.3.40}
\end{aligned}$$

Hence, the upper bound of (6.3.37) holds for $i = 1$. Suppose that the upper bound of (6.3.37) holds for $\hat{Q}^{\hat{\mu}_i}$ ($i \geq 1$). Then, for $\hat{Q}^{\hat{\mu}_{i+1}}$, we derive

$$\begin{aligned}
\hat{Q}^{\hat{\mu}_{i+1}}(x_k, u_k) &\leq \bar{\eta} \left[U(x_k, u_k) + \gamma \min_{u_{k+1}} \hat{Q}^{\hat{\mu}_i}(x_{k+1}, u_{k+1}) \right] \\
&\leq \bar{\eta} \left[U(x_k, u_k) + \gamma \bar{\eta} \rho \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \right] \\
&\leq \bar{\eta} \left[U(x_k, u_k) + \gamma \bar{\eta} \rho \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \right] \\
&\quad + \bar{\eta} \Delta \left[\lambda U(x_k, u_k) - \gamma \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \right] \\
&= \bar{\eta} (1 + \Delta \lambda) \left[U(x_k, u_k) + \gamma \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \right] \\
&= \bar{\eta} (1 + \Delta \lambda) Q^*(x_k, u_k), \tag{6.3.41}
\end{aligned}$$

where

$$\rho = 1 + \sum_{j=1}^{i-1} \frac{\lambda^j \bar{\eta}^{j-1} (\bar{\eta} - 1)}{(\lambda + 1)^j} + \frac{\lambda^{i-1} \bar{\eta}^{i-1} (\beta - 1)}{(\lambda + 1)^{i-1}}$$

(for $\hat{Q}^{\hat{\mu}_i}$ from (6.3.37)), Δ satisfies $\Delta \geq 0$ and $1 + \Delta \lambda = \bar{\eta} \rho - \Delta$. Hence, we can calculate

$$\begin{aligned}
\Delta &= \frac{\bar{\eta} \rho - 1}{1 + \lambda} \\
&= \frac{\bar{\eta} - 1}{1 + \lambda} + \sum_{j=1}^{i-1} \frac{\lambda^j \bar{\eta}^j (\bar{\eta} - 1)}{(\lambda + 1)^{j+1}} + \frac{\lambda^{i-1} \bar{\eta}^i (\beta - 1)}{(\lambda + 1)^i} \\
&= \sum_{j=1}^i \frac{\lambda^{j-1} \bar{\eta}^{j-1} (\bar{\eta} - 1)}{(\lambda + 1)^j} + \frac{\lambda^{i-1} \bar{\eta}^i (\beta - 1)}{(\lambda + 1)^i}. \tag{6.3.42}
\end{aligned}$$

Substituting (6.3.42) into (6.3.41), we obtain

$$\hat{Q}^{\hat{\mu}_{i+1}} \leq \bar{\eta} \left[1 + \sum_{j=1}^i \frac{\lambda^j \bar{\eta}^{j-1} (\bar{\eta} - 1)}{(\lambda + 1)^j} + \frac{\lambda^i \bar{\eta}^i (\beta - 1)}{(\lambda + 1)^i} \right] Q^*. \quad (6.3.43)$$

Thus, the upper bound of $\hat{Q}^{\hat{\mu}_i}$ holds for $i + 1$. According to the mathematical induction, the right-hand side of (6.3.37) holds.

The conclusion in (6.3.38) can be proved following the same steps as in the proof of Theorem 6.2.2. The proof of the theorem is complete.

Remark 6.3.2 We can find that the upper bound is a monotonically increasing function of $\bar{\eta}$. The condition $\bar{\eta} \leq 1/\lambda + 1$ ensures that the upper bound in (6.3.38) is finite and positive. A larger λ will lead to a slower convergence rate and a larger error bound. Besides, a larger λ also requires more accurate iteration to converge. When $\underline{\eta} = \bar{\eta} = 1$, the approximate Q-function sequence $\hat{Q}^{\hat{\mu}_i}$ converges to Q^* uniformly on Ω as $i \rightarrow \infty$.

For the undiscounted optimal control problem, the discount factor $\gamma = 1$, and the Q-function is redefined as

$$Q^\mu(x, u) = U(x, u) + Q^\mu(x^+, \mu(x^+)), \quad (6.3.44)$$

and the optimal Q-function satisfies

$$Q^*(x, u) = U(x, u) + \min_{u^+} Q^*(x^+, u^+). \quad (6.3.45)$$

From Theorems 6.3.1 and 6.3.2, when $\gamma = 1$, all the deductions still hold. So we have the following corollary.

Corollary 6.3.2 *For the undiscounted optimal control problem with Assumption 6.3.1 and the admissible control policy μ_0 satisfying $Q^* \leq Q^{\mu_0} \leq \beta Q^*$, $1 \leq \beta \leq \infty$, if the approximate Q-function $\hat{Q}^{\hat{\mu}_i}$ satisfies the iterative error condition (6.3.36), the approximate Q-function sequence $\{\hat{Q}^{\hat{\mu}_i}\}$ converges to a finite neighborhood of Q^* uniformly on Ω as $i \rightarrow \infty$, i.e.,*

$$\underline{\eta} Q^* \leq \lim_{i \rightarrow \infty} \hat{Q}^{\hat{\mu}_i} \leq \frac{\bar{\eta}}{1 + \lambda - \bar{\eta}\lambda} Q^*, \quad (6.3.46)$$

under the condition $\bar{\eta} \leq 1/\lambda + 1$.

6.3.4 Neural Network Implementation

In the previous section, the approximate Q-function with policy iteration is proven to converge to a finite neighborhood of the optimal one. Hence, it is feasible to

approximate the Q-function and the control policy using neural networks. We present the detailed implementation of the present algorithm using neural networks in this section.

The structural diagram of the action-dependent iterative ADP algorithm in this section is shown in Fig. 6.7. The outputs of critic network and the action network are the approximations of the Q-function and the control policy, respectively.

The approximate Q-function $\hat{Q}_j^{\hat{u}_i}(x_k, u_k)$ is expressed by

$$\hat{Q}_j^{\hat{u}_i}(x_k, u_k) = W_{c(ij)}^T \zeta(V_{c(ij)}^T [x_k^T, u_k^T]^T), \quad (6.3.47)$$

where $\zeta(\cdot)$ is the activation function, which is selected as $\tanh(\cdot)$. The target function of the critic neural network training is given by

$$\hat{Q}_{i,j+1}^*(x_k, u_k) = U(x_k, u_k) + \hat{Q}_j^{\hat{u}_i}(x_{k+1}, \hat{\mu}_i(x_{k+1})), \quad (6.3.48)$$

where $x_{k+1} = F(x_k, \hat{\mu}_i(x_k))$. Then, the error function for the critic network training is defined by

$$e_{c(i,j+1)}(x_k) = \hat{Q}_{i,j+1}^*(x_k, u_k) - \hat{Q}_{j+1}^{\hat{u}_i}(x_k, u_k), \quad (6.3.49)$$

and the performance function to be minimized is defined by

$$E_{c(i,j+1)} = \frac{1}{2} e_{c(i,j+1)}^2. \quad (6.3.50)$$

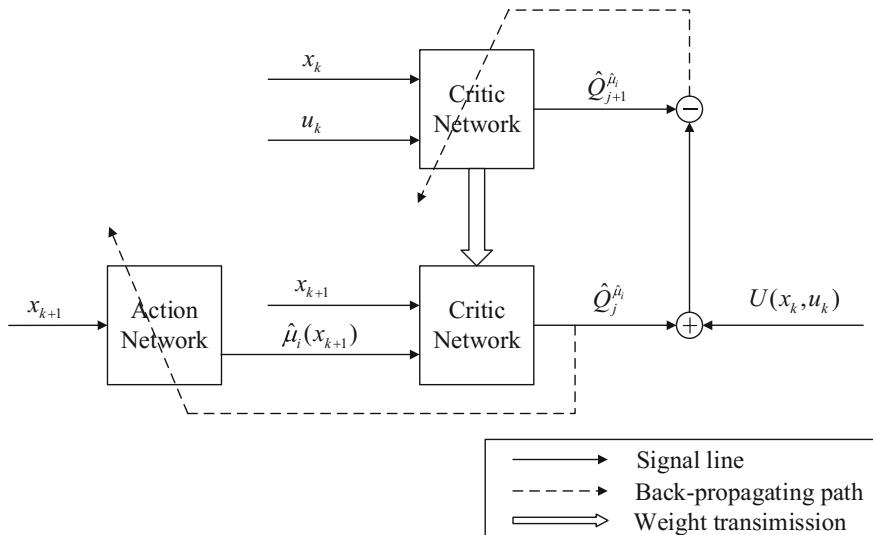


Fig. 6.7 Structural diagram of action-dependent iterative ADP

The approximate control policy $\hat{\mu}_i$ is expressed by the action network

$$\hat{\mu}_i(x_k) = W_{a(i)}^T \zeta(V_{a(i)}^T x_k). \quad (6.3.51)$$

The target function of the action network training is given by

$$\hat{\mu}_{i+1}^*(x_k) = \arg \min_{\mu} \hat{Q}^{\hat{\mu}_i}(x_k, u). \quad (6.3.52)$$

Then, the error function for training the action network is defined by

$$e_{a(i+1)}(x_k) = \hat{\mu}_{i+1}^*(x_k) - \hat{\mu}_{i+1}(x_k). \quad (6.3.53)$$

The performance function of the action network to be minimized is defined by

$$E_{a(i+1)} = \frac{1}{2} e_{a(i+1)}^T e_{a(i+1)}. \quad (6.3.54)$$

We use the gradient descent method to update the weights of critic and action networks on a training data set. The detailed process of the approximate policy iteration is given in Algorithm 6.3.1.

Algorithm 6.3.1 Approximate policy iteration for action-dependent ADP

1: Initialization:

 Initialize critic and action networks with random weights;

 Select an initial stabilizing control policy μ_0 ;

 Set the approximation errors of policy evaluation step and policy improvement step as θ and ξ , and maximum iteration numbers of policy evaluation step and policy improvement step as j_{\max} and i_{\max} .

2: Set $i = 0$.

3: For $j = 0, 1, \dots, j_{\max}$, update the Q-function $\hat{Q}_{j+1}^{\mu_i}(x_k)$ by minimizing (6.3.50) on the training set $\{x_k\}$. When $j = j_{\max}$ or the convergence conditions are met, set $\hat{Q}^{\mu_i}(x_k) = \hat{Q}_{j+1}^{\mu_i}(x_k)$ and go to Step 4.

4: Update the control policy $\hat{\mu}_{i+1}(x_k)$ by minimizing (6.3.54) on the training set $\{x_k\}$.

5: Set $i \leftarrow i + 1$.

6: Repeat Steps 3–5 until the convergence conditions are met or $i = i_{\max}$.

7: Obtain the approximate optimal control policy $\hat{\mu}_i(x_k)$.

6.3.5 Simulation Study

In this section, we use a simulation example to demonstrate the effectiveness of the developed algorithm. Consider the mass-spring system whose dynamics is:

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} 0.05x_{2,k} \\ -0.0005x_{1,k} - 0.0335x_{1,k}^3 + x_{2,k} \end{bmatrix} + \begin{bmatrix} 0 \\ 0.05 \end{bmatrix} u_k. \quad (6.3.55)$$

Define the Q-function as

$$Q(x_0, u) = \sum_{k=0}^{\infty} \gamma^k (x_k^T Q x_k + u_k^T R u_k), \quad (6.3.56)$$

where

$$Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix},$$

$R = 0.1$ and $\gamma = 0.95$.

In the simulation, we choose two three-layer neural networks as the approximation structures of controller and Q-function. The structures of the critic and action neural networks are chosen as 3–8–1 and 2–8–1, respectively. The activation function is selected as $\tanh(\cdot)$. The convergence of neural network training, the initial weights of the activation functions are chosen randomly around zero. We set the initial weights of both the critic and action networks as random values with uniform distribution in $[-0.01, 0.01]$. The preset approximation errors are $\theta = \xi = 10^{-8}$, and the maximum iteration steps of policy evaluation and policy improvement are $j_{\max} = 40$ and

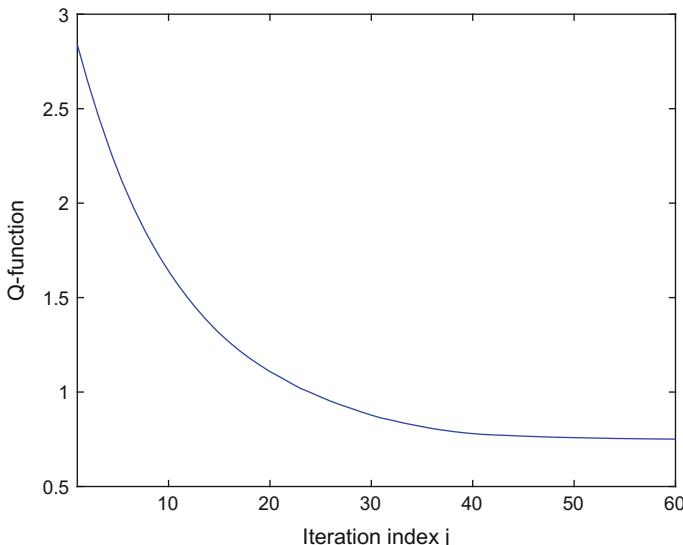


Fig. 6.8 The convergence of Q-function $\hat{Q}_j^{\hat{u}_i}$ on state $[0.5, -0.5]^T$ at $i = 1$

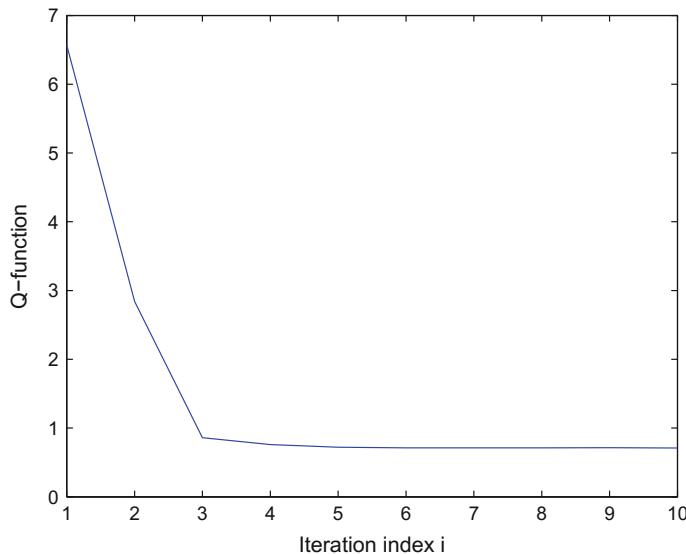


Fig. 6.9 The convergence of Q-function $\hat{Q}^{\hat{\mu}_i}$ on state $[0.5, -0.5]^T$

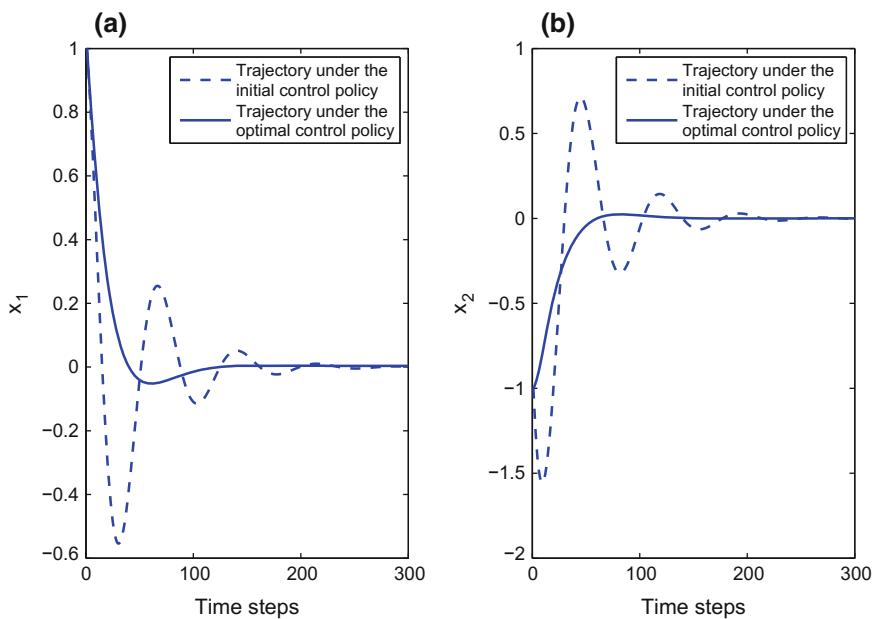


Fig. 6.10 The state trajectories starting from the state $[1, -1]^T$ **a** x_1 ; **b** x_2

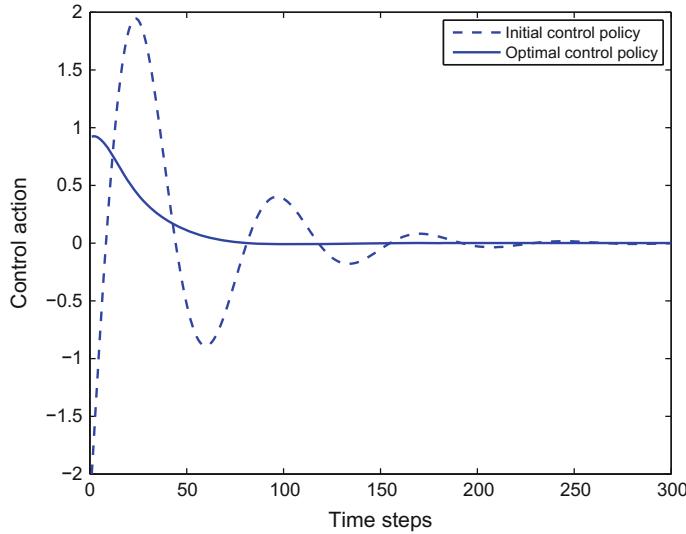


Fig. 6.11 The control signals corresponding to the states from $[1, -1]^\top$

$i_{\max} = 10$. The compact subset Ω of the state space is chosen as $0 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 1$. The training set $\{x_k\}$ contains 1000 samples choosing randomly from the compact set Ω . The initial admissible control policy is chosen as $\mu_0 = [-3, -1]x_k$.

We train the action and critic neural networks off-line with the Algorithm 6.3.1. Figure 6.8 illustrates the convergence process of the Q-function $\hat{Q}_j^{\hat{\mu}_i}$ with the iteration index j on state $x = [-0.5, 0.5]^\top$ at $i = 1$. Figure 6.9 shows the convergence curve of Q-function $\hat{Q}^{\hat{\mu}_i}$ on state $[0.5, 0.5]^\top$ with the iteration index i . Figure 6.10 shows the state trajectories from the initial state $[1, -1]^\top$ to the equilibrium under the initial control policy and the approximate optimal control policy obtained by our method, respectively. Figure 6.11 shows the action trajectories of the initial control policy and the approximate optimal control policy obtained by our method, respectively.

6.4 Conclusions

In this chapter, we established error bounds for approximate value iteration, approximate policy iteration, and approximate optimistic policy iteration. We considered approximation errors in both value function and control policy update equations. It was shown that the iterative approximate value function can converge to a finite neighborhood of the optimal value function under some mild conditions. The results provided theoretical guarantees for using neural network approximation for solving

undiscounted optimal control problems. We also developed the error bound analysis method of Q-function with approximate policy iteration for optimal control of unknown discounted discrete-time nonlinear systems.

References

1. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791
2. Almudevar A, Arruda EF (2012) Optimal approximation schedules for a class of iterative algorithms, with an application to multigrid value iteration. *IEEE Trans Autom Control* 57(12):3132–3146
3. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2008) Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans Syst Man Cybern-Part B: Cybern* 38(4):943–949
4. Bellman RE (1957) Dynamic programming. Princeton University Press, Princeton
5. Bertsekas DP (2012) Weighted sup-norm contractions in dynamic programming: a review and some new applications. Technical report LIDS-P-2884, MIT
6. Bertsekas DP (2013) Abstract dynamic programming. Athena Scientific, Belmont
7. Bertsekas DP (2016) Value and policy iterations in optimal control and adaptive dynamic programming. *IEEE Trans Neural Netw Learn Syst* (Online Available). doi:[10.1109/TNNLS.2015.2503980](https://doi.org/10.1109/TNNLS.2015.2503980)
8. Dierks T, Thumati BT, Jagannathan S (2009) Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence. *Neural Netw* 22(5):851–860
9. Grune L, Rantzer A (2008) On the infinite horizon performance of receding horizon controllers. *IEEE Trans Autom Control* 53(9):2100–2111
10. Heydari A, Balakrishnan SN (2013) Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics. *IEEE Trans Neural Netw Learn Syst* 24(1):145–157
11. Howard RA (1960) Dynamic programming and Markov processes. MIT Press, Cambridge
12. Leake RJ, Liu RW (1967) Construction of suboptimal control sequences. *SIAM J Control Optim* 5(1):54–63
13. Lewis FL, Syrmos VL (1995) Optimal control. Wiley, New York
14. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst Mag* 9(3):32–50
15. Li H, Liu D (2012) Optimal control for discrete-time affine non-linear systems using general value iteration. *IET Control Theory Appl* 6(18):2725–2736
16. Lincoln B, Rantzer A (2006) Relaxing dynamic programming. *IEEE Trans Autom Control* 51(8):1249–1260
17. Liu D, Wei Q (2013) Finite-approximation-error based optimal control approach for discrete-time nonlinear systems. *IEEE Trans Cybern* 43(2):779–789
18. Liu D, Wei Q (2014) Policy iterative adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Trans Neural Netw Learn Syst* 25(3):621–634
19. Liu D, Wang D, Zhao D, Wei Q, Jin N (2012) Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming. *IEEE Trans Autom Sci Eng* 9(3):628–634
20. Liu D, Wang D, Yang X (2013) An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs. *Inf Sci* 220:331–342
21. Liu D, Li H, Wang D (2015) Error bounds of adaptive dynamic programming algorithms for solving undiscounted optimal control problems. *IEEE Trans Neural Netw Learn Syst* 26(6):1323–1334

22. Puterman M, Shin M (1978) Modified policy iteration algorithms for discounted Markov decision problems. *Manag Sci* 24(11):1127–1137
23. Rantzer A (2006) Relaxed dynamic programming in switching systems. *IEE Proc-Control Theory Appl* 153(5):567–574
24. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
25. Tsitsiklis JN (2002) On the convergence of optimistic policy iteration. *J Mach Learn Res* 3:59–72
26. Vrabie D, Vamvoudakis KG, Lewis FL (2013) Optimal adaptive control and differential games by reinforcement learning principles. IET, London
27. Wang FY, Zhang H, Liu D (2009) Adaptive dynamic programming: an introduction. *IEEE Comput Intell Mag* 4(2):39–47
28. Wang FY, Jin N, Liu D, Wei Q (2011) Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ϵ -error bound. *IEEE Trans Neural Netw* 22(1):24–36
29. Wang D, Liu D, Wei Q, Zhao D, Jin N (2012) Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. *Automatica* 48(8):1825–1832
30. Wei Q, Wang FY, Liu D, Yang X (2014) Finite-approximation-error based discrete-time iterative adaptive dynamic programming. *IEEE Trans Cybern* 44(12):2820–2833
31. Werbos PJ (1992) Approximate dynamic programming for real-time control and neural modeling. In: White DA, Sofge DA (eds) *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches* (Chap. 13). Van Nostrand Reinhold, New York
32. Yan P, Wang D, Li H, Liu D (2016) Error bound analysis of Q-function for discounted optimal control problems with policy iteration. *IEEE Trans Syst Man Cybern Syst* (Online Available) doi: [10.1109/TSMC.2016.2563982](https://doi.org/10.1109/TSMC.2016.2563982)
33. Zhang H, Wei Q, Luo Y (2008) A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm. *IEEE Trans Syst Man Cybern-Part B: Cybern* 38(4):937–942
34. Zhang H, Luo Y, Liu D (2009) Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints. *IEEE Trans Neural Netw* 20(9):1490–1503
35. Zhang H, Song R, Wei Q, Zhang T (2011) Optimal tracking control for a class of nonlinear discrete-time systems with time delays based on heuristic dynamic programming. *IEEE Trans Neural Netw* 22(1):24–36

Part II

Continuous-Time Systems

Chapter 7

Online Optimal Control of Continuous-Time Affine Nonlinear Systems

7.1 Introduction

Optimal control problems for continuous-time nonlinear dynamical systems have been intensively studied during the past several decades [4, 14, 15, 18–20, 30, 31, 34, 39, 40]. A core challenge in deriving solutions of nonlinear optimal control problems is that it often fails to solve the Hamilton–Jacobi–Bellman (HJB) equations. It is well-known that the HJB equation is actually a nonlinear partial differential equation, which is difficult or impossible to solve by analytical methods. To cope with the problem, in this chapter, we develop approximate optimal control schemes for continuous-time affine nonlinear systems using adaptive dynamic programming (ADP) approach.

First, we consider an optimal control problem for a class of continuous-time partially unknown affine nonlinear systems. By employing an identifier–critic architecture, an online algorithm based on ADP approach is developed to derive the approximate optimal control [34]. Then, an ADP algorithm is presented to obtain the optimal control for continuous-time nonlinear systems with control constraints [31].

7.2 Online Optimal Control of Partially Unknown Affine Nonlinear Systems

Consider the continuous-time input-affine nonlinear systems described by

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t)), \quad x(0) = x_0, \quad (7.2.1)$$

where $x(t) \in \mathbb{R}^n$ is the state vector available for measurement, $u(t) \in \mathbb{R}^m$ is the control vector, $f(x) \in \mathbb{R}^n$ is an unknown nonlinear function with $f(0) = 0$, and $g(x) \in \mathbb{R}^{n \times m}$ is a matrix of known nonlinear functions. For the convenience of subsequent analysis, we provide two assumptions as follows.

Assumption 7.2.1 $f(x) + g(x)u$ is Lipschitz continuous on a compact set $\Omega \subset \mathbb{R}^n$ containing the origin, such that the solution $x(t)$ of system (7.2.1) is unique for all $x_0 \in \Omega$ and $u \in \mathbb{R}^m$. In addition, system (7.2.1) is controllable.

Assumption 7.2.2 The control matrix $g(x)$ is bounded over a compact set Ω ; that is, there exist constants g_m and g_M ($0 < g_m < g_M$) such that $g_m \leq \|g(x)\|_{\mathbb{F}} \leq g_M$ for all $x \in \Omega$.

Note that $g(\cdot) \in \mathbb{R}^{n \times m}$ is a matrix function. Our analysis results in this chapter (and Chap. 8) will use Frobenius matrix norm, which is defined as

$$\|A\|_{\mathbb{F}} = \sqrt{\sum_{i=1}^n \sum_{j=1}^m a_{ij}^2}$$

for matrix $A = (a_{ij}) \in \mathbb{R}^{n \times m}$. For the convenience of presentation, we will drop the subscript “ \mathbb{F} ” for Frobenius matrix norm in this chapter and the next. Accordingly, vector norm will also be chosen as the compatible one, i.e., the Euclidean norm for vectors which is defined as

$$\|x\| = \sqrt{\sum_{i=1}^n x_i^2}$$

for $x \in \mathbb{R}^n$. In other words, whenever we use norms in this chapter and Chap. 8, we mean Euclidean norm for vectors and Frobenius norm for matrices.

The value function for system (7.2.1) is defined by

$$V(x(t)) = \int_t^\infty r(x(s), u(s)) ds, \quad (7.2.2)$$

where

$$r(x, u) = x^T Q x + u^T R u,$$

and Q and R are constant symmetric positive-definite matrices with appropriate dimensions.

Definition 7.2.1 (cf. [11]) The solution $x(t)$ of system (7.2.1) is said to be uniformly ultimately bounded (UUB), if there exist positive constants b and c , independent of $t_0 \geq 0$, and for every $a \in (0, c)$, there is $T = T(a, b) > 0$, independent of t_0 , such that

$$\|x(t_0)\| \leq a \Rightarrow \|x(t)\| \leq b, \forall t \geq t_0 + T.$$

The control objective of this chapter is to obtain an online adaptive control not only stabilizes system (7.2.1) but also minimizes the value function (7.2.2), while ensuring that all signals in the closed-loop system are UUB.

7.2.1 Identifier–Critic Architecture for Solving HJB Equation

In control engineering, NNs are considered as powerful tools for approximating nonlinear functions due to their nonlinearity, adaptivity, self-learning, and fault tolerance. In this section, a single-hidden layer feedforward NN is applied to approximate $\mathcal{F}(x) \in \mathcal{C}^n(\mathcal{Q})$ ($\mathcal{F}(x)$ is a nonlinear function to be described later) as follows [10]:

$$\mathcal{F}(x) = W_m^T \sigma_m(Y_m^T x) + \varepsilon_m(x), \quad (7.2.3)$$

where $\sigma_m(\cdot) \in \mathbb{R}^{N_0}$ is the activation function, $\varepsilon_m(x) \in \mathbb{R}^n$ is the NN function reconstruction error, and $W_m \in \mathbb{R}^{N_0 \times n}$ and $Y_m \in \mathbb{R}^{n \times N_0}$ are the weight matrices of the hidden layer to the output layer and the input layer to the hidden layer, respectively. The number of the hidden layer nodes is denoted by N_0 . The activation function $\sigma_m(\cdot)$ is a componentwise bounded, measurable, and nondecreasing function from the real numbers onto $[-1, 1]$. Without loss of generality, in this chapter, we choose the hyperbolic tangent function as the activation function, i.e., $\sigma_m(\alpha) = \tanh(\alpha) = (e^\alpha - e^{-\alpha})/(e^\alpha + e^{-\alpha})$.

From system (7.2.1), we have

$$\dot{x}(t) = Ax + \mathcal{F}(x) + g(x)u, \quad (7.2.4)$$

where $\mathcal{F}(x) = f(x) - Ax$ and $A \in \mathbb{R}^{n \times n}$ is a known Hurwitz matrix. Using (7.2.3), (7.2.4) can be rewritten as

$$\dot{x}(t) = Ax + W_m^T \sigma_m(Y_m^T x) + g(x)u + \varepsilon_m(x). \quad (7.2.5)$$

The NN identifier approximates system (7.2.1) as

$$\dot{\hat{x}}(t) = A\hat{x} + \hat{W}_m^T \sigma_m(\hat{Y}_m^T \hat{x}) + g(\hat{x})u + C\tilde{x}(t), \quad (7.2.6)$$

where $\hat{x}(t) \in \mathbb{R}^n$ is the identifier NN state, $\hat{W}_m \in \mathbb{R}^{N_0 \times n}$ and $\hat{Y}_m \in \mathbb{R}^{n \times N_0}$ are estimated weights, $\tilde{x}(t)$ is the identification error $\tilde{x}(t) \triangleq x(t) - \hat{x}(t)$, and the design matrix $C \in \mathbb{R}^{n \times n}$ selected such that $A - C$ is a Hurwitz matrix.

Using (7.2.5) and (7.2.6), we obtain the identification error dynamics as

$$\dot{\tilde{x}}(t) = A_c \tilde{x}(t) + \tilde{W}_m^T \sigma_m(\hat{Y}_m^T \hat{x}) + \delta(x), \quad (7.2.7)$$

where $A_c = A - C$, $\tilde{W}_m = W_m - \hat{W}_m$, and $\delta(x)$ is given as

$$\delta(x) = W_m^T [\sigma_m(Y_m^T x) - \sigma_m(\hat{Y}_m^T \hat{x})] + [g(x) - g(\hat{x})]u + \varepsilon_m(x).$$

Before proceeding further, we provide some assumptions and facts, which have been used in the literature [16, 17, 23, 25, 31–33, 35, 38].

Assumption 7.2.3 The ideal identifier NN weights W_m and Y_m are bounded by known positive constants \bar{W}_M and \bar{Y}_M , respectively. That is,

$$\|W_m\| \leq \bar{W}_M, \quad \|Y_m\| \leq \bar{Y}_M.$$

Assumption 7.2.4 The NN function approximation error $\varepsilon_m(x)$ is bounded over Ω as $\|\varepsilon_m(x)\| \leq \varepsilon_M$ for every $x \in \Omega$, where $\varepsilon_M > 0$ is a known constant.

Fact 7.2.1 *The NN activation function is bounded over Ω , i.e., there exists a known constant $\sigma_M > 0$ such that $\|\sigma_m(x)\| \leq \sigma_M$ for every $x \in \Omega$.*

Fact 7.2.2 *Since A_c is a Hurwitz matrix, there exists a positive-definite symmetric matrix $P \in \mathbb{R}^{n \times n}$ satisfying the Lyapunov equation*

$$A_c^\top P + P A_c = -\theta I_n,$$

where $\theta > 0$ is a design parameter and I_n denotes the $n \times n$ identity matrix.

Theorem 7.2.1 *Let Assumptions 7.2.1–7.2.4 hold. If the identifier NN estimated weights \hat{W}_m and \hat{Y}_m are updated as*

$$\dot{\hat{W}}_m = -l_1 \sigma_m(\hat{Y}_m^\top \hat{x}) \hat{x}^\top A_c^{-1} - \tau_1 \|\hat{x}\| \hat{W}_m, \quad (7.2.8)$$

$$\dot{\hat{Y}}_m = -l_2 \text{sgn}(\hat{x}) \hat{x}^\top A_c^{-1} \hat{W}_m^\top [I_{N_0} - \Phi(\hat{Y}_m^\top \hat{x})] - \tau_2 \|\hat{x}\| \hat{Y}_m, \quad (7.2.9)$$

where $l_i > 0$ ($i = 1, 2$) are given constant parameters, τ_i ($i = 1, 2$) satisfy

$$\tau_1 > l_1 \|A_c^{-1}\|^2 / 4, \quad \tau_2 > l_2, \quad (7.2.10)$$

$$\Phi(\hat{Y}_m^\top \hat{x}) = \text{diag} \left\{ \sigma_{m1}^2 \left(\hat{Y}_{m1}^\top \hat{x} \right), \dots, \sigma_{mN_0}^2 \left(\hat{Y}_{mN_0}^\top \hat{x} \right) \right\},$$

I_{N_0} is the $N_0 \times N_0$ identity matrix, $\text{sgn}(\hat{x}) = (\text{sgn}(\hat{x}_1), \dots, \text{sgn}(\hat{x}_n))^\top$, and $\text{sgn}(\cdot)$ is the componentwise sign function. Then, the identifier NN given in (7.2.6) can ensure that the identification error $\tilde{x}(t)$ in (7.2.7) converges to a compact set

$$\mathcal{Q}_{\tilde{x}} = \left\{ \tilde{x} : \|\tilde{x}\| \leq 2\varsigma/\theta \right\}, \quad (7.2.11)$$

where $\varsigma > 0$ is a constant to be determined later (see (7.2.20) below). In addition, the NN weight estimation errors $\tilde{W}_m = W_m - \hat{W}_m$ and $\tilde{Y}_m = Y_m - \hat{Y}_m$ are all guaranteed to be UUB.

Proof Consider the Lyapunov function candidate

$$L_1(x) = L_{11}(x) + L_{12}(x), \quad (7.2.12)$$

where

$$L_{11}(x) = \frac{1}{2}\tilde{x}^T P \tilde{x} \text{ and } L_{12}(x) = \frac{1}{2}\text{tr}(\tilde{W}_m^T l_1^{-1} \tilde{W}_m) + \frac{1}{2}\text{tr}(\tilde{Y}_m^T l_2^{-1} \tilde{Y}_m).$$

Taking the time derivative of $L_{11}(x)$ along the solutions of (7.2.7) and using Facts 7.2.1 and 7.2.2, it follows

$$\begin{aligned} \dot{L}_{11}(x) &= \frac{1}{2}(\dot{\tilde{x}}^T P \tilde{x} + \tilde{x}^T P \dot{\tilde{x}}) \\ &= -\frac{\theta}{2}\tilde{x}^T \tilde{x} + \tilde{x}^T P [\tilde{W}_m^T \sigma_m(\hat{Y}_m^T \hat{x}) + \delta(x)] \\ &\leq -\frac{\theta}{2}\|\tilde{x}\|^2 + \|\tilde{x}\| \|P\| \left(\|\tilde{W}_m\| \|\sigma_M + \delta_M\| \right), \end{aligned} \quad (7.2.13)$$

where δ_M is the upper bound of $\delta(x)$, i.e., $\|\delta(x)\| \leq \delta_M$. Actually, noticing that $u(x)$ is a continuous function defined on Ω , there exists a constant $u_M > 0$ such that $\|u(x)\| \leq u_M$ for every $x \in \Omega$. Then, by Assumptions 7.2.2–7.2.4 and Fact 7.2.1, we can conclude that $\delta(x)$ given in (7.2.7) is an upper bounded function.

Taking the time derivative of $L_{12}(x)$ and using (7.2.8) and (7.2.9), we obtain

$$\begin{aligned} \dot{L}_{12}(x) &= \text{tr}\{\tilde{W}_m^T l_1^{-1} \dot{\tilde{W}}_m\} \\ &= \text{tr}\left\{\tilde{W}_m^T \sigma_m(\hat{Y}_m^T \hat{x}) \tilde{x}^T A_c^{-1} + \frac{\tau_1}{l_1} \|\tilde{x}\| \tilde{W}_m^T (W_m - \tilde{W}_m)\right\} \\ &\quad + \text{tr}\left\{\tilde{Y}_m^T \text{sgn}(\hat{x}) \tilde{x}^T A_c^{-1} (W_m - \tilde{W}_m)^T\right. \\ &\quad \left. \times [I_{N_0} - \Phi(\hat{Y}_m^T \hat{x})] + \frac{\tau_2}{l_2} \|\tilde{x}\| \tilde{Y}_m^T (Y_m - \tilde{Y}_m)\right\}. \end{aligned} \quad (7.2.14)$$

Note that

$$\text{tr}(X_1 X_2^T) = \text{tr}(X_2^T X_1) = X_2^T X_1, \quad \forall X_1, X_2 \in \mathbb{R}^{n \times 1},$$

and

$$\text{tr}[\tilde{Z}^T (Z - \tilde{Z})] \leq \|\tilde{Z}\|_F \|Z\|_F - \|\tilde{Z}\|_F^2, \quad \forall Z, \tilde{Z} \in \mathbb{R}^{m \times n}. \quad (7.2.15)$$

We emphasize that (7.2.15) is true for Frobenius matrix norm, but it is not true for other matrix norms in general. As we have declared earlier, Frobenius norm for matrices and Euclidean norm for vectors are used in this chapter. We do not use the subscript “F” for Frobenius matrix norm for the convenience of presentation in this chapter.

Then, from (7.2.14), it follows

$$\begin{aligned}
\dot{L}_{12}(x) &= \tilde{x}^\top A_c^{-1} \tilde{W}_m^\top \sigma_m (\hat{Y}_m^\top \hat{x}) + \frac{\tau_1}{l_1} \|\tilde{x}\| \text{tr} [\tilde{W}_m^\top (W_m - \tilde{W}_m)] \\
&\quad + \tilde{x}^\top A_c^{-1} (W_m - \tilde{W}_m)^\top [I_{N_0} - \Phi(\hat{Y}_m^\top \hat{x})] \tilde{Y}_m^\top \text{sgn}(\hat{x}) \\
&\quad + \frac{\tau_2}{l_2} \|\tilde{x}\| \text{tr} [\tilde{Y}_m^\top (Y_m - \tilde{Y}_m)] \\
&\leq \alpha \sigma_M \|\tilde{x}\| \|\tilde{W}_m\| + \frac{\tau_1}{l_1} \|\tilde{x}\| (\bar{W}_M \|\tilde{W}_m\| - \|\tilde{W}_m\|^2) \\
&\quad + \alpha \|I_{N_0} - \Phi(\hat{Y}_m^\top \hat{x})\| \|\tilde{x}\| (\bar{W}_M + \|\tilde{W}_m\|) \|\tilde{Y}_m\| \\
&\quad + \frac{\tau_2}{l_2} \|\tilde{x}\| (\bar{Y}_M \|\tilde{Y}_m\| - \|\tilde{Y}_m\|^2), \tag{7.2.16}
\end{aligned}$$

where $\alpha = \|A_c^{-1}\|$. Combining (7.2.13) and (7.2.16) and noticing $\|I_{N_0} - \Phi(\hat{Y}_m^\top \hat{x})\| \leq 1$, we obtain the time derivative of the Lyapunov function as

$$\begin{aligned}
\dot{L}_1(x) &\leq -\frac{\theta}{2} \|\tilde{x}\|^2 + \left\{ \delta_M \|P\| + \left[(\|P\| + \alpha) \sigma_M + \frac{\tau_1}{l_1} \bar{W}_M \right] \|\tilde{W}_m\| \right. \\
&\quad + \left(\alpha \bar{W}_M + \frac{\tau_2}{l_2} \bar{Y}_M \right) \|\tilde{Y}_m\| - \left(\frac{\tau_1}{l_1} - \frac{\alpha^2}{4} \right) \|\tilde{W}_m\|^2 \\
&\quad \left. - \left(\frac{\tau_2}{l_2} - 1 \right) \|\tilde{Y}_m\|^2 - \left(\frac{\alpha}{2} \|\tilde{W}_m\| - \|\tilde{Y}_m\| \right)^2 \right\} \|\tilde{x}\| \\
&= -\frac{\theta}{2} \|\tilde{x}\|^2 + \left\{ \delta_M \|P\| + \left(\frac{\tau_1}{l_1} - \frac{\alpha^2}{4} \right) \beta_1^2 + \left(\frac{\tau_2}{l_2} - 1 \right) \beta_2^2 \right. \\
&\quad - \left(\frac{\tau_1}{l_1} - \frac{\alpha^2}{4} \right) (\|\tilde{W}_m\| + \beta_1)^2 - \left(\frac{\tau_2}{l_2} - 1 \right) (\|\tilde{Y}_m\| + \beta_2)^2 \\
&\quad \left. - \left(\frac{\alpha}{2} \|\tilde{W}_m\| - \|\tilde{Y}_m\| \right)^2 \right\} \|\tilde{x}\|, \tag{7.2.17}
\end{aligned}$$

where

$$\beta_1 = \frac{2l_1(\alpha + \|P\|)\sigma_M + 2\tau_1 \bar{W}_M}{\alpha^2 l_1 - 4\tau_1}, \quad \beta_2 = \frac{\alpha l_2 \bar{W}_M + \tau_2 \bar{Y}_M}{2(l_2 - \tau_2)}.$$

In the above derivation, we have used the following identity,

$$ab = \frac{1}{2} \left\{ \left(\nu a + \frac{b}{\nu} \right)^2 - \left(\nu^2 a^2 + \frac{b^2}{\nu^2} \right) \right\} \tag{7.2.18}$$

for arbitrary $a, b \in \mathbb{R}$ and $\nu \neq 0$. Using (7.2.10), we derive (7.2.17) as

$$\begin{aligned}
\dot{L}_1(x) &\leq -\frac{\theta}{2} \|\tilde{x}\|^2 + \left\{ \delta_M \|P\| + \left(\frac{\tau_1}{l_1} - \frac{\alpha^2}{4} \right) \beta_1^2 + \left(\frac{\tau_2}{l_2} - 1 \right) \beta_2^2 \right\} \|\tilde{x}\| \\
&= -\left(\frac{\theta}{2} \|\tilde{x}\| - \varsigma \right) \|\tilde{x}\|, \tag{7.2.19}
\end{aligned}$$

where

$$\varsigma = \delta_M \|P\| + \left(\frac{\tau_1}{l_1} - \frac{\alpha^2}{4} \right) \beta_1^2 + \left(\frac{\tau_2}{l_2} - 1 \right) \beta_2^2. \quad (7.2.20)$$

Hence, $\dot{L}_1(x)$ is negative as long as $\|\tilde{x}\| > 2\varsigma/\theta$. That is, the identification error $\tilde{x}(t)$ converges to $\Omega_{\tilde{x}}$ defined as in (7.2.11). Meanwhile, according to the standard Lyapunov's extension theorem [12, 13] (or the Lagrange stability result [22]), this verifies the uniform ultimate boundedness of the NN weight estimation errors \tilde{W}_m and \tilde{Y}_m , which completes the proof of the theorem.

Remark 7.2.1 The first terms of (7.2.8) and (7.2.9) are derived from the standard backpropagation algorithm, and the last terms are employed to ensure the boundedness of parameter estimations. The size of $\Omega_{\tilde{x}}$ in (7.2.11) can be kept sufficiently small by properly choosing parameters, for example, θ , τ_i , and l_i ($i = 1, 2$), such that higher accuracy of identification is guaranteed. Although (7.2.8) and (7.2.9) share similar feature as in [16], a significant difference between [16] and the present work is that, in our case, we do not use Taylor series in the process of identification. Due to residual errors from the Taylor series expansion, our method is considered to be more accurate in estimating unknown system dynamics.

Since system (7.2.1) can be approximated well by (7.2.6) outside of the compact set $\Omega_{\tilde{x}}$, in what follows we replace system (7.2.1) with (7.2.6). Meanwhile, we replace the actual system state $x(t)$ with the estimated state $\hat{x}(t)$. In this circumstance, system (7.2.1) can be represented using (7.2.6) as

$$\dot{\hat{x}}(t) = h(\hat{x}) + g(\hat{x})u, \quad (7.2.21)$$

where

$$h(\hat{x}) = A\hat{x} + \hat{W}_m^T \sigma_m \left(\hat{Y}_m^T \hat{x} \right) + C\tilde{x}.$$

The value function (7.2.2) is rewritten as

$$V(\hat{x}(t)) = \int_t^\infty (Q(\hat{x}(s)) + u^T(s) R u(s)) ds, \quad (7.2.22)$$

where $Q(\hat{x}) = \hat{x}^T Q \hat{x}$.

Definition 7.2.2 (cf. [2]) A control $u(\hat{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is said to be admissible with respect to (7.2.22) on Ω , written as $u(\hat{x}) \in \mathcal{A}(\Omega)$, if $u(\hat{x})$ is continuous on Ω , $u(0) = 0$, $u(\hat{x})$ stabilizes system (7.2.21) on Ω and $V(\hat{x})$ is finite for every $\hat{x} \in \Omega$.

If the control $u(\hat{x}) \in \mathcal{A}(\Omega)$ and the value function $V(\hat{x}) \in \mathcal{C}^1(\Omega)$, then (7.2.21) and (7.2.22) are equivalent to

$$V_{\hat{x}}^T(h(\hat{x}) + g(\hat{x})u) + Q(\hat{x}) + u^T R u = 0,$$

where $V_{\hat{x}} \in \mathbb{R}^n$ represents the partial derivative of $V(\hat{x})$ with respect to \hat{x} , i.e., $V_{\hat{x}} = \frac{\partial V(\hat{x})}{\partial \hat{x}}$.

Now, define the Hamiltonian for the control $u(\hat{x})$ and the value function $V(\hat{x})$ as

$$H(\hat{x}, V_{\hat{x}}, u) = V_{\hat{x}}^T (h(\hat{x}) + g(\hat{x})u) + Q(\hat{x}) + u^T R u.$$

Then, the optimal value function $V^*(\hat{x})$ is obtained by solving the Hamilton–Jacobi–Bellman (HJB) equation

$$\min_{u(\hat{x}) \in \mathcal{A}(\Omega)} H(\hat{x}, V_{\hat{x}}^*, u) = 0. \quad (7.2.23)$$

Therefore, the closed-form expression for the optimal control can be derived as

$$u^*(\hat{x}) = -\frac{1}{2} R^{-1} g^T(\hat{x}) V_{\hat{x}}^*. \quad (7.2.24)$$

Substituting (7.2.24) into (7.2.23), we get the HJB equation as

$$(V_{\hat{x}}^*)^T h(\hat{x}) + Q(\hat{x}) - \frac{1}{4} (V_x^*)^T g(\hat{x}) R^{-1} g^T(\hat{x}) V_{\hat{x}}^* = 0. \quad (7.2.25)$$

In this sense, one shall find that (7.2.25) is actually a nonlinear partial differential equation with respect to $V^*(\hat{x})$, which is difficult to solve by analytical methods. To confront the challenge, an online NN-based optimal control scheme will be developed. Prior to proceeding further, we provide the following required assumption.

Assumption 7.2.5 $L_2(\hat{x})$ is a continuously differentiable Lyapunov function candidate for system (7.2.21) and satisfies

$$\dot{L}_2(\hat{x}) = L_{2\hat{x}}^T (h(\hat{x}) + g(\hat{x})u^*) < 0$$

with $L_{2\hat{x}}$ the partial derivative of $L_2(\hat{x})$ with respect to \hat{x} . Meanwhile, there exists a symmetric positive definite matrix $\Lambda_2(\hat{x}) \in \mathbb{R}^{n \times n}$ defined on Ω such that

$$L_{2\hat{x}}^T (h(\hat{x}) + g(\hat{x})u^*) = -L_{2\hat{x}}^T \Lambda_2(\hat{x}) L_{2\hat{x}}. \quad (7.2.26)$$

Remark 7.2.2 $h(\hat{x}) + g(\hat{x})u^*$ is often assumed to be bounded by a positive constant [3, 23, 25, 29], i.e., there exists a constant $\rho > 0$ such that $\|h(\hat{x}) + g(\hat{x})u^*\| \leq \rho$. To relax the condition, in this section, $h(\hat{x}) + g(\hat{x})u^*$ is assumed to be bounded by a function with respect to \hat{x} . Since $L_{2\hat{x}}$ is the function with respect to \hat{x} , without loss of generality, we assume that

$$\|h(\hat{x}) + g(\hat{x})u^*\| \leq \rho \|L_{2\hat{x}}\|,$$

where $\rho > 0$. Noticing $L_{2\hat{x}}^T(h(\hat{x}) + g(\hat{x})u^*) < 0$, one shall find that (7.2.26) defined as in Assumption 7.2.5 is reasonable. In addition, $L_2(\hat{x})$ can be derived through properly selecting functions, such as polynomials.

In what follows, an online optimal control scheme is constructed using a single critic NN. Due to the universal approximation property of NNs [8, 10], the optimal value function $V^*(\hat{x})$ can be represented by a single-layer NN on a compact set Ω as [1, 29]

$$V^*(\hat{x}) = W_c^T \sigma(\hat{x}) + \varepsilon_{N_1}(\hat{x}),$$

where $W_c \in \mathbb{R}^{N_1}$ is the ideal NN weight vector,

$$\sigma(\hat{x}) = [\sigma_1(\hat{x}), \dots, \sigma_{N_1}(\hat{x})]^T \in \mathbb{R}^{N_1}$$

is the vector of activation function with $\sigma_j(\hat{x}) \in \mathcal{C}^1(\Omega)$ and $\sigma_j(0) = 0$, the set $\{\sigma_j(\hat{x})\}_{j=1}^{N_1}$ is often selected to be linearly independent, N_1 is the number of the neurons, and $\varepsilon_{N_1}(\hat{x})$ is the NN function reconstruction error. The derivative of $V^*(\hat{x})$ with respect to \hat{x} is given as

$$V_{\hat{x}}^* = \nabla \sigma^T(\hat{x}) W_c + \nabla \varepsilon_{N_1}(\hat{x}), \quad (7.2.27)$$

where $\nabla \sigma(\hat{x}) = \partial \sigma(\hat{x}) / \partial \hat{x}$, $\nabla \varepsilon_{N_1} = \partial \varepsilon_{N_1}(\hat{x}) / \partial \hat{x}$, and $\nabla \sigma(0) = 0$.

Using (7.2.27), (7.2.24) can be represented as

$$u^*(\hat{x}) = -\frac{1}{2} R^{-1} g^T(\hat{x}) \nabla \sigma^T W_c - \frac{1}{2} R^{-1} g^T(\hat{x}) \nabla \varepsilon_{N_1}. \quad (7.2.28)$$

By the same token, (7.2.25) can be rewritten as

$$W_c^T \nabla \sigma h(\hat{x}) + Q(\hat{x}) - \frac{1}{4} W_c^T \nabla \sigma g(\hat{x}) R^{-1} g^T(\hat{x}) \nabla \sigma^T W_c + \varepsilon_{\text{HJB}} = 0, \quad (7.2.29)$$

where ε_{HJB} is the residual error converging to zero when N_1 is large enough [1]; that is, there exists a small constant $\varepsilon_{a_1} > 0$ such that $\|\varepsilon_{\text{HJB}}\| \leq \varepsilon_{a_1}$.

Since the ideal NN weight vector W_c is often unavailable, (7.2.28) cannot be implemented in real control process. Hence, we employ a critic NN to approximate the value function given in (7.2.22) as

$$\hat{V}(\hat{x}) = \hat{W}_c^T \sigma(\hat{x}), \quad (7.2.30)$$

where \hat{W}_c is the estimated weight of W_c . The weight estimation error for the critic NN is defined as

$$\tilde{W}_c = W_c - \hat{W}_c. \quad (7.2.31)$$

Using (7.2.30), the estimate of (7.2.24) is given by

$$\hat{u}(\hat{x}) = -\frac{1}{2}R^{-1}g^T(\hat{x})\nabla\sigma^T\hat{W}_c. \quad (7.2.32)$$

The approximate Hamiltonian is derived as

$$\begin{aligned} H(\hat{x}, \hat{W}_c) &= \hat{W}_c^T \nabla\sigma h(\hat{x}) + Q(\hat{x}) - \frac{1}{4} \hat{W}_c^T \nabla\sigma G(\hat{x}) \nabla\sigma^T \hat{W}_c \\ &\triangleq e_1, \end{aligned} \quad (7.2.33)$$

where

$$G(\hat{x}) = g(\hat{x})R^{-1}g^T(\hat{x}).$$

Combining (7.2.28), (7.2.29), and (7.2.33), we have

$$\begin{aligned} e_1 &= (W_c - \tilde{W}_c)^T \nabla\sigma h(\hat{x}) + Q(\hat{x}) - \frac{1}{4}(W_c - \tilde{W}_c)^T \nabla\sigma G(\hat{x}) \nabla\sigma^T (W_c - \tilde{W}_c) \\ &= \underbrace{W_c^T \nabla\sigma h(\hat{x}) + Q(\hat{x}) - \frac{1}{4}W_c^T \nabla\sigma G(\hat{x}) \nabla\sigma^T W_c}_{-\varepsilon_{\text{HJB}}} \\ &\quad - \tilde{W}_c^T \nabla\sigma h(\hat{x}) + \frac{1}{2}\tilde{W}_c^T \nabla\sigma G(\hat{x}) \nabla\sigma^T W_c - \frac{1}{4}\tilde{W}_c^T \nabla\sigma G(\hat{x}) \nabla\sigma^T \tilde{W}_c \\ &= -\tilde{W}_c^T \nabla\sigma h(\hat{x}) + \tilde{W}_c^T \nabla\sigma g(\hat{x}) \left(-u^*(\hat{x}) - \frac{1}{2}R^{-1}g^T(\hat{x})\nabla\varepsilon_{N_1} \right) \\ &\quad - \frac{1}{4}\tilde{W}_c^T \nabla\sigma G(\hat{x}) \nabla\sigma^T \tilde{W}_c - \varepsilon_{\text{HJB}} \\ &= -\tilde{W}_c^T \nabla\sigma \left(\xi(\hat{x}) + \frac{1}{2}G(\hat{x})\nabla\varepsilon_{N_1} \right) - \frac{1}{4}\tilde{W}_c^T \nabla\sigma G(\hat{x}) \nabla\sigma^T \tilde{W}_c - \varepsilon_{\text{HJB}}, \end{aligned} \quad (7.2.34)$$

where $\xi(\hat{x}) = h(\hat{x}) + g(\hat{x})u^*(\hat{x})$.

To obtain the minimum value of e_1 , one often chooses \hat{W}_c to minimize the squared residual error $E = \frac{1}{2}e_1^2$. Using the gradient descent algorithm and (7.2.33), the weight tuning law for the critic NN is generally given as [3, 29, 37]

$$\begin{aligned} \dot{\hat{W}}_c &= -\frac{\eta_1}{(1 + \phi^T\phi)^2} \frac{\partial E}{\partial \hat{W}_c} \\ &= -\frac{\eta_1}{(1 + \phi^T\phi)^2} e_1 \frac{\partial e_1}{\partial \hat{W}_c} \\ &= -\eta_1 \frac{\phi}{(1 + \phi^T\phi)^2} e_1, \end{aligned} \quad (7.2.35)$$

where $\phi = \nabla\sigma(h(\hat{x}) + g(\hat{x})\hat{u}(\hat{x}))$, $\eta_1 > 0$ is a design constant, and the term $(1 + \phi^T\phi)^2$ is employed for normalization.

However, there exist two issues about the tuning rule (7.2.35):

- (i) If the initial control is not admissible, then tuning the critic NN weights based on (7.2.35) to minimize $E = (1/2)e_1^\top e_1$ may not guarantee the stability of system (7.2.21) during the critic NN learning process.
- (ii) The persistence of excitation (PE) of $\phi/(1 + \phi^\top \phi)$ is required to guarantee the weights of the critic NN to converge to actual optimal values [3, 7, 26, 29, 36, 37]. Nevertheless, the PE condition is often intractable to verify. In addition, a small exploratory signal is often added to the control input in order to satisfy the PE condition, which might cause instability of the closed-loop system during the implementation of the algorithm.

To address the above two issues, a novel weight update law for the critic NN is developed as follows:

$$\begin{aligned}\dot{\hat{W}}_c = & -\eta_1 \bar{\phi} \left(\mathcal{E}(\hat{x}) - \frac{1}{4} \hat{W}_c^\top \nabla \sigma G(\hat{x}) \nabla \sigma^\top \hat{W}_c \right) \\ & - \eta_1 \sum_{j=1}^{N_1} \bar{\phi}_{(j)} \left(\mathcal{E}(\hat{x}_{t_j}) - \frac{1}{4} \hat{W}_c^\top \nabla \sigma_{(j)} G(\hat{x}_{t_j}) \nabla \sigma_{(j)}^\top \hat{W}_c \right) \\ & + \frac{\eta_1}{2} \Pi(\hat{x}, \hat{u}) \nabla \sigma G(\hat{x}) L_{2\hat{x}},\end{aligned}\quad (7.2.36)$$

where $\mathcal{E}(\hat{x}) = \hat{W}_c^\top \nabla \sigma h(\hat{x}) + Q(\hat{x})$, $G(\hat{x})$ is given in (7.2.33), $\bar{\phi} = \phi/m_s^2$, $m_s = 1 + \phi^\top \phi$, $j \in \{1, \dots, N_1\}$ denotes the index of a stored/recoded data point $\hat{x}(t_j)$ (written as \hat{x}_{t_j}), $\bar{\phi}_{(j)} = \bar{\phi}(\hat{x}_{t_j})$, $m_{s_j} = 1 + \phi^\top(\hat{x}_{t_j})\phi(\hat{x}_{t_j})$, $\nabla \sigma_{(j)} = \nabla \sigma(\hat{x}_{t_j})$, $L_{2\hat{x}}$ is defined as in Assumption 7.2.5, and $\Pi(\hat{x}, \hat{u})$ is given as

$$\Pi(\hat{x}, \hat{u}) = \begin{cases} 0, & \text{if } L_{2\hat{x}}^\top \left(h(\hat{x}) - \frac{1}{2} G(\hat{x}) \nabla \sigma^\top \hat{W}_c \right) < 0, \\ 1, & \text{otherwise.} \end{cases}\quad (7.2.37)$$

Remark 7.2.3 Several notes about the weight tuning rule (7.2.36) are listed as follows:

- (a) The first term in (7.2.36) shares the same feature as (7.2.35), which aims to minimize the objective function $E = (1/2)e_1^\top e_1$.
- (b) The second term in (7.2.36) is utilized to relax the PE condition. If there is no second term in (7.2.36), one shall find $\dot{\hat{W}}_c = 0$ when $\hat{x} = 0$. That is, the weight of the critic NN will not be updated. In this circumstance, the critic NN might not converge to the optimal weights. Therefore, the PE condition is often employed [3, 7, 26, 29, 36, 37]. Interestingly, the second term given in (7.2.36) can also avoid this issue as long as the set $\{\bar{\phi}_{(j)}\}_{j=1}^{N_1}$ is selected to be linearly independent.

Now, we show this fact by contradiction. Suppose that $\dot{\hat{W}}_c = 0$ when $\hat{x} = 0$. From (7.2.36), we obtain

$$\sum_{j=1}^{N_1} \bar{\phi}_{(j)} e_{1j} = 0,$$

where

$$e_{1j} = \mathcal{E}(\hat{x}_{t_j}) - \frac{1}{4} \hat{W}_c^\top \nabla \sigma_{(j)} G(\hat{x}_{t_j}) \nabla \sigma_{(j)}^\top \hat{W}_c.$$

Since $\{\bar{\phi}_{(j)}\}_{j=1}^{N_1}$ is linearly independent, we can obtain $e_{1j} = 0$ ($j = 1, \dots, N_1$). However, this case will not happen until the system state stays at the equilibrium point, for the points \hat{x}_{t_j} , $j \in \{1, \dots, N_1\}$, are randomly selected. In other words, there at least exists a $j_0 \in \{1, \dots, N_1\}$ such that $e_{1j_0} \neq 0$ during the critic NN learning process. So, there is a contradiction. Hence, the second term given in (7.2.36) guarantees that $\dot{\hat{W}}_c \neq 0$ during the critic NN learning process.

- (c) The last term in (7.2.36) is employed to ensure stability of the closed-loop system while the critic NN learns optimal weights. Consider system (7.2.21) with the control given in (7.2.32). In this case, we denote the derivative of Lyapunov function candidate by Θ and it can be calculated as

$$\begin{aligned} \Theta &= \dot{L}_2(\hat{x}) = L_{2\hat{x}}^\top \dot{\hat{x}}(t) \\ &= L_{2\hat{x}}^\top (h(\hat{x}) + g(\hat{x})\hat{u}) \\ &= L_{2\hat{x}}^\top \left(h(\hat{x}) - \frac{1}{2} g(\hat{x}) R^{-1} g^\top(\hat{x}) \nabla \sigma^\top \hat{W}_c \right) \\ &= L_{2\hat{x}}^\top \left(h(\hat{x}) - \frac{1}{2} G(\hat{x}) \nabla \sigma^\top \hat{W}_c \right) \end{aligned} \quad (7.2.38)$$

where $G(\hat{x})$ is defined in (7.2.33). If the closed-loop system is unstable, then there exists $\Theta > 0$. To keep the closed-loop system stable (i.e., $\Theta < 0$), we calculate the gradient of Θ as

$$-\eta_1 \frac{\partial \Theta}{\partial \hat{W}_c} = -\eta_1 \frac{\partial \left[L_{2\hat{x}}^\top \left(h(\hat{x}) - \frac{1}{2} G(\hat{x}) \nabla \sigma^\top \hat{W}_c \right) \right]}{\partial \hat{W}_c} = \frac{\eta_1}{2} \nabla \sigma G(\hat{x}) L_{2\hat{x}}. \quad (7.2.39)$$

Equation (7.2.39) shows the reason that we employ the last term of (7.2.36). In fact, observing the definition of $\Pi(\hat{x}, \hat{u})$ given in (7.2.37), we find that if system (7.2.21) is stable (i.e., $\Theta < 0$), then $\Pi(\hat{x}, \hat{u}) = 0$ and the last term given in (7.2.36) disappears. If system (7.2.21) is unstable, then $\Pi(\hat{x}, \hat{u}) = 1$ and the last term given in (7.2.36) is activated. Due to the existence of the last term of (7.2.36), it does not require an initial stabilizing control law for system (7.2.21). The property shall be shown in the subsequent simulation study.

By Remark 7.2.3, the set $\{\bar{\phi}_{(j)}\}_1^{N_1}$ should be linearly independent. Nevertheless, it is not an easy task to directly check this condition. Hence, we introduce a lemma as follows.

Lemma 7.2.1 *If the set $\{\sigma(\hat{x}_{t_j})\}_1^{N_1}$ is linearly independent and $\hat{u}(\hat{x})$ stabilizes system (7.2.21), then the following set*

$$\{\nabla\sigma_{(j)}(h(\hat{x}_{t_j}) + g(\hat{x}_{t_j})\hat{u})\}_1^{N_1}$$

is also linearly independent.

Since the proof is similar to [2], we omit it here.

Notice that

$$\bar{\phi}_{(j)} = \frac{\phi(\hat{x}_{t_j})}{(1 + \phi^\top(\hat{x}_{t_j})\phi(\hat{x}_{t_j}))^2},$$

where

$$\phi(\hat{x}_{t_j}) = \nabla\sigma_{(j)}(h(\hat{x}_{t_j}) + g(\hat{x}_{t_j})\hat{u}).$$

By Lemma 7.2.1, we find that if $\{\sigma(\hat{x}_{t_j})\}_1^{N_1}$ is linearly independent, then $\{\bar{\phi}_{(j)}\}_1^{N_1}$ is also linearly independent. That is, to ensure $\{\bar{\phi}_{(j)}\}_1^{N_1}$ to be linearly independent, the following condition should be satisfied.

Condition 7.2.1 Let $\mathfrak{D} = [\sigma(\hat{x}_{t_1}), \dots, \sigma(\hat{x}_{t_{N_1}})] \in \mathbb{R}^{N_1 \times N_1}$ be the recorded data matrix. There exist sufficiently large amount of recorded data such that \mathfrak{D} can be chosen nonsingular, that is, $\det \mathfrak{D} \neq 0$.

Remark 7.2.4 Condition 7.2.1 can be satisfied by selecting and recording data during the learning process of NNs over a finite time interval. Compared with the PE condition, a clear advantage of Condition 7.2.1 is that it can easily be checked online [5]. In addition, Condition 7.2.1 makes full use of history data, which can improve the speed of convergence of parameters. This feature will be also shown in the simulation study.

By the definition of ϕ given in (7.2.35) and using (7.2.28), (7.2.31) and (7.2.32), we have

$$\begin{aligned} \phi &= \nabla\sigma(h(\hat{x}) + g(\hat{x})\hat{u}(\hat{x})) \\ &= \nabla\sigma\left(h(\hat{x}) - \frac{1}{2}g(\hat{x})R^{-1}g^\top(\hat{x})\sigma^\top(W_c - \tilde{W}_c)\right) \\ &= \nabla\sigma\left(\xi(\hat{x}) + \frac{1}{2}G(\hat{x})\nabla\varepsilon_{N_1}\right) + \frac{1}{2}\nabla\sigma G(\hat{x})\nabla\sigma^\top\tilde{W}_c, \end{aligned} \quad (7.2.40)$$

where $\xi(\hat{x})$ is given in (7.2.34). From (7.2.31), (7.2.34), (7.2.36), and (7.2.40), we obtain

$$\begin{aligned}
\dot{\tilde{W}}_c &= \eta_1 \bar{\phi} \left(\mathcal{E}(\hat{x}) - \frac{1}{4} \hat{W}_c^\top \nabla \sigma G(\hat{x}) \nabla \sigma^\top \hat{W}_c \right) \\
&\quad + \eta_1 \sum_{j=1}^{N_1} \bar{\phi}_{(j)} \left(\mathcal{E}(\hat{x}_{t_j}) - \frac{1}{4} \hat{W}_c^\top \nabla \sigma_{(j)} G(\hat{x}_{t_j}) \nabla \sigma_{(j)}^\top \hat{W}_c \right) - \frac{\eta_1}{2} \Pi(\hat{x}, \hat{u}) \nabla \sigma G(\hat{x}) L_{2\hat{x}} \\
&= \eta_1 \frac{\phi}{m_s^2} \left(\hat{W}_c^\top \nabla \sigma h(\hat{x}) + Q(\hat{x}) - \frac{1}{4} \hat{W}_c^\top \nabla \sigma G(\hat{x}) \nabla \sigma^\top \hat{W}_c \right) \\
&\quad + \eta_1 \sum_{j=1}^{N_1} \frac{\phi_{(j)}}{m_s^2 s_j} \left(\hat{W}_c^\top \nabla \sigma_{(j)} h(\hat{x}_{t_j}) + Q(\hat{x}_{t_j}) - \frac{1}{4} \hat{W}_c^\top \nabla \sigma_{(j)} G(\hat{x}_{t_j}) \nabla \sigma_{(j)}^\top \hat{W}_c \right) \\
&\quad - \frac{\eta_1}{2} \Pi(\hat{x}, \hat{u}) \nabla \sigma G(\hat{x}) L_{2\hat{x}} \\
&= \eta_1 \frac{\phi}{m_s^2} \left(W_c^\top \nabla \sigma h(\hat{x}) + Q(\hat{x}) - \frac{1}{4} W_c^\top \nabla \sigma G(\hat{x}) \nabla \sigma^\top W_c \right. \\
&\quad \left. - \tilde{W}_c^\top \nabla \sigma h(\hat{x}) - \frac{1}{4} \tilde{W}_c^\top \nabla \sigma G(\hat{x}) \nabla \sigma^\top \tilde{W}_c + \frac{1}{2} \tilde{W}_c^\top \nabla \sigma G(\hat{x}) \nabla \sigma^\top W_c \right) \\
&\quad + \eta_1 \sum_{j=1}^{N_1} \frac{\phi_{(j)}}{m_s^2 s_j} \left(W_c^\top \nabla \sigma_{(j)} h(\hat{x}_{t_j}) + Q(\hat{x}_{t_j}) - \frac{1}{4} W_c^\top \nabla \sigma_{(j)} G(\hat{x}_{t_j}) \nabla \sigma_{(j)}^\top W_c \right) \\
&\quad - \tilde{W}_c^\top \nabla \sigma_{(j)} h(\hat{x}_{t_j}) - \frac{1}{4} \tilde{W}_c^\top \nabla \sigma_{(j)} G(\hat{x}_{t_j}) \nabla \sigma_{(j)}^\top \tilde{W}_c + \frac{1}{2} \tilde{W}_c^\top \nabla \sigma_{(j)} G(\hat{x}_{t_j}) \nabla \sigma_{(j)}^\top W_c \\
&\quad - \frac{\eta_1}{2} \Pi(\hat{x}, \hat{u}) \nabla \sigma G(\hat{x}) L_{2\hat{x}} \\
&= - \frac{\eta_1}{m_s^2} \left[\nabla \sigma \left(\xi(\hat{x}) + \frac{1}{2} G(\hat{x}) \nabla \varepsilon_{N_1} \right) + \frac{1}{2} \nabla \sigma G(\hat{x}) \nabla \sigma^\top \tilde{W}_c \right] \\
&\quad \times \left[\tilde{W}_c^\top \nabla \sigma h(\hat{x}) + \tilde{W}_c^\top \nabla \sigma g(\hat{x}) \left(u^*(\hat{x}) + \frac{1}{2} R^{-1} g^\top(\hat{x}) \nabla \varepsilon_{N_1} \right) \right. \\
&\quad \left. + \frac{1}{4} \tilde{W}_c^\top \nabla \sigma G(\hat{x}) \nabla \sigma^\top \tilde{W}_c + \varepsilon_{\text{HJB}} \right] \\
&\quad - \sum_{j=1}^{N_1} \frac{\eta_1}{m_s^2 s_j} \left[\nabla \sigma_{(j)} \left(\xi(\hat{x}_{t_j}) + \frac{1}{2} G(\hat{x}_{t_j}) \nabla \varepsilon_{N_1(j)} \right) + \frac{1}{2} \nabla \sigma_{(j)} G(\hat{x}_{t_j}) \nabla \sigma_{(j)}^\top \tilde{W}_c \right] \\
&\quad \times \left[\tilde{W}_c^\top \nabla \sigma_{(j)} h(\hat{x}_{t_j}) + \tilde{W}_c^\top \nabla \sigma_{(j)} g(\hat{x}_{t_j}) \left(u^*(\hat{x}_{t_j}) + \frac{1}{2} R^{-1} g^\top(\hat{x}_{t_j}) \nabla \varepsilon_{N_1(j)} \right) \right. \\
&\quad \left. + \frac{1}{4} \tilde{W}_c^\top \nabla \sigma_{(j)} G(\hat{x}_{t_j}) \nabla \sigma_{(j)}^\top \tilde{W}_c + \varepsilon_{\text{HJB}} \right] \\
&\quad - \frac{\eta_1}{2} \Pi(\hat{x}, \hat{u}) \nabla \sigma G(\hat{x}) L_{2\hat{x}} \\
&= - \frac{\eta_1}{m_s^2} \left(\nabla \sigma \zeta(\hat{x}) + \frac{1}{2} \overline{G}(\hat{x}) \tilde{W}_c \right) \left(\tilde{W}_c^\top \nabla \sigma \zeta(\hat{x}) + \frac{1}{4} \tilde{W}_c^\top \overline{G}(\hat{x}) \tilde{W}_c + \varepsilon_{\text{HJB}} \right)
\end{aligned}$$

$$\begin{aligned}
& - \sum_{j=1}^{N_1} \frac{\eta_1}{m_{\hat{s}_j}^2} \left(\nabla \sigma_{(j)} \zeta(\hat{x}_{t_j}) + \frac{1}{2} \overline{G}(\hat{x}_{t_j}) \tilde{W}_c \right) \left(\tilde{W}_c^\top \nabla \sigma_{(j)} \zeta(\hat{x}_{t_j}) \right. \\
& \left. + \frac{1}{4} \tilde{W}_c^\top \overline{G}(\hat{x}_{t_j}) \tilde{W}_c + \varepsilon_{\text{HJB}} \right) - \frac{\eta_1}{2} \Pi(\hat{x}, \hat{u}) \nabla \sigma G(\hat{x}) L_{2\hat{x}}, \tag{7.2.41}
\end{aligned}$$

where

$$\zeta(\hat{x}) = \xi(\hat{x}) + \frac{1}{2} G(\hat{x}) \nabla \varepsilon_{N_1}, \quad \overline{G}(\hat{x}) = \nabla \sigma G(\hat{x}) \nabla \sigma^\top, \quad \overline{G}(\hat{x}_{t_j}) = \nabla \sigma_{(j)} G(\hat{x}_{t_j}) \nabla \sigma_{(j)}^\top.$$

The schematic diagram of the present control algorithm is shown in Fig. 7.1.

7.2.2 Stability Analysis of Closed-Loop System

We start with an assumption and then present the stability analysis result.

Assumption 7.2.6 There exist known constants $b_\sigma > 0$ and $\varepsilon_{b_1} > 0$ such that $\|\nabla \sigma(\hat{x})\| < b_\sigma$ and $\|\nabla \varepsilon_{N_1}(\hat{x})\| < \varepsilon_{b_1}$ for every $\hat{x} \in \Omega$.

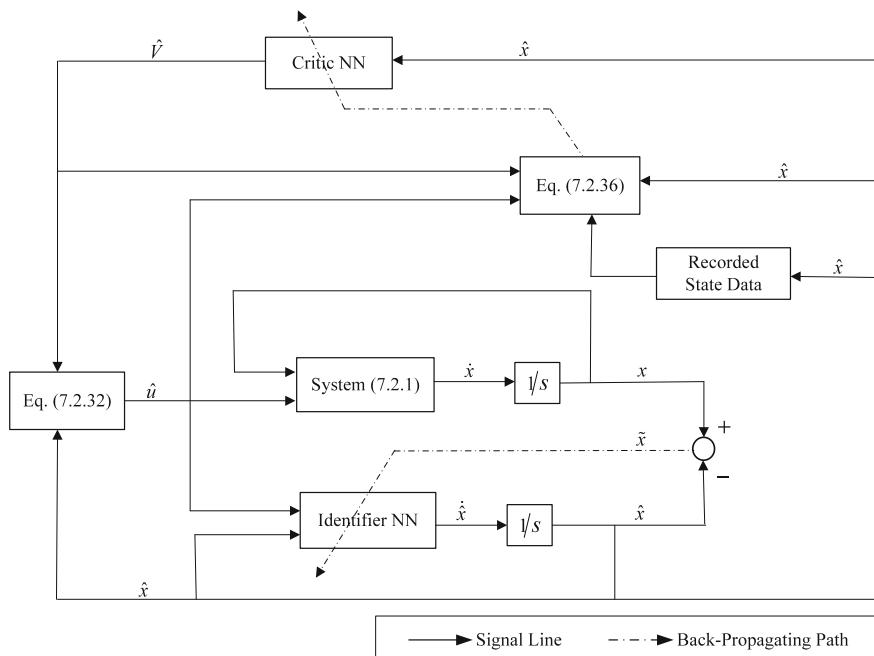


Fig. 7.1 The schematic diagram of the identifier-critic architecture

Theorem 7.2.2 *Given the input-affine nonlinear dynamics described by (7.2.1) with associated HJB equation (7.2.25), let Assumptions 7.2.1–7.2.6 hold and take the control input for system (7.2.1) as in (7.2.32). Moreover, let weight update laws of the identifier NN be (7.2.8) and (7.2.9), and let weight tuning rule for the critic NN be (7.2.36). Then, the identification error $\tilde{x}(t)$, and the weight estimation errors \tilde{W}_m , \tilde{Y}_m , and \tilde{W}_c are all UUB.*

Proof Consider the Lyapunov function candidate

$$L(x) = L_1(x) + L_2(\hat{x}) + \frac{1}{2} \tilde{W}_c^\top \eta_1^{-1} \tilde{W}_c, \quad (7.2.42)$$

where $L_1(x)$ is defined in (7.2.12) and $L_2(\hat{x})$ is defined as in Assumption 7.2.5.

Taking the time derivative of (7.2.42) using (7.2.19) and (7.2.38), we derive

$$\begin{aligned} \dot{L}(x) &= \dot{L}_1(x) + \dot{L}_2(\hat{x}) + \tilde{W}_c^\top \eta_1^{-1} \dot{\tilde{W}}_c \\ &\leq -\left(\frac{\theta}{2} \|\tilde{x}\| - \varsigma\right) \|\tilde{x}\| + L_{2\hat{x}}^\top \left(h(\hat{x}) - \frac{1}{2} G(\hat{x}) \nabla \sigma^\top \hat{W}_c\right) \\ &\quad + \tilde{W}_c^\top \eta_1^{-1} \dot{\tilde{W}}_c, \end{aligned} \quad (7.2.43)$$

where ς is given in (7.2.20). Using (7.2.41), the last term of (7.2.43) is developed as

$$\tilde{W}_c^\top \eta_1^{-1} \dot{\tilde{W}}_c = \mathfrak{N}_1 + \mathfrak{N}_2 - \frac{1}{2} \tilde{W}_c^\top \Pi(\hat{x}, \hat{u}) \nabla \sigma G(\hat{x}) L_{2\hat{x}}, \quad (7.2.44)$$

where

$$\begin{aligned} \mathfrak{N}_1 &= -\frac{1}{m_s^2} \left(\tilde{W}_c^\top \nabla \sigma \zeta(\hat{x}) + \frac{1}{2} \tilde{W}_c^\top \bar{G}(\hat{x}) \tilde{W}_c \right) \\ &\quad \times \left(\tilde{W}_c^\top \nabla \sigma \zeta(\hat{x}) + \frac{1}{4} \tilde{W}_c^\top \bar{G}(\hat{x}) \tilde{W}_c + \varepsilon_{\text{HJB}} \right), \end{aligned}$$

$$\begin{aligned} \mathfrak{N}_2 &= -\sum_{j=1}^{N_1} \frac{1}{m_{s_j}^2} \left(\tilde{W}_c^\top \nabla \sigma_{(j)} \zeta(\hat{x}_{t_j}) + \frac{1}{2} \tilde{W}_c^\top \bar{G}(\hat{x}_{t_j}) \tilde{W}_c \right) \\ &\quad \times \left(\tilde{W}_c^\top \nabla \sigma_{(j)} \zeta(\hat{x}_{t_j}) + \frac{1}{4} \tilde{W}_c^\top \bar{G}(\hat{x}_{t_j}) \tilde{W}_c + \varepsilon_{\text{HJB}} \right). \end{aligned}$$

Now, we consider the first term \mathfrak{N}_1 . We have

$$\begin{aligned}\mathfrak{N}_1 = & -\frac{1}{m_s^2} \left\{ \left(\tilde{W}_c^\top \nabla \sigma \zeta(\hat{x}) \right)^2 + \frac{1}{8} \left(\tilde{W}_c^\top \bar{G}(\hat{x}) \tilde{W}_c \right)^2 \right. \\ & + \frac{3}{4} \left(\tilde{W}_c^\top \nabla \sigma \zeta(\hat{x}) \right) \left(\tilde{W}_c^\top \bar{G}(\hat{x}) \tilde{W}_c \right) \\ & \left. + \tilde{W}_c^\top \nabla \sigma \mathcal{L}(\hat{x}) \varepsilon_{\text{HJB}} + \frac{1}{2} \tilde{W}_c^\top \bar{G}(\hat{x}) \tilde{W}_c \varepsilon_{\text{HJB}} \right\}. \quad (7.2.45)\end{aligned}$$

Applying (7.2.18) (with $v = 1$) to the last three terms of (7.2.45), it follows

$$\begin{aligned}\mathfrak{N}_1 = & -\frac{1}{m_s^2} \left\{ \frac{1}{2} \left(3 \tilde{W}_c^\top \nabla \sigma \zeta(\hat{x}) + \frac{1}{4} \tilde{W}_c^\top \bar{G}(\hat{x}) \tilde{W}_c \right)^2 \right. \\ & + \frac{1}{2} \left(\tilde{W}_c^\top \nabla \sigma \zeta(\hat{x}) + \varepsilon_{\text{HJB}} \right)^2 + \frac{1}{2} \left(\frac{1}{4} \tilde{W}_c^\top \bar{G}(\hat{x}) \tilde{W}_c + 2 \varepsilon_{\text{HJB}} \right)^2 \\ & \left. - 4 \left(\tilde{W}_c^\top \nabla \sigma \zeta(\hat{x}) \right)^2 + \frac{1}{16} \left(\tilde{W}_c^\top \bar{G}(\hat{x}) \tilde{W}_c \right)^2 - \frac{5}{2} \varepsilon_{\text{HJB}}^2 \right\} \\ \leq & -\frac{1}{m_s^2} \left\{ \frac{1}{16} \left(\tilde{W}_c^\top \bar{G}(\hat{x}) \tilde{W}_c \right)^2 - 4 \left(\tilde{W}_c^\top \nabla \sigma \zeta(\hat{x}) \right)^2 - \frac{5}{2} \varepsilon_{\text{HJB}}^2 \right\}. \quad (7.2.46)\end{aligned}$$

Similarly, we have

$$\mathfrak{N}_2 \leq - \sum_{j=1}^{N_1} \frac{1}{m_{s_j}^2} \left\{ \frac{1}{16} \left(\tilde{W}_c^\top \bar{G}(\hat{x}_{t_j}) \tilde{W}_c \right)^2 - 4 \left(\tilde{W}_c^\top \nabla \sigma_{(j)} \zeta(\hat{x}_{t_j}) \right)^2 - \frac{5}{2} \varepsilon_{\text{HJB}}^2 \right\}. \quad (7.2.47)$$

Noticing that the PE condition guarantees

$$m_s = 1 + \phi^\top \phi$$

to be bounded, there exists a positive constant τ_0 such that

$$\tau_0 \leq \frac{1}{m_s^2} = \frac{1}{(1 + \phi^\top \phi)^2} \leq 1$$

and

$$\tau_0 \leq \frac{1}{m_{s_j}^2} = \frac{1}{(1 + \phi(x_{t_j})^\top \phi(x_{t_j}))^2} \leq 1.$$

Substituting (7.2.46) and (7.2.47) into (7.2.44), we obtain

$$\begin{aligned}\tilde{W}_c^\top \eta_1^{-1} \dot{\tilde{W}}_c \leq & -\frac{1}{16} \left\{ \sum_{j=1}^{N_1} \frac{1}{m_{s_j}^2} \left(\tilde{W}_c^\top \bar{G}(\hat{x}_{t_j}) \tilde{W}_c \right)^2 + \frac{1}{m_s^2} \left(\tilde{W}_c^\top \bar{G}(\hat{x}) \tilde{W}_c \right)^2 \right\} \\ & + 4 \left\{ \sum_{j=1}^{N_1} \frac{1}{m_{s_j}^2} \left(\tilde{W}_c^\top \nabla \sigma_{(j)} \zeta(\hat{x}_{t_j}) \right)^2 + \frac{1}{m_s^2} \left(\tilde{W}_c^\top \nabla \sigma \zeta(\hat{x}) \right)^2 \right\}\end{aligned}$$

$$\begin{aligned}
& + \frac{5}{2} \left(\frac{1}{m_s^2} + \sum_{j=1}^{N_1} \frac{1}{m_{s_j}^2} \right) \varepsilon_{\text{HJB}}^2 - \frac{1}{2} \tilde{W}_c^\top \Pi(\hat{x}, \hat{u}) \nabla \sigma G(\hat{x}) L_{2\hat{x}} \\
& \leq - \frac{\tau_0}{16} \left\{ \sum_{j=1}^{N_1} \mu_{\text{inf}}^2(\bar{G}(\hat{x}_{t_j})) + \mu_{\text{inf}}^2(\bar{G}(\hat{x})) \right\} \|\tilde{W}_c\|^4 \\
& \quad + 4b_\sigma^2 \left\{ \sum_{j=1}^{N_1} \vartheta_{\text{sup}}^2(\zeta(\hat{x}_{t_j})) + \vartheta_{\text{sup}}^2(\zeta(\hat{x})) \right\} \|\tilde{W}_c\|^2 \\
& \quad + \frac{5}{2} (N_1 + 1) \varepsilon_{a_1}^2 - \frac{1}{2} \tilde{W}_c^\top \Pi(\hat{x}, \hat{u}) \nabla \sigma G(\hat{x}) L_{2\hat{x}}, \tag{7.2.48}
\end{aligned}$$

where $\mu_{\text{inf}}(\mathcal{Y})$ denotes the lower bound of \mathcal{Y} ($\mathcal{Y} = \bar{G}(\hat{x}_{t_j}), \bar{G}(\hat{x})$), $\vartheta_{\text{sup}}(\mathcal{Z})$ represents the upper bound of \mathcal{Z} ($\mathcal{Z} = \zeta(\hat{x}_{t_j}), \zeta(\hat{x})$), N_1 is the number of nodes in the hidden layer of the critic NN, and $\|\varepsilon_{\text{HJB}}\| \leq \varepsilon_{a_1}$ as assumed earlier in (7.2.29).

Combining (7.2.43) and (7.2.48), we derive

$$\begin{aligned}
\dot{L}(x) & \leq L_{2\hat{x}}^\top \left(h(\hat{x}) - \frac{1}{2} G(\hat{x}) \nabla \sigma^\top \tilde{W}_c \right) - \frac{1}{2} \tilde{W}_c^\top \Pi(\hat{x}, \hat{u}) \nabla \sigma G(\hat{x}) L_{2\hat{x}} \\
& \quad - \frac{\mathfrak{T}_1}{16} \|\tilde{W}_c\|^4 + 4\mathfrak{T}_2 \|\tilde{W}_c\|^2 - \frac{\theta}{2} \left(\|\tilde{x}\| - \frac{\xi}{\theta} \right)^2 + \frac{\xi^2}{2\theta} \\
& \quad + \frac{5}{2} (N_1 + 1) \varepsilon_{a_1}^2, \tag{7.2.49}
\end{aligned}$$

where

$$\mathfrak{T}_1 = \left[\mu_{\text{inf}}^2(\bar{G}(\hat{x})) + \sum_{j=1}^{N_1} \mu_{\text{inf}}^2(\bar{G}(\hat{x}_{t_j})) \right] \tau_0,$$

and

$$\mathfrak{T}_2 = b_\sigma^2 \vartheta_{\text{sup}}^2(\zeta(\hat{x})) + b_\sigma^2 \sum_{j=1}^{N_1} \vartheta_{\text{sup}}^2(\zeta(\hat{x}_{t_j})).$$

In view of the definition of $\Pi(\hat{x}, \hat{u})$ in (7.2.37), we divide (7.2.49) into the following two cases for discussion.

Case 1: $\Pi(\hat{x}, \hat{u}) = 0$. In this case, we can derive that the first term in (7.2.49) is negative. Observing the fact that $L_{2\hat{x}}^\top \dot{\hat{x}} < 0$ and by using the dense property of \mathbb{R} [28], we can conclude that there exists a constant $\tau > 0$ such that

$$L_{2\hat{x}}^\top \dot{\hat{x}} < -\tau \|L_{2\hat{x}}\| \leq 0.$$

Then, (7.2.49) becomes

$$\begin{aligned}\dot{L}(x) \leq & -\tau \|L_{2\hat{x}}\| - \frac{\theta}{2} \left(\|\tilde{x}\| - \frac{\varsigma}{\theta} \right)^2 \\ & - \frac{\mathfrak{T}_1}{16} \left(\|\tilde{W}_c\|^2 - \frac{32\mathfrak{T}_2}{\mathfrak{T}_1} \right)^2 + \frac{64\mathfrak{T}_2^2}{\mathfrak{T}_1} \\ & + \frac{1}{2\theta} [\varsigma^2 + 5\theta(N_1 + 1)\varepsilon_{a_1}^2].\end{aligned}\quad (7.2.50)$$

Hence, (7.2.50) yields $\dot{L}(x) < 0$ as long as one of the following conditions holds:

$$\|L_{2\hat{x}}\| > \frac{64\mathfrak{T}_2^2}{\tau\mathfrak{T}_1} + \frac{\varsigma^2 + 5\theta(N_1 + 1)\varepsilon_{a_1}^2}{2\tau\theta}, \quad (7.2.51)$$

or

$$\|\tilde{x}\| > \sqrt{\frac{128\mathfrak{T}_2^2}{\theta\mathfrak{T}_1} + \frac{\varsigma^2 + 5\theta(N_1 + 1)\varepsilon_{a_1}^2}{\theta^2}} + \frac{\varsigma}{\theta}, \quad (7.2.52)$$

or

$$\|\tilde{W}_c\| > \sqrt{\frac{32\mathfrak{T}_2}{\mathfrak{T}_1} + \frac{\sqrt{8\mathfrak{T}_1[\varsigma^2/\theta + 5(N_1 + 1)\varepsilon_{a_1}^2] + 1024\mathfrak{T}_2^2}}{\mathfrak{T}_1}}. \quad (7.2.53)$$

Case 2: $\Pi(\hat{x}, \hat{u}) = 1$. By the definition of $\Pi(\hat{x}, \hat{u})$ given in (7.2.37), we find that, in this case, the first term in (7.2.49) is nonnegative which implies that the control (7.2.32) may not stabilize system (7.2.21). Then, using (7.2.29) and (7.2.34), (7.2.49) becomes

$$\begin{aligned}\dot{L}(x) \leq & L_{2\hat{x}}^T \left(\xi(\hat{x}) + \frac{1}{2} G(\hat{x}) \nabla \varepsilon_{N_1} \right) - \frac{\theta}{2} \left(\|\tilde{x}\| - \frac{\varsigma}{\theta} \right)^2 + \frac{\varsigma^2}{2\theta} \\ & - \frac{\mathfrak{T}_1}{16} \left(\|\tilde{W}_c\|^2 - \frac{32\mathfrak{T}_2}{\mathfrak{T}_1} \right)^2 + \frac{64\mathfrak{T}_2^2}{\mathfrak{T}_1} + \frac{5}{2}(N_1 + 1)\varepsilon_{a_1}^2.\end{aligned}\quad (7.2.54)$$

Using Assumptions 7.2.5 and 7.2.6, (7.2.54) can be rewritten as

$$\begin{aligned}\dot{L}(x) \leq & -\lambda_{\min}(\Lambda_2(\hat{x})) \left(\|L_{2\hat{x}}\| - \frac{\varepsilon_{b_1} \vartheta_{\sup}(G(\hat{x}))}{4\lambda_{\min}(\Lambda_2(\hat{x}))} \right)^2 \\ & - \frac{\theta}{2} \left(\|\tilde{x}\| - \frac{\varsigma}{\theta} \right)^2 - \frac{\mathfrak{T}_1}{16} \left(\|\tilde{W}_c\|^2 - \frac{32\mathfrak{T}_2}{\mathfrak{T}_1} \right)^2 \\ & + \frac{\varepsilon_{b_1}^2 \vartheta_{\sup}^2(G(\hat{x}))}{16\lambda_{\min}(\Lambda_2(\hat{x}))} + \frac{64\mathfrak{T}_2^2}{\mathfrak{T}_1} + \frac{1}{2\theta} [\varsigma^2 + 5\theta(N_1 + 1)\varepsilon_{a_1}^2],\end{aligned}\quad (7.2.55)$$

where $\vartheta_{\sup}(\cdot)$ is defined as in (7.2.48).

Thus, (7.2.55) implies $\dot{L}(x) < 0$ as long as one of the following conditions holds:

$$\|L_{2\hat{x}}\| > \frac{\varepsilon_{b_1} \vartheta_{\sup}(G(\hat{x}))}{4\lambda_{\min}(A_2(\hat{x}))} + \sqrt{\frac{d}{\lambda_{\min}(A_2(\hat{x}))}}, \quad (7.2.56)$$

or

$$\|\tilde{x}\| > \sqrt{\frac{2d}{\theta}} + \frac{\varsigma}{\theta}, \quad (7.2.57)$$

or

$$\|\tilde{W}_c\| > 2 \sqrt{\frac{8\mathfrak{T}_2}{\mathfrak{T}_1} + \sqrt{\frac{d}{\mathfrak{T}_1}}}, \quad (7.2.58)$$

where

$$d = \frac{\varepsilon_{b_1}^2 \vartheta_{\sup}^2(G(\hat{x}))}{16\lambda_{\min}(A_2(\hat{x}))} + \frac{64\mathfrak{T}_2^2}{\mathfrak{T}_1} + \frac{1}{2\theta} [\varsigma^2 + 5\theta(N_1 + 1)\varepsilon_{a_1}^2].$$

Combining Case 1 and Case 2 and using the standard Lyapunov's extension theorem [12, 13] (or the Lagrange stability result [22]), one can conclude that the state identification error $\tilde{x}(t)$ and NN weight estimation errors \tilde{W}_m , \tilde{Y}_m , and \tilde{W}_c are all UUB. This completes the proof.

Remark 7.2.5 The uniform ultimate boundedness of \tilde{W}_m and \tilde{Y}_m is obtained as follows: Inequalities (7.2.51)–(7.2.53) (or (7.2.56)–(7.2.58)) guarantee $\dot{L}(x) < 0$. Then, we can conclude that $L(x(t))$ is the strictly decreasing function with respect to t ($t \geq 0$). Hence, we can derive $L(x(t)) < L(x(0))$, where $L(x(0))$ is a bounded positive constant. Using $L(x)$ defined as in (7.2.42), we have $\frac{1}{2}\text{tr}(\tilde{W}_m^T l_1^{-1} \tilde{W}_m) + \frac{1}{2}\text{tr}(\tilde{Y}_m^T l_2^{-1} \tilde{Y}_m) < L(x(0))$. Using the definition of Frobenius matrix norm [9, 16], we obtain that \tilde{W}_m and \tilde{Y}_m are UUB.

7.2.3 Simulation Study

Consider the continuous-time input-affine nonlinear systems described by

$$\dot{x} = f(x) + g(x)u, \quad (7.2.59)$$

where

$$f(x) = \begin{bmatrix} -x_1 + x_2 \\ -0.5x_1 - 0.5x_2 + 0.5x_2[\cos(2x_1) + 2]^2 \end{bmatrix}, \quad g(x) = \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix}.$$

The value function is defined as in (7.2.2), where Q and R are chosen as identity matrices of appropriate dimensions. The prior knowledge of $f(x)$ is assumed to be unavailable. To obtain the knowledge of system dynamics, an identifier NN given in (7.2.6) is employed. The gains for the identifier NN are selected as

$$A = \begin{bmatrix} -1 & 1 \\ -0.5 & -0.5 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ -0.5 & 0 \end{bmatrix},$$

$$l_1 = 20, \quad l_2 = 10, \quad \tau_1 = 6.1, \quad \tau_2 = 15,$$

$$N_0 = 8,$$

and the gain for the critic NN is selected as $\eta_1 = 2.5$. The activation function of the critic NN is chosen with $N_1 = 3$ neurons with $\sigma(x) = [x_1^2, x_2^2, x_1 x_2]^\top$, and the critic NN weight is denoted as $\hat{W}_c = [\hat{W}_{c1}, \hat{W}_{c2}, \hat{W}_{c3}]^\top$.

Remark 7.2.6 It is significant to emphasize that the number of neurons required for any particular application is still an open problem. Selecting the proper number of neurons for NNs is more of an art than science [8, 27]. In this example, the number of neurons is obtained by computer simulations. We find that selecting eight neurons in the hidden layer for the identifier NN can lead to satisfactory simulation results. Meanwhile, in order to compare our algorithm with the algorithms proposed in [3, 29], we choose three neurons in the hidden layer for the critic NN, and the simulation results are satisfied.

The initial weights \hat{W}_m and \hat{Y}_m for the identifier NN are selected randomly within an interval of $[-10, 10]$ and $[-5, 5]$, respectively. Meanwhile, the initial weights of the critic NN are chosen to be zeros, and the initial system state is set to $x_0 = [3.5, -3.5]^\top$. In this circumstance, the initial control cannot stabilize system (7.2.59). In the present algorithm, no initial stabilizing control is required. Moreover, by using the method proposed in [5, 6], the recorded data can easily be made qualified for Condition 7.2.1.

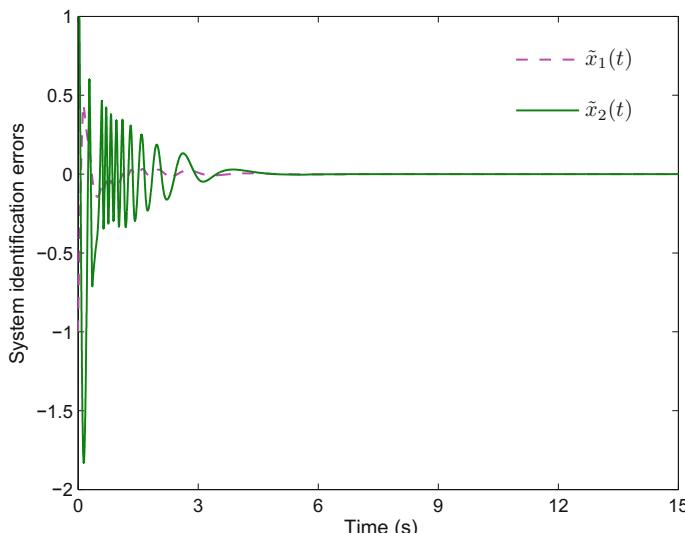


Fig. 7.2 System identification errors $\tilde{x}_1(t)$ and $\tilde{x}_2(t)$

Computer simulation results are shown in Figs. 7.3, 7.4, 7.5, 7.6, 7.7 and 7.8. Figure 7.2 illustrates the performance of the system identification errors $\tilde{x}_1(t)$ and $\tilde{x}_2(t)$. Figure 7.3 shows the trajectories of system states $x(t)$. Figure 7.4 shows the Frobenius norm of the weights $\|\hat{W}_m\|$ and $\|\hat{Y}_m\|$ of the identifier NN. Figure 7.5 shows the convergence of the critic NN weights. Figure 7.6 shows the control u . Figure 7.7 illustrates the system states without considering the third term in (7.2.36), where the system becomes unstable.

To compare with [7], we use Fig. 7.8 to show the system states with the algorithm proposed in [7]. It should be mentioned that the PE condition is necessary in [7]. To guarantee the PE condition, a small exploratory signal $\mathcal{N}(t) = \sin^5(t) \cos(t) + \sin^5(2t) \cos(0.1t)$ is added to the control u for the first 9 s. That is, Fig. 7.8 is obtained with the exploratory signal added to the control input during the first 9 s. In addition, it is worth pointing out that by using the methods proposed in [3] and [29] and employing the small exploratory signal $\mathcal{N}(t)$, one can also get stable system states. We omit the simulation results here since the results are similar to Fig. 7.8 [3, 29]. It is quite straightforward to notice that the trajectories of system states given in [3, 29] share common feature with Fig. 7.8, which is oscillatory before it converges to the equilibrium point. This is caused by the PE exploratory signal.

Several notes about the simulation results are listed as follows.

- From Fig. 7.2, it is observed that the identifier NN can approximate the real system very fast and well.
- From Fig. 7.3, one can observe that there are almost no oscillations of system state. As aforementioned, the PE signal always leads to oscillations of the system states

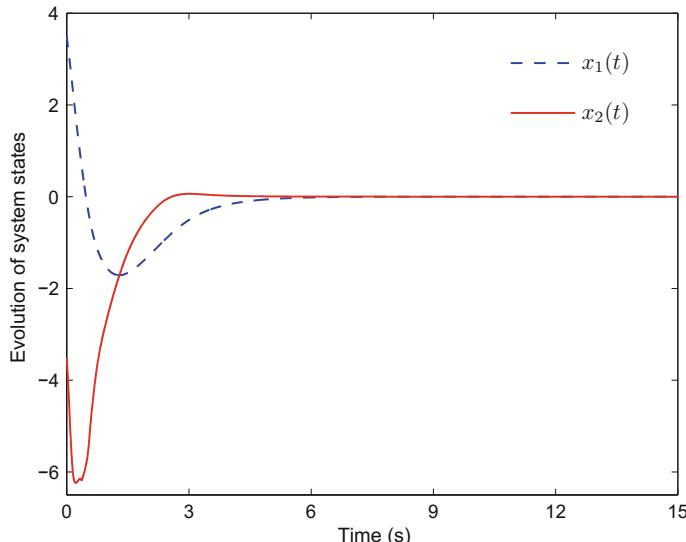


Fig. 7.3 Evolution of system state $x(t)$

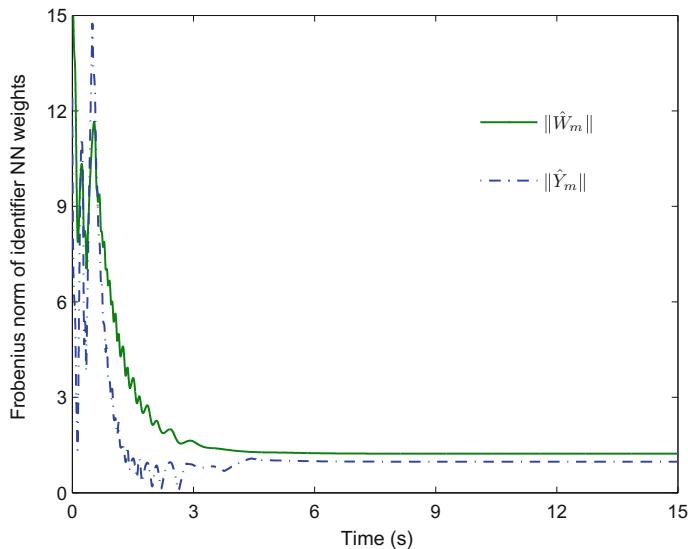


Fig. 7.4 Frobenius norm of identifier NN weights $\|\hat{W}_m\|$ and $\|\hat{Y}_m\|$

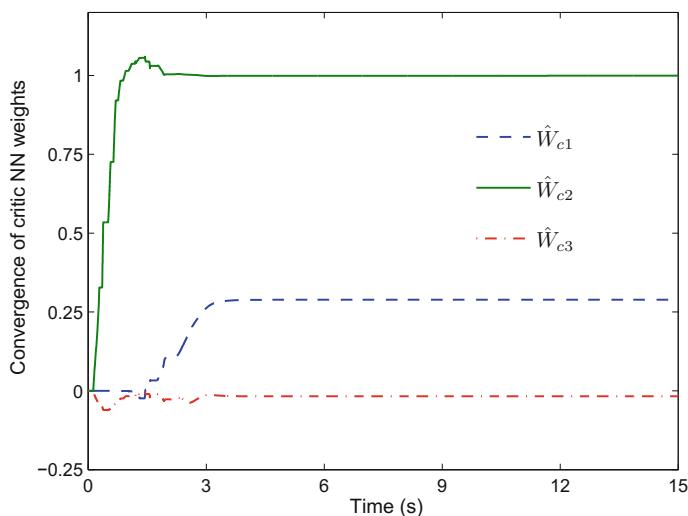


Fig. 7.5 Convergence of critic NN weights

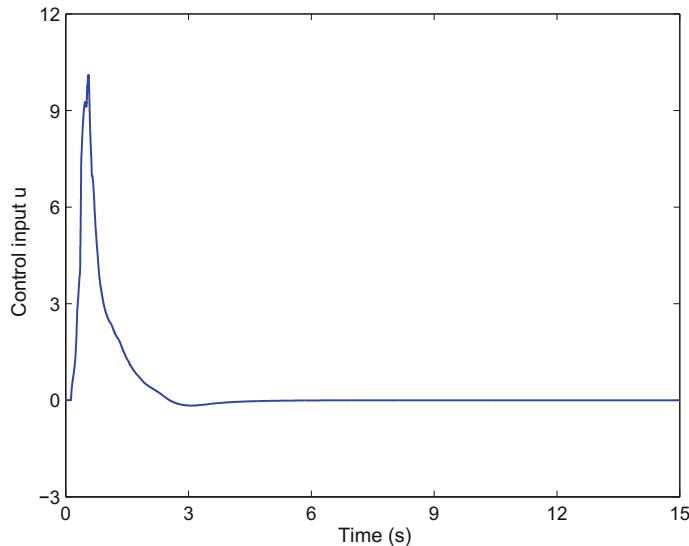


Fig. 7.6 Control input u corresponding to Fig. 7.3

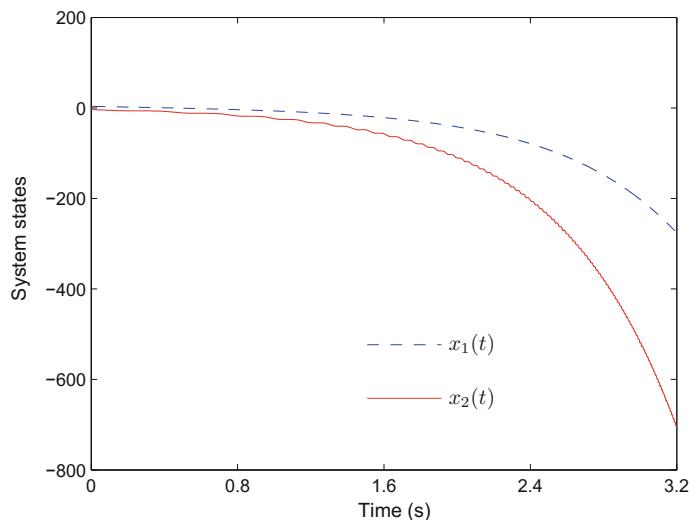


Fig. 7.7 Trajectories of states without considering the third term in (7.2.36)

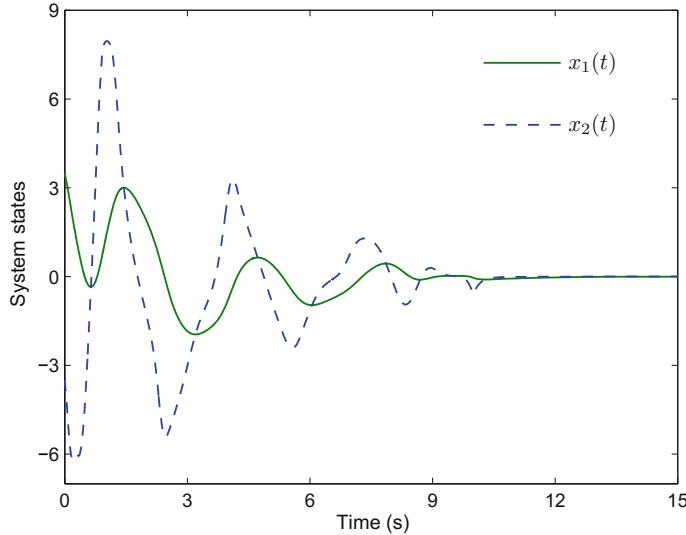


Fig. 7.8 System states with the algorithm proposed in [7]

(see Fig. 7.8 and the simulation results presented in [3, 7, 29]). This verifies that the restrictive PE condition is relaxed by using recorded and instantaneous data simultaneously. Hence, an advantage of the present algorithm as compared to the methods in [3, 7, 29] lies in that the PE condition is relaxed.

- From Figs. 7.4 and 7.5, we can find that the identifier NN and the critic NN are tuned simultaneously. Meanwhile, the estimated weights of the identifier NN and the critic NN are all guaranteed to be UUB.
- From Figs. 7.5 and 7.6, one can find that the initial weights of the critic NN and the initial control are all zeros. In this circumstance, the initial control cannot stabilize system (7.2.59). Nevertheless, the initial control must stabilize the system in [3, 29]. Therefore, in comparison with the methods proposed in [3, 29], a distinct advantage of the algorithm developed in this section lies in that the initial stabilizing control is not required any more.
- From Fig. 7.7, one shall find that without the third term in (7.2.36), the system is unstable during the learning process of the critic NN.

7.3 Online Optimal Control of Affine Nonlinear Systems with Constrained Inputs

Consider the continuous-time nonlinear systems described by the form

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t)) \quad (7.3.1)$$

with the state $x(t) \in \mathbb{R}^n$ and the control input $u(t) \in \mathfrak{U} \subset \mathbb{R}^m$, $\mathfrak{U} = \{u \in \mathbb{R}^m : |u_i| \leq \tau, i = 1, \dots, m\}$, where $\tau > 0$ is the saturation bound, and $f(x(t)) \in \mathbb{R}^n$ and $g(x(t)) \in \mathbb{R}^{n \times m}$ are known continuous functions of the state $x(t)$. It is assumed that $f(0) = 0$ and $f(x) + g(x)u$ is Lipschitz continuous on Ω containing the origin, such that the solution of system (7.3.1) is unique for any given initial state $x_0 \in \Omega$ and control $u \in \mathfrak{U}$. System (7.3.1) is stable in the sense that there exists a continuous control $u \in \mathfrak{U}$ such that the system is UUB over Ω .

The value function for system (7.3.1) is defined by

$$V(x(t)) = \int_t^\infty (Q(x(s)) + \mathcal{Y}(u(s))) ds, \quad (7.3.2)$$

where $Q(x)$ is continuously differentiable and positive definite, i.e., $Q(x) \in \mathcal{C}^1(\Omega)$, $Q(x) > 0$ for all $x \neq 0$ and $x = 0 \Leftrightarrow Q(x) = 0$, and $\mathcal{Y}(u)$ is positive definite. To confront bounded controls in system (7.3.1) and inspired by the work of [1, 20, 24], we define $\mathcal{Y}(u)$ as

$$\mathcal{Y}(u) = 2\tau \int_0^u \Psi^{-\top}(\nu/\tau) d\nu = 2\tau \sum_{i=1}^m \int_0^{u_i} \psi^{-1}(\nu_i/\tau) d\nu_i,$$

where $\Psi^{-1}(\nu/\tau) = (\psi^{-1}(\nu_1/\tau), \psi^{-1}(\nu_2/\tau), \dots, \psi^{-1}(\nu_m/\tau))^\top$, $\Psi \in \mathbb{R}^m$, and $\Psi^{-\top}$ denotes $(\Psi^{-1})^\top$. Meanwhile, $\psi(\cdot)$ is a strictly monotonic odd function satisfying $|\psi(\cdot)| < 1$ and belonging to \mathcal{C}^p ($p \geq 1$) and $\mathcal{L}_2(\Omega)$. It is significant to state that $\mathcal{Y}(u)$ is positive definite since $\psi^{-1}(\cdot)$ is a monotonic odd function. Without loss of generality, in this section, we choose $\psi(\cdot) = \tanh(\cdot)$.

Given a control $u(x) \in \mathcal{A}(\Omega)$, if the associated value function $V(x) \in \mathcal{C}^1(\Omega)$, the infinitesimal version of (7.3.2) is the so-called Lyapunov equation

$$V_x^\top(f(x) + g(x)u) + Q(x) + 2\tau \int_0^u \tanh^{-\top}(\nu/\tau) d\nu = 0,$$

where $V_x \in \mathbb{R}^n$ denotes the partial derivative of $V(x)$ with respect to x . Define the Hamiltonian for the control $u(x) \in \mathcal{A}(\Omega)$ and the associated value function $V(x)$ as

$$H(x, V_x, u) = V_x^\top(f(x) + g(x)u) + Q(x) + 2\tau \int_0^u \tanh^{-\top}(\nu/\tau) d\nu.$$

The optimal value $V^*(x)$ can be obtained by solving the HJB equation

$$\min_{u(x) \in \mathcal{A}(\Omega)} H(x, V_x^*, u) = H(x, V_x^*, u^*) = 0. \quad (7.3.3)$$

The closed-form expression for constrained optimal control is

$$u^*(x) = -\tau \tanh\left(\frac{1}{2\tau} g^T(x) V_x^*\right). \quad (7.3.4)$$

Substituting (7.3.4) into (7.3.3), the HJB equation for constrained nonlinear systems becomes

$$\begin{aligned} (V_x^*)^T f(x) - 2\tau^2 \mathfrak{C}^T(x) \tanh(\mathfrak{C}(x)) + Q(x) \\ + 2\tau \int_0^{-\tau \tanh(\mathfrak{C}(x))} \tanh^{-T}(v/\tau) dv = 0, \end{aligned} \quad (7.3.5)$$

where $\mathfrak{C}(x) = \frac{1}{2\tau} g^T(x) V_x^*$. Using the integration formulas of inverse hyperbolic functions, we note that

$$\begin{aligned} 2\tau \int_0^{-\tau \tanh(\mathfrak{C}(x))} \tanh^{-T}(v/\tau) dv \\ = 2\tau \sum_{i=1}^m \int_0^{-\tau \tanh(\mathfrak{C}_i(x))} \tanh^{-T}(v_i/\tau) dv_i \\ = 2\tau^2 \mathfrak{C}^T(x) \tanh(\mathfrak{C}(x)) + \tau^2 \sum_{i=1}^m \ln [1 - \tanh^2(\mathfrak{C}_i(x))], \end{aligned}$$

where $\mathfrak{C}(x) = (\mathfrak{C}_1(x), \dots, \mathfrak{C}_m(x))^T$ with $\mathfrak{C}_i(x) \in \mathbb{R}$, $i = 1, \dots, m$. Then, we can rewrite (7.3.5) as

$$V_x^{*T} f(x) + Q(x) + \tau^2 \sum_{i=1}^m \ln [1 - \tanh^2(\mathfrak{C}_i(x))] = 0. \quad (7.3.6)$$

Nevertheless, (7.3.6) is intractable to solve since it is a nonlinear partial differential equation with respect to $V^*(x)$. To overcome the difficulty, we develop an online control scheme to solve (7.3.6) based on ADP approaches. Before proceeding further, we provide the following required assumption.

Assumption 7.3.1 Assume that $L_3(x)$ is a continuously differentiable Lyapunov function candidate for system (7.3.1) and satisfies that

$$\dot{L}_3(x) = L_{3x}^T (f(x) + g(x)u^*) < 0$$

with L_{3x} the partial derivative of $L_3(x)$ with respect to x . Meanwhile, there exists a positive-definite matrix $\Lambda_2(x) \in \mathbb{R}^{n \times n}$ defined on \mathcal{Q} such that

$$L_{3x}^T (f(x) + g(x)u^*) = -L_{3x}^T \Lambda_2(x) L_{3x}. \quad (7.3.7)$$

7.3.1 Solving HJB Equation via Critic Architecture

In this section, we design an online optimal control scheme using a single critic NN. According to [8, 10], the optimal value $V^*(x)$ can be represented by a single-layer NN on a compact set \mathcal{Q} as

$$V^*(x) = W_c^\top \sigma(x) + \varepsilon_{N_2}(x),$$

where $W_c \in \mathbb{R}^{N_2}$ is the ideal NN weight vector, $\sigma(x) = (\sigma_1(x), \dots, \sigma_{N_2}(x))^\top \in \mathbb{R}^{N_2}$ is the activation function with $\sigma_j(x) \in \mathcal{C}^1(\mathcal{Q})$ and $\sigma_j(0) = 0$, the set $\{\sigma_j(x)\}_{j=1}^{N_2}$ is selected to be linearly independent, N_2 is the number of the neurons, and $\varepsilon_{N_2}(x)$ is the NN function reconstruction error. The derivative of $V^*(x)$ with respect to x is obtained as

$$V_x^* = \nabla \sigma^\top(x) W_c + \nabla \varepsilon_{N_2}(x) \quad (7.3.8)$$

with $\nabla \sigma(x) = \partial \sigma(x) / \partial x$, $\nabla \varepsilon_{N_2}(x) = \partial \varepsilon_{N_2}(x) / \partial x$, and $\nabla \sigma(0) = 0$.

Substituting (7.3.8) into (7.3.4), and using the Taylor series expansion, we have

$$u^*(x) = -\tau \tanh\left(\frac{1}{2\tau} g^\top(x) \nabla \sigma^\top W_c\right) + \varepsilon_{u^*}, \quad (7.3.9)$$

where $\varepsilon_{u^*} = -\frac{1}{2}(\underline{1} - \tanh^2(\chi))g^\top(x) \nabla \varepsilon_{N_2}(x)$, $\chi \in \mathbb{R}^m$ is chosen between

$$\frac{1}{2\tau} g^\top(x) \nabla \sigma^\top W_c \text{ and } \frac{1}{2\tau} g^\top(x) (\nabla \sigma^\top W_c + \nabla \varepsilon_{N_2}(x)),$$

and $\underline{1} = (1, \dots, 1)^\top \in \mathbb{R}^m$.

Using (7.3.8), (7.3.6) can be rewritten as

$$W_c^\top \nabla \sigma f(x) + Q(x) + \tau^2 \sum_{i=1}^m \ln [1 - \tanh^2(\mathfrak{B}_{1i}(x))] + \varepsilon_{\text{HJB}} = 0, \quad (7.3.10)$$

where

$$\mathfrak{B}_1(x) = \frac{1}{2\tau} g^\top(x) \nabla \sigma^\top W_c,$$

and $\mathfrak{B}_1(x) = (\mathfrak{B}_{11}(x), \dots, \mathfrak{B}_{1m}(x))^\top$ with $\mathfrak{B}_{1i}(x) \in \mathbb{R}$, $i = 1, \dots, m$, and ε_{HJB} is the HJB approximation error [1].

Remark 7.3.1 It was shown in [1] that there exists a small constant $\varepsilon_h > 0$ and a positive integer N' (depending only on ε_h) such that $N_2 > N'$ implies $\|\varepsilon_{\text{HJB}}\| \leq \varepsilon_h$.

Since the ideal critic NN weight matrix W_c is typically unknown, (7.3.9) cannot be implemented in the real control process. Therefore, we employ a critic NN to

approximate the optimal value function $V^*(x)$ as

$$\hat{V}(x) = \hat{W}_c^\top \sigma(x), \quad (7.3.11)$$

where \hat{W}_c is the estimate of W_c . The weight estimation error for the critic NN is defined as

$$\tilde{W}_c = W_c - \hat{W}_c. \quad (7.3.12)$$

Using (7.3.11), the estimates of (7.3.4) is given by

$$\hat{u}(x) = -\tau \tanh\left(\frac{1}{2\tau} g^\top(x) \nabla \sigma^\top \hat{W}_c\right). \quad (7.3.13)$$

From (7.3.11) and (7.3.13), we derive the approximate Hamiltonian as

$$H(x, \hat{W}_c) = \hat{W}_c^\top \nabla \sigma f(x) + Q(x) + \tau^2 \sum_{i=1}^m \ln [1 - \tanh^2(\mathfrak{B}_{2i}(x))] \triangleq e_2, \quad (7.3.14)$$

where

$$\mathfrak{B}_2(x) = \frac{1}{2\tau} g^\top(x) \nabla \sigma^\top \hat{W}_c,$$

and $\mathfrak{B}_2(x) = (\mathfrak{B}_{21}(x), \dots, \mathfrak{B}_{2m}(x))^\top$ with $\mathfrak{B}_{2i}(x) \in \mathbb{R}$, $i = 1, \dots, m$. Subtracting (7.3.10) from (7.3.14) and using (7.3.12), we obtain

$$e_2 = -\tilde{W}_c^\top \nabla \sigma f(x) + \tau^2 \sum_{i=1}^m [\Gamma(\mathfrak{B}_{2i}(x)) - \Gamma(\mathfrak{B}_{1i}(x))] - \varepsilon_{\text{HJB}}, \quad (7.3.15)$$

where $\Gamma(\mathfrak{B}_{\ell i}(x)) = \ln [1 - \tanh^2(\mathfrak{B}_{\ell i}(x))]$, $\ell = 1, 2$ and $i = 1, \dots, m$. Noticing that for every $\mathfrak{B}_{\ell i}(x) \in \mathbb{R}$, $\Gamma(\mathfrak{B}_{\ell i}(x))$ can be represented accurately as

$$\begin{aligned} \Gamma(\mathfrak{B}_{\ell i}(x)) &= \ln (1 - \tanh^2(\mathfrak{B}_{\ell i}(x))) \\ &= \begin{cases} \ln 4 - 2\mathfrak{B}_{\ell i}(x) - 2 \ln (1 + \exp(-2\mathfrak{B}_{\ell i}(x))), & \mathfrak{B}_{\ell i}(x) > 0, \\ \ln 4 + 2\mathfrak{B}_{\ell i}(x) - 2 \ln (1 + \exp(2\mathfrak{B}_{\ell i}(x))), & \mathfrak{B}_{\ell i}(x) < 0. \end{cases} \end{aligned}$$

That is,

$$\Gamma(\mathfrak{B}_{\ell i}(x)) = \ln 4 - 2\mathfrak{B}_{\ell i}(x)\text{sgn}(\mathfrak{B}_{\ell i}(x)) - 2 \ln [1 + \exp(-2\mathfrak{B}_{\ell i}(x)\text{sgn}(\mathfrak{B}_{\ell i}(x)))],$$

where $\text{sgn}(\mathfrak{B}_{\ell i}(x))$ is the sign function.

Note that

$$\begin{aligned} \sum_{i=1}^m \Gamma(\mathfrak{B}_{\ell i}(x)) &= m \ln 4 - 2 \mathfrak{B}_{\ell}^T(x) \operatorname{sgn}(\mathfrak{B}_{\ell}(x)) \\ &\quad - 2 \sum_{i=1}^m \ln [1 + \exp(-2 \mathfrak{B}_{\ell i}(x) \operatorname{sgn}(\mathfrak{B}_{\ell i}(x)))] . \end{aligned} \quad (7.3.16)$$

Therefore, combining (7.3.15) and (7.3.16), we get

$$\begin{aligned} e_2 &= 2\tau^2 [\mathfrak{B}_1^T(x) \operatorname{sgn}(\mathfrak{B}_1(x)) - \mathfrak{B}_2^T(x) \operatorname{sgn}(\mathfrak{B}_2(x))] \\ &\quad - \tilde{W}_c^T \nabla \sigma f(x) + \tau^2 \Delta_{\mathfrak{B}} - \varepsilon_{\text{HJB}} \\ &= \tau [W_c^T \nabla \sigma g(x) \operatorname{sgn}(\mathfrak{B}_1(x)) - \hat{W}_c^T \nabla \sigma g(x) \operatorname{sgn}(\mathfrak{B}_2(x))] \\ &\quad - \tilde{W}_c^T \nabla \sigma f(x) + \tau^2 \Delta_{\mathfrak{B}} - \varepsilon_{\text{HJB}} \\ &= -\tilde{W}_c^T \nabla \sigma f(x) + \tau \tilde{W}_c^T \nabla \sigma g(x) \operatorname{sgn}(\mathfrak{B}_2(x)) + \mathfrak{D}_1(x), \end{aligned} \quad (7.3.17)$$

where

$$\begin{aligned} \Delta_{\mathfrak{B}} &= 2 \sum_{i=1}^m \ln \frac{1 + \exp[-2 \mathfrak{B}_{1i}(x) \operatorname{sgn}(\mathfrak{B}_{1i}(x))]}{1 + \exp[-2 \mathfrak{B}_{2i}(x) \operatorname{sgn}(\mathfrak{B}_{2i}(x))]}, \\ \mathfrak{D}_1(x) &= \tau W_c^T \nabla \sigma g(x) [\operatorname{sgn}(\mathfrak{B}_1(x)) - \operatorname{sgn}(\mathfrak{B}_2(x))] + \tau^2 \Delta_{\mathfrak{B}} - \varepsilon_{\text{HJB}}. \end{aligned}$$

Remark 7.3.2 From the expression of $\Delta_{\mathfrak{B}}$, we have $\Delta_{\mathfrak{B}} \in [-m \ln 4, m \ln 4]$. Meanwhile, by Remark 7.3.1, we can conclude that $\mathfrak{D}_1(x)$ is a bounded function since W_c is generally bounded and $\tau > 0$ is a constant.

To get the minimum value of e_2 , it is desired to minimize the objective function $E = (1/2)e_2^T e_2$ with the gradient descent algorithm. However, tuning the critic NN weights to minimize E alone does not guarantee the stability of system (7.3.1) during the learning process of NNs if the initial control is not admissible. To ensure the algorithm for online implementation, a novel weight update law for the critic NN is developed by

$$\begin{aligned} \dot{\hat{W}}_c &= -\eta_2 \bar{\phi} \left(Q(x) + \hat{W}_c^T \nabla \sigma f(x) + \tau^2 \sum_{i=1}^m \ln [1 - \tanh^2(\mathfrak{B}_{2i}(x))] \right) \\ &\quad + \frac{\eta_2}{2} \Sigma(x, \hat{u}) \nabla \sigma g(x) [I_m - \mathcal{N}(\mathfrak{B}_2(x))] g^T(x) L_{3x} \\ &\quad + \eta_2 \left(\tau \nabla \sigma g(x) [\tanh(\mathfrak{B}_2(x)) - \operatorname{sgn}(\mathfrak{B}_2(x))] \frac{\varphi^T}{m_s} \hat{W}_c \right. \\ &\quad \left. - (F_2 \hat{W}_c - F_1 \varphi^T \hat{W}_c) \right), \end{aligned} \quad (7.3.18)$$

where $\bar{\phi} = \phi/m_s^2$, $m_s = 1 + \phi^\top \phi$,

$$\phi = \nabla \sigma f(x) - \tau \nabla \sigma g(x) \tanh(\mathfrak{B}_2(x)),$$

$\varphi = \phi/m_s$, $\eta_2 > 0$ is a design parameter,

$$\mathcal{N}(\mathfrak{B}_2(x)) = \text{diag}\{\tanh^2(\mathfrak{B}_{21}(x)), \dots, \tanh^2(\mathfrak{B}_{2m}(x))\},$$

I_m is the $m \times m$ identity matrix, L_{3x} is defined as in Assumption 7.3.1, F_1 and F_2 are tuning parameters with suitable dimensions, and $\Sigma(x, \hat{u})$ is described by

$$\Sigma(x, \hat{u}) = \begin{cases} 0, & \text{if } L_{3x}^\top (f(x) - \tau g(x) \tanh(\mathfrak{B}_2(x))) < 0, \\ 1, & \text{otherwise.} \end{cases} \quad (7.3.19)$$

Remark 7.3.3 The first term in (7.3.18) is used to minimize $E = (1/2)e_2^\top e_2$ and is obtained by utilizing the normalized gradient descent algorithm. The rest are employed to ensure the stability of the optimal closed-loop system while the critic NN learns the optimal weights. $\Sigma(x, \hat{u})$ is designed based on Lyapunov's condition for stability. Observing the expression of $\Sigma(x, \hat{u})$ and noticing that $L_3(x)$ is the Lyapunov function candidate for system (7.3.1) defined in Assumption 7.3.1, if system (7.3.1) is stable (i.e., $L_{3x}^\top (f(x) - \tau g(x) \tanh(\mathfrak{B}_2(x))) < 0$), then $\Sigma(x, \hat{u}) = 0$ and the second term given in (7.3.18) disappears. If system (7.3.1) is unstable, then $\Sigma(x, \hat{u}) = 1$ and the second term given in (7.3.18) is activated. Due to this property of $\Sigma(x, \hat{u})$, it does not require an initial stabilizing control law for system (7.3.1).

Remark 7.3.4 From the expressions of (7.3.14), one shall find that $x = 0$ gives rise to $H(x, \hat{W}_c) = e_2 = 0$. If $F_2 = F_1 \varphi^\top$ in (7.3.18), then $\dot{\hat{W}}_c = 0$. In this sense, the critic NN will no longer be updated. However, the optimal control might not have been obtained at finite time t_0 such that $x(t_0) = 0$. To avoid this circumstance from occurring, a small exploratory signal will be needed, i.e., the PE condition is required. It is worth pointing out that by using the method presented in Sect. 7.2, the PE condition can be removed. However, due to the purpose of this section, we do not employ the approach of Sect. 7.2.

Using ϕ given in (7.3.18), we obtain $\nabla \sigma f(x) = \phi + \tau \nabla \sigma g(x) \tanh(\mathfrak{B}_2(x))$. Therefore, (7.3.17) can be represented as

$$e_2 = -\tilde{W}_c^\top \phi + \tau \tilde{W}_c^\top \nabla \sigma g(x) \mathfrak{N}(x) + \mathfrak{D}_1(x), \quad (7.3.20)$$

where $\mathfrak{N}(x) = \text{sgn}(\mathfrak{B}_2(x)) - \tanh(\mathfrak{B}_2(x))$.

From (7.3.12), (7.3.14), (7.3.18), and (7.3.20), we have

$$\begin{aligned}
\dot{\tilde{W}}_c &= \eta_2 \frac{\varphi}{m_s} \left[-\tilde{W}_c^\top \phi + \tau \tilde{W}_c^\top \nabla \sigma g(x) \mathfrak{N}(x) + \mathfrak{D}_1(x) \right] \\
&\quad - \frac{\eta_2}{2} \Sigma(x, \hat{u}) \nabla \sigma g(x) [I_m - \mathcal{N}(\mathfrak{B}_2(x))] g^\top(x) L_{3x} \\
&\quad + \eta_2 \left[\tau \nabla \sigma g(x) \mathfrak{N}(x) \frac{\varphi^\top}{m_s} \hat{W}_c + (F_2 \hat{W}_c - F_1 \varphi^\top \hat{W}_c) \right]. \tag{7.3.21}
\end{aligned}$$

7.3.2 Stability Analysis of Closed-Loop System with Constrained Inputs

We introduce an assumption before establishing the stability analysis result.

Assumption 7.3.2 The ideal critic NN weight W_c is bounded by a known constant $W_{M_c} > 0$, i.e., $\|W_c\| \leq W_{M_c}$. Meanwhile, the NN approximation error $\varepsilon_{N_2}(x)$ is bounded by a known constant $\varepsilon_M > 0$ over Ω , i.e., $\|\varepsilon_{N_2}(x)\| \leq \varepsilon_M$ for every $x \in \Omega$. In addition, ε_{u^*} in (7.3.9) is upper bounded by a known constant $\varepsilon_{a_2} > 0$, i.e., $\|\varepsilon_{u^*}\| \leq \varepsilon_{a_2}$.

Theorem 7.3.1 Consider the continuous-time nonlinear system described by (7.3.1) with associated HJB equation (7.3.6). Let Assumptions 7.2.2, 7.3.1, and 7.3.2 hold, and let the control input for system (7.3.1) be given by (7.3.13). Moreover, let the critic NN weight tuning law be given by (7.3.18). Then, the function L_{3x} and the weight estimation error \tilde{W}_c are guaranteed to be UUB.

Proof Consider the Lyapunov function candidate

$$L(x) = L_3(x) + \frac{1}{2} \tilde{W}_c^\top \eta_2^{-1} \tilde{W}_c, \tag{7.3.22}$$

where $L_3(x)$ is defined as in Assumption 7.3.1.

Taking the time derivative of (7.3.22) using Assumption 7.3.1 and (7.3.13), we have

$$\dot{L}(x) = L_{3x}^\top (f(x) - \tau g(x) \tanh(\mathfrak{B}_2(x))) + \dot{\tilde{W}}_c^\top \eta_2^{-1} \tilde{W}_c. \tag{7.3.23}$$

Using (7.3.21), we derive the last term in (7.3.23) as

$$\begin{aligned}
\dot{\tilde{W}}_c^\top \eta_2^{-1} \tilde{W}_c &= \left[-\tilde{W}_c^\top \phi + \tau \tilde{W}_c^\top \nabla \sigma g(x) \mathfrak{N}(x) + \mathfrak{D}_1(x) \right] \frac{\varphi^\top}{m_s} \tilde{W}_c \\
&\quad - \frac{1}{2} \Sigma(x, \hat{u}) L_{3x}^\top g(x) [I_m - \mathcal{N}(\mathfrak{B}_2(x))] g^\top(x) \nabla \sigma^\top \tilde{W}_c \\
&\quad + \tau \tilde{W}_c^\top \nabla \sigma g(x) \mathfrak{N}(x) \frac{\varphi^\top}{m_s} \hat{W}_c + \tilde{W}_c^\top (F_2 \hat{W}_c - F_1 \varphi^\top \hat{W}_c) \\
&= -\tilde{W}_c^\top \varphi \varphi^\top \tilde{W}_c + \bar{\mathfrak{D}}_1(x) \varphi^\top \tilde{W}_c + \tilde{W}_c^\top \bar{\mathfrak{D}}_2(x)
\end{aligned}$$

$$-\frac{1}{2}\Sigma(x, \hat{u})L_{3x}^T g(x)[I_m - \mathcal{N}(\mathfrak{B}_2(x))]g^T(x)\nabla\sigma^T\tilde{W}_c \\ + \tilde{W}_c^T(F_2\hat{W}_c - F_1\varphi^T\hat{W}_c), \quad (7.3.24)$$

where $\bar{\mathfrak{D}}_1(x) = \frac{\mathfrak{D}_1(x)}{m_s}$ and $\bar{\mathfrak{D}}_2(x) = \tau \nabla\sigma g(x)\mathfrak{N}(x)\frac{\varphi^T}{m_s}W_c$.

Observe that

$$\tilde{W}_c^T(F_2\hat{W}_c - F_1\varphi^T\hat{W}_c) = \tilde{W}_c^TF_2W_c - \tilde{W}_c^TF_2\tilde{W}_c - \tilde{W}_c^TF_1\varphi^TW_c + \tilde{W}_c^TF_1\varphi^T\tilde{W}_c.$$

Denote $\mathcal{Z}^T = [\tilde{W}_c^T\varphi, \tilde{W}_c^T]$. Then, (7.3.24) can be developed as

$$\dot{\tilde{W}}_c^T\eta_2^{-1}\tilde{W}_c = -\mathcal{Z}^TK\mathcal{Z} + \mathcal{Z}^TG - \frac{1}{2}\Sigma(x, \hat{u})L_{3x}^Tg(x) \\ \times [I_m - \mathcal{N}(\mathfrak{B}_2(x))]g^T(x)\nabla\sigma^T\tilde{W}_c, \quad (7.3.25)$$

where

$$K = \begin{bmatrix} I & -\frac{1}{2}F_1^T \\ -\frac{1}{2}F_1 & F_2 \end{bmatrix}, \quad G = \begin{bmatrix} \bar{\mathfrak{D}}_1(x) \\ \bar{\mathfrak{D}}_2(x) + F_2W_c - F_1\varphi^TW_c \end{bmatrix}.$$

Combining (7.3.23) with (7.3.25) and selecting F_1 and F_2 such that K is positive definite, we obtain

$$\dot{L}(x) \leq L_{3x}^T(f(x) - \tau g(x)\tanh(\mathfrak{B}_2(x))) - \lambda_{\min}(K)\|\mathcal{Z}\|^2 + \vartheta_M\|\mathcal{Z}\| \\ - \frac{1}{2}\Sigma(x, \hat{u})L_{3x}^Tg(x)[I_m - \mathcal{N}(\mathfrak{B}_2(x))]g^T(x)\nabla\sigma^T\tilde{W}_c, \quad (7.3.26)$$

where ϑ_M is the upper bound of $\|G\|$, i.e., $\|G\| \leq \vartheta_M$.

Due to the definition of $\Sigma(x, \hat{u})$ in (7.3.19), (7.3.26) is divided into the following two cases for discussion.

Case 1: $\Sigma(x, \hat{u}) = 0$. By the definition of $\Sigma(x, \hat{u})$ in (7.3.19), the first term in (7.3.26) is negative. Since $\|x\| > 0$ is guaranteed by persistent excitation, one can draw the conclusion that there exists a constant τ such that $0 < \tau \leq \|\dot{x}\|$ implies $L_{3x}^T\dot{x} \leq -\tau\|L_{3x}\|$ based on *dense property* of \mathbb{R} [28]. Then, (7.3.26) can be developed by

$$\dot{L}(x) \leq -\tau\|L_{3x}\| - \lambda_{\min}(K)\left(\|\mathcal{Z}\| - \frac{\vartheta_M}{2\lambda_{\min}(K)}\right)^2 + \frac{\vartheta_M^2}{4\lambda_{\min}(K)}. \quad (7.3.27)$$

Therefore, (7.3.27) implies $\dot{L}(x) < 0$ as long as one of the following conditions holds:

$$\|L_{3x}\| > \frac{\vartheta_M^2}{4\tau\lambda_{\min}(K)}, \quad \text{or} \quad \|\mathcal{Z}\| > \frac{\vartheta_M}{\lambda_{\min}(K)}. \quad (7.3.28)$$

Since

$$\frac{a}{(1+a)^2} \leq \frac{1}{4}, \quad \forall a,$$

from the definition of φ , we have

$$\|\varphi\|^2 = \frac{\phi^\top \phi}{(1 + \phi^\top \phi)^2}$$

and thus $\|\varphi\| \leq \frac{1}{2}$. Noticing that $\|\mathcal{Z}\| \leq \sqrt{1 + \|\varphi\|^2} \|\tilde{W}_c\|$, we obtain

$$\|\mathcal{Z}\| \leq \frac{\sqrt{5}}{2} \|\tilde{W}_c\|.$$

Then, from (7.3.28), we have

$$\|\tilde{W}_c\| > \frac{2\vartheta_M}{\sqrt{5}\lambda_{\min}(K)}.$$

Case 2: $\Sigma(x, \hat{u}) = 1$. In this case, the first term given in (7.3.26) is nonnegative which implies that the control (7.3.13) may not stabilize system (7.3.1). Then, (7.3.26) becomes

$$\begin{aligned} \dot{L}(x) &\leq L_{3x}^\top f(x) - \tau L_{3x}^\top g(x) \left\{ \tanh(\mathfrak{B}_2(x)) \right. \\ &\quad \left. + \frac{1}{2\tau} [I_m - \mathcal{N}(\mathfrak{B}_2(x))] g^\top(x) \nabla \sigma^\top \tilde{W}_c \right\} \\ &\quad - \lambda_{\min}(K) \left(\|\mathcal{Z}\| - \frac{\vartheta_M}{2\lambda_{\min}(K)} \right)^2 + \frac{\vartheta_M^2}{4\lambda_{\min}(K)}. \end{aligned} \quad (7.3.29)$$

Denote $\mathcal{B}(\mathfrak{B}_i(x)) = \tanh(\mathfrak{B}_i(x))$, $i = 1, 2$. By using the Taylor series, we have

$$\begin{aligned} \mathcal{B}(\mathfrak{B}_1(x)) - \mathcal{B}(\mathfrak{B}_2(x)) &= \dot{\mathcal{B}}(\mathfrak{B}_2(x)) (\mathfrak{B}_1(x) - \mathfrak{B}_2(x)) + O((\mathfrak{B}_1(x) - \mathfrak{B}_2(x))^2) \\ &= \frac{1}{2\tau} [I_m - \mathcal{N}(\mathfrak{B}_2(x))] g^\top(x) \nabla \sigma^\top \tilde{W}_c + O((\mathfrak{B}_1(x) - \mathfrak{B}_2(x))^2). \end{aligned} \quad (7.3.30)$$

From (7.3.30), we get

$$\begin{aligned} \tanh(\mathfrak{B}_2(x)) + \frac{1}{2\tau} [I_m - \mathcal{N}(\mathfrak{B}_2(x))] g^\top(x) \nabla \sigma^\top \tilde{W}_c \\ = \tanh(\mathfrak{B}_1(x)) - O((\mathfrak{B}_1(x) - \mathfrak{B}_2(x))^2). \end{aligned} \quad (7.3.31)$$

Substituting (7.3.31) into (7.3.29) and adding and subtracting $L_{1x}^T g(x)\varepsilon_{u^*}$ to the right-hand side of (7.3.29), we derive

$$\begin{aligned}\dot{L}(x) &\leq L_{3x}^T (f(x) + g(x)u^*) - L_{3x}^T g(x)\varepsilon_{u^*} \\ &\quad + \tau L_{3x}^T g(x) O((\mathfrak{B}_1(x) - \mathfrak{B}_2(x))^2) \\ &\quad - \lambda_{\min}(K) \left(\|\mathcal{Z}\| - \frac{\vartheta_M}{2\lambda_{\min}(K)} \right)^2 + \frac{\vartheta_M^2}{4\lambda_{\min}(K)}.\end{aligned}\quad (7.3.32)$$

Using Assumption 7.3.1, we can rewrite (7.3.32) as

$$\begin{aligned}\dot{L}(x) &\leq -\lambda_{\min}(\Lambda_2(x)) \left(\|L_{3x}\| - \frac{\rho_M}{2\lambda_{\min}(\Lambda_2(x))} \right)^2 \\ &\quad - \lambda_{\min}(K) \left(\|\mathcal{Z}\| - \frac{\vartheta_M}{2\lambda_{\min}(K)} \right)^2 + \mu_0,\end{aligned}\quad (7.3.33)$$

where $\rho_M = g_M(\varepsilon_{a_2} + \tau\varepsilon_{b_2})$, ε_{b_2} is the upper bound of $O((\mathfrak{B}_1(x) - \mathfrak{B}_2(x))^2)$, and μ_0 is given as

$$\mu_0 = \frac{\rho_M^2}{4\lambda_{\min}(\Lambda_2(x))} + \frac{\vartheta_M^2}{4\lambda_{\min}(K)}.$$

Hence, (7.3.33) yields $\dot{L}(x) < 0$ as long as one of the following conditions holds:

$$\|L_{3x}\| > \frac{\rho_M}{2\lambda_{\min}(\Lambda_2(x))} + \sqrt{\frac{\mu_0}{\lambda_{\min}(\Lambda_2(x))}},$$

or

$$\|\mathcal{Z}\| > \frac{\vartheta_M}{2\lambda_{\min}(K)} + \sqrt{\frac{\mu_0}{\lambda_{\min}(K)}}.\quad (7.3.34)$$

Observe that $\|\mathcal{Z}\| \leq \frac{\sqrt{5}}{2} \|\tilde{W}_c\|$. Then, from (7.3.34), we have

$$\|\tilde{W}_c\| > \frac{\vartheta_M}{\sqrt{5}\lambda_{\min}(K)} + 2\sqrt{\frac{\mu_0}{5\lambda_{\min}(K)}}.$$

Combining Case 1 and Case 2 and by the standard Lyapunov's extension theorem [12, 13] (or the Lagrange stability result [22]), we conclude that the function L_{3x} and the weight estimation error \tilde{W}_c are UUB. This completes the proof.

Remark 7.3.5 Because $L_3(x)$ given in Assumption 7.3.1 is often constructed by selecting polynomials, one can conclude that L_{3x} is also a polynomial with respect to x . Since Theorem 7.3.1 has verified that L_{3x} is UUB, one can easily obtain that the trajectory of the closed-loop system is also UUB.

7.3.3 *Simulation Study*

Consider the continuous-time affine nonlinear system described by [29]

$$\dot{x}(t) = f(x) + g(x)u, \quad (7.3.35)$$

where

$$f(x) = \begin{bmatrix} -x_1 + x_2 \\ -0.5x_1 - 0.5x_2 + 0.5x_2[\cos(2x_1) + 2]^2 \end{bmatrix},$$

$$g(x) = \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix}.$$

The objective is to control the system with control limits of $|u| \leq 1.2$. The value function is given by

$$V(x(0)) = \int_0^\infty \left(Q(x) + 2\tau \int_0^u \tanh^{-1}(v/\tau) dv \right) dt,$$

where

$$Q(x) = x_1^2 + x_2^2.$$

The gains of the critic NN are chosen as $\eta_2 = 38$, $\tau = 1.2$, $N_2 = 8$. The activation function of the critic NN is chosen as

$$\sigma(x) = [x_1^2, x_2^2, x_1x_2, x_1^4, x_2^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3]^T,$$

and the critic NN weights are denoted as

$$\hat{W}_c = [\hat{W}_{c1}, \dots, \hat{W}_{c8}]^T.$$

The initial weights of the critic NN are chosen as zeros, and the initial system state is selected as $x_0 = [3, -0.5]^T$. It is significant to point out that under this circumstance, the initial control cannot stabilize system (7.3.35). That is, no initial stabilizing control is required for the implementation of this algorithm. To guarantee the PE condition, a small exploratory signal

$$\mathcal{N}(t) = \sin^5(t) \cos(t) + \sin^5(2t) \cos(0.2t)$$

is added to the control law $u(t)$ for the first 10 s.

Using the algorithm developed in Sect. 7.3, we obtain the computer simulation results as in Figs. 7.9, 7.10, 7.11, 7.12 and 7.13. Figure 7.9 shows the trajectories of system state $x(t)$, where the exploratory signal $\mathcal{N}(t)$ is added to the control input during the first 10 s. It is shown that the learning process is finished before the end

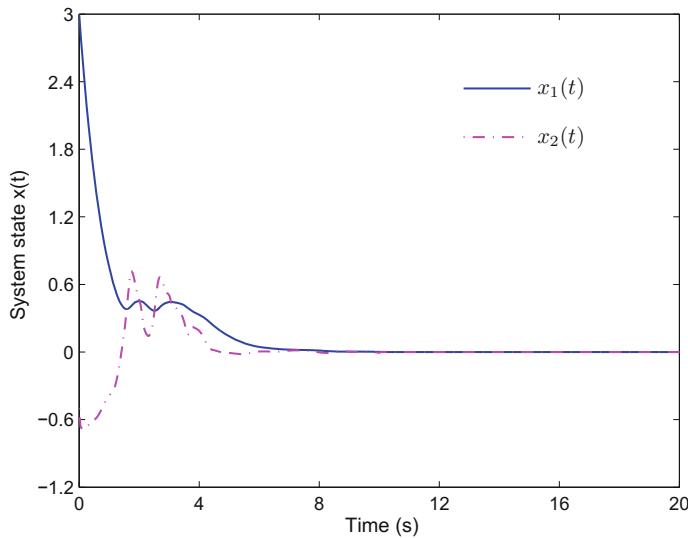


Fig. 7.9 Evolution of system state $x(t)$

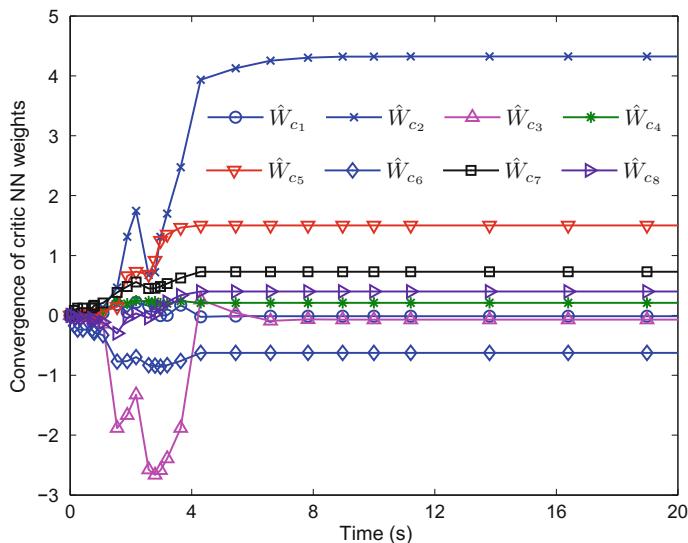


Fig. 7.10 Convergence of critic NN weights

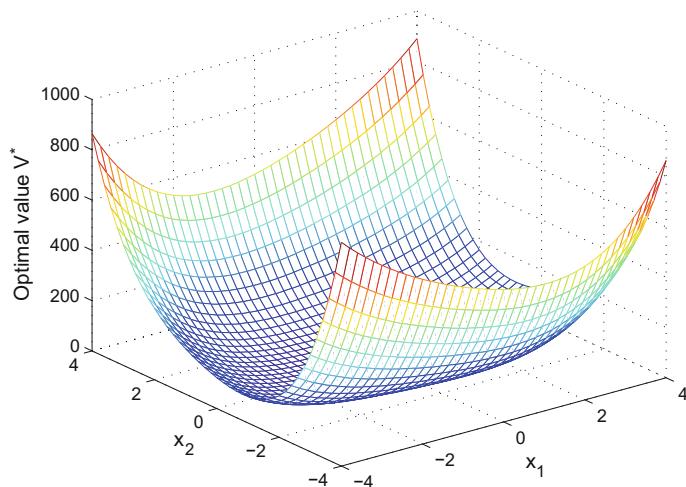


Fig. 7.11 Optimal value function

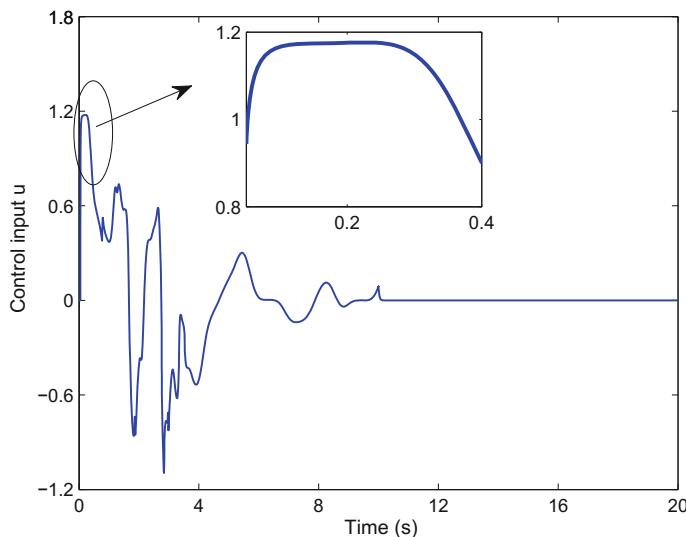


Fig. 7.12 Control input u

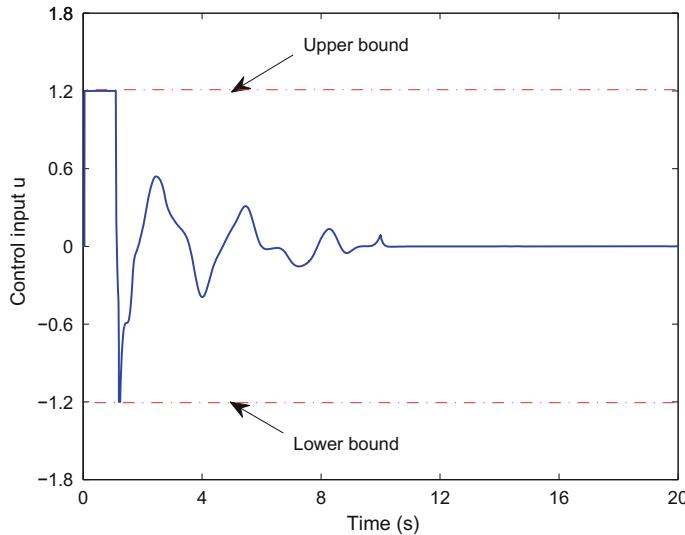


Fig. 7.13 Control input u without considering control constraints

of the exploratory signal. Figure 7.10 shows the convergence of critic NN weights. Figure 7.11 shows the optimal value function. Figure 7.12 illustrates the optimal controller with control constraints. In order to make comparisons, we use Fig. 7.13 to show the controller designed without considering control constraints. The actuator saturation actually exists; therefore, the control input is limited to the bounded value when it overruns the saturation bound. From Figs. 7.9–7.13, it is observed that the optimal control can be obtained using a single NN. Meanwhile, the system state and the estimated weights of the critic NN are all guaranteed to be UUB, while keeping the closed-loop system stable. Moreover, comparing Fig. 7.12 with Fig. 7.13, one can find that the restriction of control constraints has been overcome.

7.4 Conclusions

In this chapter, we developed an identifier–critic architecture based on ADP methodology to derive the approximate optimal control for partially unknown continuous-time nonlinear systems. Meanwhile, an ADP algorithm is developed to obtain the optimal control for constrained continuous-time nonlinear systems. A limitation of the present methods is that the prior knowledge of $g(x)$ is required to be available. In addition, it should be mentioned that the present approaches are applicable to continuous-time input-affine nonlinear systems. Nevertheless, if the system is input-nonaffine, the present methods might not work. Therefore, new ADP methods should be developed to obtain the optimal control of continuous-time nonaffine nonlinear systems.

References

1. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791
2. Beard RW, Sardis GN, Wen JT (1997) Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation. *Automatica* 33(12):2159–2177
3. Bhasin S, Kamalapurkar R, Johnson M, Vamvoudakis KG, Lewis FL, Dixon WE (2013) A novel actor-critic-identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica* 49(1):82–92
4. Bryson AE, Ho YC (1975) Applied optimal control: optimization, estimation and control. CRC Press, Boca Raton
5. Chowdhary G (2010) Concurrent learning for convergence in adaptive control without persistency of excitation. Ph.D. Thesis, Georgia Institute of Technology, USA
6. Chowdhary G, Johnson E (2011) A singular value maximizing data recording algorithm for concurrent learning. In: Proceedings of the American control conference. pp 3547–3552
7. Dierks T, Jagannathan S (2010) Optimal control of affine nonlinear continuous-time systems. In: Proceedings of the American control conference. pp 1568–1573
8. Haykin S (2009) Neural networks and learning machines, 3rd edn. Prentice-Hall, Upper Saddle River
9. Horn RA, Johnson CR (2012) Matrix analysis. Cambridge University Press, New York
10. Hornik K, Stinchcombe M, White H (1990) Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Netw* 3(5):551–560
11. Khalil HK (2001) Nonlinear systems. Prentice-Hall, Upper Saddle River
12. LaSalle JP, Lefschetz S (1967) Stability by Liapunov's direct method with applications. Academic Press, New York
13. Lewis FL, Jagannathan S, Yesildirek A (1999) Neural network control of robot manipulators and nonlinear systems. Taylor & Francis, London
14. Lewis FL, Vrabie D, Syrmos VL (2012) Optimal control. Wiley, Hoboken
15. Lewis FL, Vrabie D, Vamvoudakis KG (2012) Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst Mag* 32(6):76–105
16. Lewis FL, Yesildirek A, Liu K (1996) Multilayer neural-net robot controller with guaranteed tracking performance. *IEEE Trans Neural Netw* 7(2):388–399
17. Li H, Liu D (2012) Optimal control for discrete-time affine non-linear systems using general value iteration. *IET Control Theory Appl* 6(18):2725–2736
18. Liu D, Wang D, Wang FY, Li H, Yang X (2014) Neural-network-based online HJB solution for optimal robust guaranteed cost control of continuous-time uncertain nonlinear systems. *IEEE Trans Cybern* 44(12):2834–2847
19. Liu D, Wang D, Yang X (2013) An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs. *Inf Sci* 220:331–342
20. Liu D, Wei Q (2013) Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems. *IEEE Trans Cybern* 43(2):779–789
21. Lyshevski SE (1998) Optimal control of nonlinear continuous-time systems: design of bounded controllers via generalized nonquadratic functionals. In: Proceedings of the American Control Conference. pp 205–209
22. Michel AN, Hou L, Liu D (2015) Stability of dynamical systems: on the role of monotonic and non-monotonic Lyapunov functions. Birkhäuser, Boston
23. Modares H, Lewis FL (2014) Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica* 50(7):1780–1792
24. Modares H, Lewis F, Naghibi-Sistani MB (2014) Online solution of nonquadratic two-player zero-sum games arising in the H_∞ control of constrained input systems. *Int J Adapt Control Signal Process* 28(3–5):232–254

25. Modares H, Lewis FL, Naghibi-Sistani MB (2014) Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. *Automatica* 50(1):193–202
26. Nodland D, Zargarzadeh H, Jagannathan S (2013) Neural network-based optimal adaptive output feedback control of a helicopter UAV. *IEEE Trans Neural Netw Learn Syst* 24(7):1061–1073
27. Padhi R, Unnikrishnan N, Wang X, Balakrishnan S (2006) A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems. *Neural Netw* 19(10):1648–1660
28. Rudin W (1976) Principles of mathematical analysis. McGraw-Hill, New York
29. Vamvoudakis KG, Lewis FL (2010) Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* 46(5):878–888
30. Wang D, Liu D, Wei Q, Zhao D, Jin N (2012) Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. *Automatica* 48(8):1825–1832
31. Yang X, Liu D, Huang Y (2013) Neural-network-based online optimal control for uncertain nonlinear continuous-time systems with control constraints. *IET Control Theory Appl* 7(17):2037–2047
32. Yang X, Liu D, Wang D (2014) Reinforcement learning for adaptive optimal control of unknown continuous-time nonlinear systems with input constraints. *Int J Control* 87(3):553–566
33. Yang X, Liu D, Wang D, Wei Q (2014) Discrete-time online learning control for a class of unknown nonaffine nonlinear systems using reinforcement learning. *Neural Netw* 55:30–41
34. Yang X, Liu D, Wei Q (2014) Online approximate optimal control for affine non-linear systems with unknown internal dynamics using adaptive dynamic programming. *IET Control Theory Appl* 8(16):1676–1688
35. Yu W (2009) Recent advances in intelligent control systems. Springer, London
36. Zhang H, Cui L, Luo Y (2013) Near-optimal control for nonzero-sum differential games of continuous-time nonlinear systems using single-network ADP. *IEEE Trans Cybern* 43(1):206–216
37. Zhang H, Cui L, Zhang X, Luo Y (2011) Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Trans Neural Netw* 22(12):2226–2236
38. Zhang H, Liu D, Luo Y, Wang D (2013) Adaptive dynamic programming for control: algorithms and stability. Springer, London
39. Zhang H, Qin C, Luo Y (2014) Neural-network-based constrained optimal control scheme for discrete-time switched nonlinear system using dual heuristic programming. *IEEE Trans Autom Sci Eng* 11(3):839–849
40. Zhong X, He H, Zhang H, Wang Z (2014) Optimal control for unknown discrete-time nonlinear markov jump systems using adaptive dynamic programming. *IEEE Trans Neural Netw Learn Syst* 25(12):2141–2155

Chapter 8

Optimal Control of Unknown Continuous-Time Nonaffine Nonlinear Systems

8.1 Introduction

The output of *nonaffine* nonlinear systems depends nonlinearly on the control signal, which is significantly different from *affine* nonlinear systems. Due to this feature, control approaches of affine nonlinear systems do not always hold for nonaffine nonlinear systems. Therefore, it is necessary to develop control methods for nonaffine nonlinear systems. Recently, optimal control problems for nonaffine nonlinear systems have attracted considerable attention. Many remarkable approaches have been proposed to tackle the problem [6, 19, 29, 32].

In this chapter, we present two novel control algorithms based on ADP methods to obtain the optimal control of continuous-time nonaffine nonlinear systems with completely unknown dynamics. First, we construct an ADP-based identifier–actor–critic architecture to achieve the approximate optimal control of continuous-time unknown nonaffine nonlinear systems [28]. The identifier is constructed by a dynamic neural network, which transforms nonaffine nonlinear systems into a kind of affine nonlinear systems. Actor–critic dual networks are employed to derive the optimal control for the newly formulated affine nonlinear systems. Then, we develop a novel ADP-based observer–critic architecture to obtain the approximate optimal control of nonaffine nonlinear systems with unknown dynamics [19]. The present observer is composed of a three-layer feedforward neural network, which aims to get the knowledge of system states. Meanwhile, a single critic neural network is employed for estimating the performance of the systems as well as for constructing the optimal control signal.

8.2 Optimal Control of Unknown Nonaffine Nonlinear Systems with Constrained Inputs

Consider the continuous-time nonlinear systems given by

$$\dot{x}(t) = F(x(t), u(t)), \quad (8.2.1)$$

where $x(t) \in \mathbb{R}^n$ is the state vector available for measurement and $u(t) \in \mathfrak{U} \subset \mathbb{R}^m$ is the control vector, and $\mathfrak{U} = \{u \in \mathbb{R}^m : |u_i| \leq \tau, i = 1, \dots, m\}$ with the saturation bound $\tau > 0$. $F(x, u)$ is an unknown nonaffine nonlinear function. We assume that $F(x, u)$ is Lipschitz continuous on \mathcal{Q} containing the origin, such that the solution $x(t)$ of system (8.2.1) is unique for arbitrarily given initial state $x_0 \in \mathcal{Q}$ and control $u \in \mathfrak{U}$, and $F(0, 0) = 0$.

The value function for system (8.2.1) is defined by

$$V(x(t)) = \int_t^\infty (Q(x(s)) + \mathcal{Y}(u(s))) ds, \quad (8.2.2)$$

where $Q(x)$ is continuously differentiable and positive definite, i.e., $Q(x) \in \mathcal{C}^1(\mathcal{Q})$, $Q(x) > 0$ for all $x \neq 0$ and $x = 0 \Leftrightarrow Q(x) = 0$, and $\mathcal{Y}(u)$ is defined as

$$\mathcal{Y}(u) = 2 \int_0^u (\tau \Psi^{-1}(v/\tau))^T R dv \triangleq 2\tau \sum_{i=1}^m \int_0^{u_i} r_i \psi^{-1}(v_i/\tau) dv_i,$$

where $\Psi^{-1}(v/\tau) = (\psi^{-1}(v_1/\tau), \psi^{-1}(v_2/\tau), \dots, \psi^{-1}(v_m/\tau))^T \in \mathbb{R}^m$, and Ψ^{-T} denotes $(\Psi^{-1})^T$. Meanwhile, $\psi(\cdot)$ is a strictly monotonic odd function satisfying $|\psi(\cdot)| < 1$ and belonging to \mathcal{C}^p ($p \geq 1$) and $\mathcal{L}_2(\mathcal{Q})$. $R = \text{diag}\{r_1, \dots, r_m\}$ with $r_i > 0$, $i = 1, \dots, m$. It is necessary to state that $\mathcal{Y}(u)$ is positive definite since $\psi^{-1}(\cdot)$ is a monotonic odd function and R is positive definite. Without loss of generality, we choose $\psi(\cdot) = \tanh(\cdot)$ and R is assumed to be the $m \times m$ identity matrix, i.e., $R = I_m$.

If the value function $V(x) \in \mathcal{C}^1$, by taking the time derivative of both sides of (8.2.2) and moving the terms on the right-hand side to the left, we have

$$V_x^T F(x, u) + Q(x) + 2\tau \int_0^u \tanh^{-T}(v/\tau) dv = 0, \quad (8.2.3)$$

where $V_x \in \mathbb{R}^n$ denotes the partial derivative of $V(x)$ with respect to x . It should be mentioned that (8.2.3) is a sort of a Lyapunov equation for nonlinear systems [2].

Define the Hamiltonian for the control $u \in \mathcal{A}(\mathcal{Q})$ and the value function $V(x)$ by

$$H(x, V_x, u) = V_x^T F(x, u) + Q(x) + 2\tau \int_0^u \tanh^{-T}(v/\tau) dv. \quad (8.2.4)$$

Then, the optimal cost $V^*(x)$ can be obtained by solving the Hamilton–Jacobi–Bellman (HJB) equation

$$\min_{u(x) \in \mathcal{A}(\Omega)} H(x, V_x^*, u) = 0. \quad (8.2.5)$$

When considering the input-affine form, system (8.2.1) is often given as

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t)) \triangleq F(x(t), u(t)),$$

where $f(x(t)) \in \mathbb{R}^n$ and $g(x(t)) \in \mathbb{R}^{n \times m}$. Then, using (8.2.4) and (8.2.5), the closed-form expression for constrained optimal control is derived as

$$u^*(x) = -\tau \tanh\left(\frac{1}{2\tau} g^\top(x) V_x^*\right). \quad (8.2.6)$$

However, for continuous-time nonaffine nonlinear system (8.2.1), the optimal control $u^*(x)$ cannot be obtained as given in (8.2.6) using (8.2.4) and (8.2.5). The main reason is that

$$\partial H(x, V_x, u) / \partial u = 0$$

might be a high-order nonlinear differential equation with respect to u , which is intractable to solve analytically. In addition, due to the unavailability of the dynamics of system (8.2.1), it brings about more difficulties to obtain the optimal control $u^*(x)$.

Therefore, in order to deal with the optimal control problem through (8.2.4) and (8.2.5), we should first obtain the knowledge of the system dynamics. In what follows, we present a dynamic NN to identify the unknown system dynamics.

8.2.1 Identifier Design via Dynamic Neural Networks

According to [30], system (8.2.1) can be approximated by a dynamic NN as

$$\dot{x}(t) = Ax(t) + W_{m_1}^\top \phi(x(t)) + W_{m_2}^\top \rho(x(t))u(t) + \varepsilon(t), \quad (8.2.7)$$

where $A \in \mathbb{R}^{n \times n}$, $W_{m_1} \in \mathbb{R}^{n \times n}$, and $W_{m_2} \in \mathbb{R}^{n \times n}$ are ideal NN weight matrices, and $\varepsilon(t) \in \mathbb{R}^n$ is a bounded dynamic NN function reconstruction error. The vector function $\phi(x) \in \mathbb{R}^n$ is assumed to be n -dimensional with the elements increasing monotonically. The matrix function $\rho(x) \in \mathbb{R}^{n \times m}$ is assumed to be $\rho(x) = (\rho_1(Y_1^\top x), \dots, \rho_n(Y_n^\top x))^\top$, where $Y_i \in \mathbb{R}^{n \times m}$ is a constant matrix and $\rho_i(\cdot)$ is a nondecreasing function. The typical representations of $\phi(x)$ and $\rho(x)$ are sigmoid functions, such as the hyperbolic tangent function. From the property of sigmoid functions, for every $\zeta_1, \zeta_2 \in \mathbb{R}^n$, there exists $\kappa_i > 0$ ($i = 1, 2$) such that

$$\begin{aligned} \|\phi(\zeta_1) - \phi(\zeta_2)\| &\leq \kappa_1 \|\zeta_1 - \zeta_2\|, \\ \|\rho(\zeta_1) - \rho(\zeta_2)\| &\leq \kappa_2 \|\zeta_1 - \zeta_2\|. \end{aligned} \quad (8.2.8)$$

Remark 8.2.1 Though (8.2.7) shares the same feature as in [30], the matrix A is assumed to be unknown in (8.2.7), rather than a stable matrix as in [30]. Therefore, (8.2.7) can be considered as a more general case to estimate system (8.2.1) using dynamic NNs.

In this chapter, we define the dynamic NN identifier to approximate system (8.2.1) as

$$\dot{\hat{x}}(t) = \hat{A}(t)\hat{x}(t) + \hat{W}_{m_1}^T(t)\phi(\hat{x}(t)) + \hat{W}_{m_2}^T(t)\rho(\hat{x}(t))u(t) + \eta\tilde{x}(t), \quad (8.2.9)$$

where $\hat{x}(t) \in \mathbb{R}^n$ is the dynamic NN state, $\hat{A}(t) \in \mathbb{R}^{n \times n}$, $\hat{W}_{m_1}(t) \in \mathbb{R}^{n \times n}$, and $\hat{W}_{m_2}(t) \in \mathbb{R}^{n \times n}$ are estimated dynamic NN weight matrices, and $\eta\tilde{x}(t)$ is the residual error term with the design parameter $\eta > 0$ and the identification error $\tilde{x}(t) = x(t) - \hat{x}(t)$.

By (8.2.7) and (8.2.9), the identification error dynamics can be developed as

$$\begin{aligned} \dot{\tilde{x}}(t) &= A\tilde{x}(t) + \tilde{A}(t)\hat{x}(t) + W_{m_1}^T\tilde{\phi} + W_{m_2}^T\tilde{\rho}u(t) - \eta\tilde{x}(t) \\ &\quad + \tilde{W}_{m_1}^T(t)\phi(\hat{x}(t)) + \tilde{W}_{m_2}^T(t)\rho(\hat{x}(t))u(t) + \varepsilon(t), \end{aligned} \quad (8.2.10)$$

where $\tilde{A}(t) = A - \hat{A}(t)$, $\tilde{W}_{m_1}(t) = W_{m_1} - \hat{W}_{m_1}(t)$, $\tilde{W}_{m_2}(t) = W_{m_2} - \hat{W}_{m_2}(t)$, $\tilde{\phi} = \phi(x(t)) - \phi(\hat{x}(t))$, and $\tilde{\rho} = \rho(x(t)) - \rho(\hat{x}(t))$.

Prior to showing the stability of the identification error $\tilde{x}(t)$, two assumptions are introduced. It should be mentioned that these assumptions are common techniques, which have been used in [9, 31, 32].

Assumption 8.2.1 The ideal NN weight matrices A , W_{m_1} , and W_{m_2} satisfy

$$AA^T \leq \bar{P}_1, \quad W_{m_1}^T W_{m_1} \leq \bar{P}_2, \quad \text{and} \quad W_{m_2}^T W_{m_2} \leq \bar{P}_3,$$

where \bar{P}_i ($i = 1, 2, 3$) are prior known symmetric positive-definite matrices.

Assumption 8.2.2 The dynamic NN function reconstruction error $\varepsilon(t)$ is upper bounded by the state estimation error $\tilde{x}(t)$, i.e.,

$$\varepsilon^T(t)\varepsilon(t) \leq \delta_M \tilde{x}^T(t)\tilde{x}(t),$$

where δ_M is a positive constant.

Theorem 8.2.1 Let Assumptions 8.2.1 and 8.2.2 be satisfied. If the estimated dynamic NN weight matrices $\hat{A}(t)$, $\hat{W}_{m_1}(t)$, and $\hat{W}_{m_2}(t)$ are updated as

$$\begin{aligned} \dot{\hat{A}}(t) &= \Lambda_1 \hat{x}(t) \tilde{x}^T(t), \\ \dot{\hat{W}}_{m_1}(t) &= \Lambda_2 \phi(\hat{x}(t)) \tilde{x}^T(t), \\ \dot{\hat{W}}_{m_2}(t) &= \Lambda_3 \rho(\hat{x}(t)) u(t) \tilde{x}^T(t), \end{aligned} \quad (8.2.11)$$

where $\Lambda_i \in \mathbb{R}^{n \times n}$ ($i = 1, 2, 3$) are design matrices, then the identifier developed in (8.2.9) can ensure identification of the state, in the sense that the identification error in (8.2.10) goes to zero, i.e., $\lim_{t \rightarrow \infty} \tilde{x}(t) = 0$.

Proof Consider the Lyapunov function candidate

$$L(x) = L_1(x) + L_2(x), \quad (8.2.12)$$

where

$$\begin{aligned} L_1(x) &= \frac{1}{2} \tilde{x}^T(t) \tilde{x}(t), \\ L_2(x) &= \frac{1}{2} \text{tr}(\tilde{A}^T(t) \Lambda_1^{-1} \tilde{A}(t) + \tilde{W}_{m_1}^T(t) \Lambda_2^{-1} \tilde{W}_{m_1}(t) + \tilde{W}_{m_2}^T(t) \Lambda_3^{-1} \tilde{W}_{m_2}(t)). \end{aligned}$$

Taking the time derivative of $L_1(x)$ and using the identification error (8.2.10), we obtain

$$\begin{aligned} \dot{L}_1(x) &= \tilde{x}^T(t) A \tilde{x}(t) + \tilde{x}^T(t) \tilde{A}(t) \hat{x}(t) + \tilde{x}^T(t) W_{m_1}^T \tilde{\phi} \\ &\quad + \tilde{x}^T(t) W_{m_2}^T \tilde{\rho} u(t) - \eta \tilde{x}^T(t) \tilde{x}(t) + \tilde{x}^T(t) \tilde{W}_{m_1}^T(t) \phi(\hat{x}(t)) \\ &\quad + \tilde{x}^T(t) \tilde{W}_{m_2}^T(t) \rho(\hat{x}(t)) u(t) + \tilde{x}^T(t) \varepsilon(t). \end{aligned} \quad (8.2.13)$$

Applying the Cauchy–Schwarz inequality $a^T b \leq (1/2)a^T a + (1/2)b^T b$ (cf. [4]) to $\tilde{x}^T(t) \varepsilon(t)$ and $\tilde{x}^T(t) A \tilde{x}(t)$ in (8.2.13), it follows

$$\begin{aligned} \tilde{x}^T(t) \varepsilon(t) &\leq \frac{1}{2} \tilde{x}^T(t) \tilde{x}(t) + \frac{1}{2} \varepsilon^T(t) \varepsilon(t), \\ \tilde{x}^T(t) A \tilde{x}(t) &\leq \frac{1}{2} \tilde{x}^T(t) A A^T \tilde{x}(t) + \frac{1}{2} \tilde{x}^T(t) \tilde{x}(t). \end{aligned} \quad (8.2.14)$$

Note that from (8.2.1), $u(t)$ is bounded, i.e.,

$$\|u(t)\| \leq \sqrt{\sum_{i=1}^m \tau^2} \triangleq \alpha.$$

Using (8.2.8) and the Cauchy–Schwarz inequality, we get

$$\begin{aligned} \tilde{x}^T(t) W_{m_1}^T \tilde{\phi} &\leq \frac{1}{2} \tilde{x}^T(t) W_{m_1}^T W_{m_1} \tilde{x}(t) + \frac{1}{2} \tilde{\phi}^T \tilde{\phi} \\ &\leq \frac{1}{2} \tilde{x}^T(t) W_{m_1}^T W_{m_1} \tilde{x}(t) + \frac{\kappa_1^2}{2} \tilde{x}^T(t) \tilde{x}(t), \\ \tilde{x}^T(t) W_{m_2}^T \tilde{\rho} u(t) &\leq \frac{1}{2} \tilde{x}^T(t) W_{m_2}^T W_{m_2} \tilde{x}(t) + \frac{1}{2} u^T(t) \tilde{\rho}^T \tilde{\rho} u(t) \\ &\leq \frac{1}{2} \tilde{x}^T(t) W_{m_2}^T W_{m_2} \tilde{x}(t) + \frac{(\alpha \kappa_2)^2}{2} \tilde{x}^T(t) \tilde{x}(t). \end{aligned} \quad (8.2.15)$$

Substituting (8.2.14) and (8.2.15) into (8.2.13), we have

$$\begin{aligned}\dot{L}_1(x) &\leq \frac{1}{2}\tilde{x}^T(t)(AA^T + W_{m_1}^T W_{m_1} + W_{m_2}^T W_{m_2})\tilde{x}(t) \\ &\quad + \frac{1}{2}(2 + \kappa_1^2 + \alpha^2\kappa_2^2 - 2\eta)\tilde{x}^T(t)\tilde{x}(t) \\ &\quad + \tilde{x}^T(t)\tilde{A}(t)\hat{x}(t) + \tilde{x}^T(t)\tilde{W}_{m_1}^T(t)\phi(\hat{x}(t)) \\ &\quad + \tilde{x}^T(t)\tilde{W}_{m_2}^T(t)\rho(\hat{x}(t))u(t) + \frac{1}{2}\varepsilon^T(t)\varepsilon(t).\end{aligned}\quad (8.2.16)$$

On the other hand, taking the time derivative of $L_2(x)$ and using the weight update law (8.2.11), we obtain

$$\begin{aligned}\dot{L}_2(x) &= -\text{tr}(\tilde{A}^T(t)\hat{x}(t)\tilde{x}^T(t) + \tilde{W}_{m_1}^T(t)\phi(\hat{x}(t))\tilde{x}^T(t) \\ &\quad + \tilde{W}_{m_2}^T(t)\rho(\hat{x}(t))u(t)\tilde{x}^T(t)).\end{aligned}\quad (8.2.17)$$

Observe that $\text{tr}(X_1 X_2^T) = \text{tr}(X_2^T X_1) = X_2^T X_1$, $\forall X_1, X_2 \in \mathbb{R}^{n \times 1}$. Then, (8.2.17) can be rewritten as

$$\begin{aligned}\dot{L}_2(x) &= -\tilde{x}^T(t)\tilde{A}^T(t)\hat{x}(t) - \tilde{x}^T(t)\tilde{W}_{m_1}^T(t)\phi(\hat{x}(t)) \\ &\quad - \tilde{x}^T(t)\tilde{W}_{m_2}^T(t)\rho(\hat{x}(t))u(t).\end{aligned}\quad (8.2.18)$$

Combining (8.2.16) and (8.2.18) and employing Assumptions 8.2.1 and 8.2.2, we obtain

$$\begin{aligned}\dot{L}(x) &\leq \frac{1}{2}\tilde{x}^T(t)(AA^T + W_{m_1}^T W_{m_1} + W_{m_2}^T W_{m_2})\tilde{x}(t) \\ &\quad + \frac{1}{2}(\kappa_1^2 + \kappa_2^2\alpha^2 + 2 - 2\eta)\tilde{x}^T(t)\tilde{x}(t) + \frac{1}{2}\varepsilon^T(t)\varepsilon(t) \\ &\leq -\frac{1}{2}\tilde{x}^T(t)\left[(2\eta - \kappa_1^2 - \alpha^2\kappa_2^2 - \delta_M - 2)I_n - \sum_{i=1}^3 \bar{P}_i\right]\tilde{x}(t).\end{aligned}$$

Denote

$$\mathfrak{B} = (2\eta - \kappa_1^2 - \alpha^2\kappa_2^2 - \delta_M - 2)I_n - \sum_{i=1}^3 \bar{P}_i.$$

Selecting η such that \mathfrak{B} is positive definite, we have

$$\dot{L}(x) \leq -\frac{1}{2}\tilde{x}^T(t)\mathfrak{B}\tilde{x}(t) \leq -\frac{1}{2}\lambda_{\min}(\mathfrak{B})\|\tilde{x}(t)\|^2.\quad (8.2.19)$$

Equations (8.2.12) and (8.2.19) guarantee that $\tilde{x}(t)$, $\tilde{A}(t)$, $\tilde{W}_{m_1}(t)$, and $\tilde{W}_{m_2}(t)$ are bounded since $L(x)$ is decreasing. Integrating both sides of (8.2.19), we obtain

$$\int_0^\infty \|\tilde{x}(t)\|^2 dt \leq \frac{2(L(0) - L(\infty))}{\lambda_{\min}(\mathfrak{B})}.$$

By Barbalat's lemma [13], $\lim_{t \rightarrow \infty} \|\tilde{x}(t)\| = 0$, i.e., $\lim_{t \rightarrow \infty} \tilde{x}(t) = 0$. Note that the right-hand side of the above expression is finite since $\lambda_{\min}(\mathfrak{B}) > 0$ and $L(\infty) < L(0) < \infty$ by virtue of (8.2.19). This completes the proof.

Remark 8.2.2 It is worth pointing out that the eigenvalues of a positive-definite matrix are all positive values. Therefore, in order to guarantee (8.2.19) to be negative, η should be selected such that \mathfrak{B} is positive definite. Meanwhile, \bar{P}_1 , \bar{P}_2 , and \bar{P}_3 are selected sufficiently large to guarantee the validity of Assumption 8.2.1, since the ideal NN weights A , W_{m_1} , and W_{m_2} are typically unknown. In this sense, η can only be chosen by trial-and-error.

Remark 8.2.3 By (8.2.11) and $\lim_{t \rightarrow \infty} \tilde{x}(t) = 0$, we derive $\lim_{t \rightarrow \infty} \dot{\hat{A}}(t) = 0$, $\lim_{t \rightarrow \infty} \dot{\hat{W}}_{m_1}(t) = 0$, and $\lim_{t \rightarrow \infty} \dot{\hat{W}}_{m_2}(t) = 0$. Therefore, $\hat{A}(t)$, $\hat{W}_{m_1}(t)$, and $\hat{W}_{m_2}(t)$ will all tend to constants, written as \hat{A} , \hat{W}_{m_1} , and \hat{W}_{m_2} , respectively. Hence, using the identifier (8.2.9), system (8.2.1) can be represented as

$$\dot{x}(t) = \hat{A}x(t) + \hat{W}_{m_1}^\top \phi(x(t)) + \hat{W}_{m_2}^\top \rho(x(t))u(t). \quad (8.2.20)$$

By Remark 8.2.3, and using (8.2.6), we derive the optimal control for the given problem as

$$u^*(x) = -\tau \tanh\left(\frac{1}{2\tau} \rho^\top(x) \hat{W}_{m_2} V_x^*\right). \quad (8.2.21)$$

Then, the HJB equation (8.2.5) becomes

$$\begin{aligned} V_x^{*\top} (\hat{A}x + \hat{W}_{m_1}^\top \phi(x)) - 2\tau^2 \Phi^\top(x) \tanh(\Phi(x)) + Q(x) \\ + 2\tau \int_0^{-\tau \tanh(\Phi(x))} \tanh^{-1}(v/\tau) dv = 0, \end{aligned} \quad (8.2.22)$$

where

$$\Phi(x) = \frac{1}{2\tau} \rho^\top(x) \hat{W}_{m_2} V_x^*.$$

Denote

$$\Phi(x) = (\Phi_1(x), \dots, \Phi_m(x))^\top \in \mathbb{R}^m \text{ with } \Phi_i(x) \in \mathbb{R}, i = 1, \dots, m.$$

According to the integration formulas of inverse hyperbolic tangent function, we note that

$$\begin{aligned}
2\tau \int_0^{-\tau \tanh(\Phi(x))} \tanh^{-T}(\nu/\tau) d\nu \\
= 2\tau \sum_{i=1}^m \int_0^{-\tau \tanh(\Phi_i(x))} \tanh^{-T}(\nu_i/\tau) d\nu_i \\
= 2\tau^2 \Phi^T(x) \tanh(\Phi(x)) + \tau^2 \sum_{i=1}^m \ln [1 - \tanh^2(\Phi_i(x))]. \quad (8.2.23)
\end{aligned}$$

Therefore, (8.2.22) can be rewritten as

$$V_x^{*T}(\hat{A}x + \hat{W}_{m_1}^T \phi(x)) + Q(x) + \tau^2 \sum_{i=1}^m \ln [1 - \tanh^2(\Phi_i(x))] = 0.$$

8.2.2 Actor–Critic Architecture for Solving HJB Equation

According to the universal approximation property of feedforward NNs [11], the value function $V(x)$ given in (8.2.2) can accurately be represented by an NN on a compact set Ω as

$$V(x) = W_c^T \sigma(x) + \varepsilon_{N_0}(x), \quad (8.2.24)$$

where $W_c \in \mathbb{R}^{N_0}$ is the ideal NN weight vector, N_0 is the number of neurons,

$$\sigma(x) = (\sigma_1(x), \sigma_2(x), \dots, \sigma_{N_0}(x))^T \in \mathbb{R}^{N_0}$$

is the activation function with $\sigma_j(x) \in \mathcal{C}^1(\Omega)$ and $\sigma_j(0) = 0$, the set $\{\sigma_j(x)\}_1^{N_0}$ is often selected to be linearly independent, and $\varepsilon_{N_0}(x)$ is a bounded NN function reconstruction error. The derivative of $V(x)$ with respect to x is given by

$$V_x = \nabla \sigma^T(x) W_c + \nabla \varepsilon_{N_0}(x), \quad (8.2.25)$$

where $\nabla \sigma(x) = \partial \sigma(x) / \partial x$, $\nabla \varepsilon_{N_0}(x) = \partial \varepsilon_{N_0}(x) / \partial x$, and $\nabla \sigma(0) = 0$.

Using (8.2.25) and Remark 8.2.3, (8.2.3) becomes

$$\begin{aligned}
W_c^T \nabla \sigma(\hat{A}x + \hat{W}_{m_1}^T \phi(x) + \hat{W}_{m_2}^T \rho(x)u) \\
+ 2\tau \sum_{i=1}^m \int_0^{u_i} \tanh^{-T}(\nu_i/\tau) d\nu_i + Q(x) = \varepsilon_{LE}, \quad (8.2.26)
\end{aligned}$$

where ε_{LE} is the residual error and defined as

$$\varepsilon_{LE} = -\nabla \varepsilon_{N_0}^T(\hat{A}x + \hat{W}_{m_1}^T \phi(x) + \hat{W}_{m_2}^T \rho(x)u).$$

Since the ideal NN weight W_c is typically unknown, (8.2.24) cannot be implemented in real-time control process. Therefore, we employ a critic NN to approximate the value function $V(x)$ as

$$\hat{V}(x) = \hat{W}_c^T \sigma(x), \quad (8.2.27)$$

where \hat{W}_c is the estimation of W_c . The weight estimation error for the critic NN is defined as

$$\tilde{W}_c = W_c - \hat{W}_c.$$

Then, (8.2.26) can be rewritten as

$$\begin{aligned} \hat{W}_c^T \nabla \sigma(\hat{A}x + \hat{W}_{m_1}^T \phi(x) + \hat{W}_{m_2}^T \rho(x)u) \\ + 2\tau \sum_{i=1}^m \int_0^u \tanh^{-1}(v/\tau) dv + Q(x) = \delta_\varepsilon, \end{aligned} \quad (8.2.28)$$

where δ_ε is the Bellman residual error. From (8.2.26) and (8.2.28), we derive

$$\delta_\varepsilon = -\tilde{W}_c^T \nabla \sigma(\hat{A}x + \hat{W}_{m_1}^T \phi(x) + \hat{W}_{m_2}^T \rho(x)u) + \varepsilon_{LE}. \quad (8.2.29)$$

To get the minimum value of δ_ε , it is desired to minimize the objective function

$$E = \frac{1}{2} \delta_\varepsilon^2.$$

Using the gradient descent algorithm, the weight update law for the critic NN is developed as

$$\dot{\hat{W}}_c = -\frac{l_c}{(1 + h^T h)^2} \frac{\partial E}{\partial \hat{W}_c} = -l_c \frac{h}{(1 + h^T h)^2} \delta_\varepsilon, \quad (8.2.30)$$

where

$$h = \nabla \sigma(\hat{A}x + \hat{W}_{m_1}^T \phi(x) + \hat{W}_{m_2}^T \rho(x)u),$$

$l_c > 0$ is the critic NN learning rate, and the term $(1 + h^T h)^2$ is employed for normalization.

On the other hand, by (8.2.21) and (8.2.27), the control policy can be approximated by an action NN as

$$\hat{u}_c(x) = -\tau \tanh\left(\frac{1}{2\tau} \rho^T(x) \hat{W}_{m_2} \nabla \sigma^T \hat{W}_c\right).$$

From the expression $\hat{u}_c(x)$, one can find the action NN shares the same weights \hat{W}_c as the critic NN. However, in standard weight tuning laws [2, 26], the critic NN and the action NN are tuned sequentially, with the weights of the other NN being kept constant. It is generally considered that this type of weight update law is more

time-consuming than tuning the two NN weights simultaneously as in the present case.

To tune simultaneously the weights of the critic NN and the action NN, a separate action NN is employed. In this sense, the control input is redefined as

$$\hat{u}_a(x) = -\tau \tanh\left(\frac{1}{2\tau} \rho^\top(x) \hat{W}_{m_2} \nabla \sigma^\top \hat{W}_a\right), \quad (8.2.31)$$

where $\hat{W}_a \in \mathbb{R}^{N_0}$ denotes the current estimation of $W_a = W_c$. The weight estimation error for the action NN is defined as $\tilde{W}_a = W_c - \hat{W}_a$. The weight update law for the action NN shall be developed in the subsequent section based on the stability analysis.

Remark 8.2.4 Unlike the actor update law derived in [5, 32] by minimizing the Bellman residual error, we develop the actor tuning law based on the stability analysis which shares similar spirits as [21, 25]. The concrete form of the action NN update law is developed next.

8.2.3 Stability Analysis of Closed-Loop System

We start with three assumptions and then establish the stability result.

Assumption 8.2.3 The parameters defined in (8.2.20) are upper bounded such that $\|\hat{A}\| \leq a_1$, $\|\hat{W}_{m_1}\| \leq a_2$, and $\|\hat{W}_{m_2}\| \leq a_3$.

Assumption 8.2.4 There exist known constants $b_\phi > 0$ and $b_\rho > 0$ such that $\|\phi(x)\| \leq b_\phi \|x\|$ and $\|\rho(x)\| \leq b_\rho \|x\|$ for every $x \in \Omega$. Meanwhile, there exist known constants $b_\sigma > 0$ and $b_{\sigma x} > 0$ such that $\|\sigma(x)\| < b_\sigma$ and $\|\nabla \sigma(x)\| < b_{\sigma x}$ for every $x \in \Omega$.

Assumption 8.2.5 The NN reconstruction error $\varepsilon_{N_0}(x)$ and its derivative with respect to x are upper bounded as $\|\varepsilon_{N_0}(x)\| < b_\varepsilon$ and $\|\nabla \varepsilon_{N_0}(x)\| < b_{\varepsilon x}$.

Theorem 8.2.2 Consider the nonlinear system described by (8.2.1) with the structure unknown. Let Assumptions 8.2.3–8.2.5 hold. Take the critic NN and the action NN as in (8.2.27) and (8.2.31), respectively. Let the weight update law for the critic NN be

$$\dot{\hat{W}}_c = -l_c \frac{h}{(1 + h^\top h)^2} \left[h^\top \hat{W}_c + 2\tau \int_0^{\hat{u}_a} \tanh^{-1}(\nu/\tau) d\nu + Q(x) \right], \quad (8.2.32)$$

where $h = \nabla \sigma(\hat{A}x + \hat{W}_{m_1}^\top \phi(x) + \hat{W}_{m_2}^\top \rho(x) \hat{u}_a)$. Define $\bar{h} \triangleq h/(1 + h^\top h)$. Let the action NN be turned by

$$\begin{aligned}\dot{\hat{W}}_a = -l_a & \left[(K_2 \hat{W}_a - K_1 \bar{h}^\top \hat{W}_c) - \tau \nabla \sigma(x) \hat{W}_{m_2}^\top \rho(x) \right. \\ & \left. \times [\tanh(\mathfrak{A}_a) - \text{sgn}(\mathfrak{A}_a)] \frac{\bar{h}^\top}{1 + \bar{h}^\top \bar{h}} \hat{W}_c \right],\end{aligned}\quad (8.2.33)$$

where

$$\mathfrak{A}_a = \frac{1}{2\tau} \rho^\top(x) \hat{W}_{m_2} \nabla \sigma^\top(x) \hat{W}_a,$$

K_1 and K_2 are the tuning vector and the tuning matrix with suitable dimensions which will be detailed in the proof, and $l_a > 0$ is the learning rate of the action NN. Then, the system state $x(t)$, and the actor-critic weight estimation errors \tilde{W}_a , and \tilde{W}_c are all guaranteed to be UUB, when the number of neurons N_0 is selected large enough.

Proof Consider the Lyapunov function candidate

$$L(x) = V(x) + L_1(x) + L_2(x), \quad (8.2.34)$$

where

$$L_1 = \frac{1}{2} \tilde{W}_c^\top l_c^{-1} \tilde{W}_c$$

and

$$L_2 = \frac{1}{2} \tilde{W}_a^\top l_a^{-1} \tilde{W}_a.$$

Taking the time derivative of $V(x)$ based on (8.2.20) and (8.2.24), we have

$$\begin{aligned}\dot{V}(x) &= (W_c^\top \nabla \sigma + \nabla \varepsilon_{N_0}^\top) (\hat{A}x + \hat{W}_{m_1}^\top \phi(x) + \hat{W}_{m_2}^\top \rho(x) \hat{u}_a) \\ &= W_c^\top \nabla \sigma (\hat{A}x + \hat{W}_{m_1}^\top \phi(x)) - \tau W_c^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) \tanh(\mathfrak{A}_a) + \mathcal{E},\end{aligned}\quad (8.2.35)$$

where \mathfrak{A}_a is defined as in (8.2.33), and \mathcal{E} is given as

$$\mathcal{E} = \nabla \varepsilon_{N_0}^\top (\hat{A}x + \hat{W}_{m_1}^\top \phi(x) - \tau \hat{W}_{m_2}^\top \rho(x) \tanh(\mathfrak{A}_a)).$$

Using Assumptions 8.2.3–8.2.5, we obtain

$$\|\mathcal{E}\| \leq b_{ex} (a_1 + a_2 b_\phi + \tau a_3 b_\rho) \|x\|. \quad (8.2.36)$$

On the other hand, from (8.2.21), (8.2.22), and (8.2.25), we define the residual error as

$$\begin{aligned}
& W_c^\top \nabla \sigma(\hat{A}x + \hat{W}_{m_1}^\top \phi(x)) - \tau W_c^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) \tanh(\mathfrak{A}_c) \\
& + 2\tau \int_0^{-\tau \tanh(\mathfrak{A}_c)} \tanh^{-\top}(\nu/\tau) d\nu + Q(x) = \varepsilon_{\text{HJB}},
\end{aligned} \tag{8.2.37}$$

where

$$\mathfrak{A}_c = \frac{1}{2\tau} \rho^\top(x) \hat{W}_{m_2} \nabla \sigma^\top(x) W_c,$$

and ε_{HJB} converges to zero when the number of neurons N_0 is large enough [2, 25]. Assume that ε_{HJB} is upper bounded such that $\|\varepsilon_{\text{HJB}}\| \leq \varepsilon_{\text{max}}$, where $\varepsilon_{\text{max}} > 0$ is a small number.

According to (8.2.23),

$$2\tau \int_0^{-\tau \tanh(\mathfrak{A}_c)} \tanh^{-\top}(\nu/\tau) d\nu$$

is a positive definite, and combining (8.2.35)–(8.2.37), we have

$$\begin{aligned}
\dot{V}(x(t)) &= -Q(x) - 2\tau \int_0^{-\tau \tanh(\mathfrak{A}_c)} \tanh^{-\top}(\nu/\tau) d\nu \\
&+ \tau W_c^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) [\tanh(\mathfrak{A}_c) - \tanh(\mathfrak{A}_a)] + \varepsilon_{\text{HJB}} + \mathcal{E} \\
&\leq -Q(x) + 2\tau \|W_c^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x)\| + \varepsilon_{\text{HJB}} + \mathcal{E} \\
&\leq -Q(x) + \beta \|x\| + \varepsilon_{\text{max}},
\end{aligned} \tag{8.2.38}$$

where $\beta = 2\tau a_3 b_{\sigma x} b_\rho \|W_c\| + b_{\varepsilon x} (a_1 + a_2 b_\phi + \tau a_3 b_\rho)$.

Taking the time derivative of $L_1(t)$, we derive

$$\begin{aligned}
\dot{L}_1(t) &= \tilde{W}_c^\top I_c^{-1} \dot{\tilde{W}}_c = \tilde{W}_c^\top \frac{h}{(1 + h^\top h)^2} \left(\hat{W}_c^\top h + 2\tau \int_0^{\hat{u}_a} \tanh^{-\top}(\nu/\tau) d\nu + Q(x) \right) \\
&= \tilde{W}_c^\top \frac{h}{(1 + h^\top h)^2} \left(\hat{W}_c^\top h + 2\tau \int_0^{-\tau \tanh(\mathfrak{A}_a)} \tanh^{-\top}(\nu/\tau) d\nu \right. \\
&\quad \left. - W_c^\top \xi - 2\tau \int_0^{-\tau \tanh(\mathfrak{A}_c)} \tanh^{-\top}(\nu/\tau) d\nu + \varepsilon_{\text{HJB}} \right) \\
&= \frac{\tilde{W}_c^\top h}{(1 + h^\top h)^2} [\Gamma(\mathfrak{A}_a) - \Gamma(\mathfrak{A}_c) - \tilde{W}_c^\top h + W_c^\top (h - \xi) + \varepsilon_{\text{HJB}}], \tag{8.2.39}
\end{aligned}$$

where h is defined in (8.2.32),

$$\begin{aligned}\xi &= \nabla \sigma \left(\hat{W}_{m_1}^T \phi(x) - \tau \hat{W}_{m_2}^T \rho(x) \tanh(\mathfrak{A}_c) \right), \\ \Gamma(\mathfrak{A}_i) &= 2\tau \int_0^{-\tau \tanh(\mathfrak{A}_i)} \tanh^{-1}(v/\tau) dv, \quad i = 1, 2.\end{aligned}$$

Denote $\mathfrak{A}_i(x) = (\mathfrak{A}_{i1}(x), \dots, \mathfrak{A}_{im}(x))^T$ with $\mathfrak{A}_{ij}(x) \in \mathbb{R}$, $i = 1, 2$; $j = 1, \dots, m$. Then, $\Gamma(\mathfrak{A}_i)$ can be expressed as

$$\begin{aligned}\Gamma(\mathfrak{A}_i) &= 2\tau \sum_{j=1}^m \int_0^{-\tau \tanh(\mathfrak{A}_{ij})} \tanh^{-1}(v_j/\tau) dv_j \\ &= 2\tau^2 \mathfrak{A}_i^T \tanh(\mathfrak{A}_i) + \tau^2 \sum_{j=1}^m \ln \left[1 - \tanh^2(\mathfrak{A}_{ij}) \right].\end{aligned}\quad (8.2.40)$$

For every $\mathfrak{A}_{ij} \in \mathbb{R}$, $\ln(1 - \tanh^2(\mathfrak{A}_{ij}))$ can be expressed as

$$\ln(1 - \tanh^2(\mathfrak{A}_{ij})) = \begin{cases} \ln 4 - 2\mathfrak{A}_{ij} - 2 \ln(1 + \exp(-2\mathfrak{A}_{ij})), & \mathfrak{A}_{ij} > 0; \\ \ln 4 + 2\mathfrak{A}_{ij} - 2 \ln(1 + \exp(2\mathfrak{A}_{ij})), & \mathfrak{A}_{ij} < 0. \end{cases}$$

That is,

$$\ln(1 - \tanh^2(\mathfrak{A}_{ij})) = \ln 4 - 2\mathfrak{A}_{ij} \operatorname{sgn}(\mathfrak{A}_{ij}) - 2 \ln \left[1 + \exp(-2\mathfrak{A}_{ij} \operatorname{sgn}(\mathfrak{A}_{ij})) \right], \quad (8.2.41)$$

where $\operatorname{sgn}(\mathfrak{A}_{ij}) \in \mathbb{R}$ is the sign function with respect to \mathfrak{A}_{ij} .

Combining (8.2.40) and (8.2.41), it follows

$$\begin{aligned}\Gamma(\mathfrak{A}_a) - \Gamma(\mathfrak{A}_c) &= 2\tau^2 [\mathfrak{A}_a^T \tanh(\mathfrak{A}_a) - \mathfrak{A}_c^T \tanh(\mathfrak{A}_c)] \\ &\quad + 2\tau^2 [\mathfrak{A}_c^T \operatorname{sgn}(\mathfrak{A}_c) - \mathfrak{A}_a^T \operatorname{sgn}(\mathfrak{A}_a)] + \Theta_{\mathfrak{A}},\end{aligned}\quad (8.2.42)$$

where

$$\Theta_{\mathfrak{A}} = 2 \sum_{j=1}^m \ln \frac{1 + \exp(-2\mathfrak{A}_{2j}^T \operatorname{sgn}(\mathfrak{A}_{2j}))}{1 + \exp(-2\mathfrak{A}_{1j}^T \operatorname{sgn}(\mathfrak{A}_{1j}))}.$$

From the expression of $\Theta_{\mathfrak{A}}$, one can conclude that

$$\Theta_{\mathfrak{A}} \in [-m \ln 4, m \ln 4].$$

Using \mathfrak{A}_a defined as in (8.2.33) and \mathfrak{A}_c defined as in (8.2.37), (8.2.42) can be written as

$$\begin{aligned}\Gamma(\mathfrak{A}_a) - \Gamma(\mathfrak{A}_c) &= \tau \hat{W}_a^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) \tanh(\mathfrak{A}_a) - \tau W_c^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) \tanh(\mathfrak{A}_c) \\ &\quad - \tau W_c^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) [\operatorname{sgn}(\mathfrak{A}_a) - \operatorname{sgn}(\mathfrak{A}_c)] \\ &\quad + \tau \tilde{W}_a^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) \operatorname{sgn}(\mathfrak{A}_a) + \Theta_{\mathfrak{A}}.\end{aligned}\quad (8.2.43)$$

Therefore, from (8.2.39) and (8.2.43), it follows

$$\begin{aligned}\dot{L}_1(x) &= \tilde{W}_c^\top \frac{h}{(1+h^\top h)^2} \{ \tau \tilde{W}_a^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) [\operatorname{sgn}(\mathfrak{A}_a) - \tanh(\mathfrak{A}_a)] \\ &\quad - \tau W_c^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) [\operatorname{sgn}(\mathfrak{A}_a) - \operatorname{sgn}(\mathfrak{A}_c)] - \tilde{W}_c^\top h + \varepsilon_{\text{HJB}} + \Theta_{\mathfrak{A}} \} \\ &= \tau \tilde{W}_a^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) [\tanh(\mathfrak{A}_a) - \operatorname{sgn}(\mathfrak{A}_a)] \frac{\bar{h}^\top}{1+h^\top h} \hat{W}_c \\ &\quad - \tilde{W}_c^\top \bar{h} \bar{h}^\top \tilde{W}_c + \tilde{W}_c^\top \bar{h} \bar{D}_1(x) + \tilde{W}_a^\top \bar{D}_2(x),\end{aligned}\quad (8.2.44)$$

where

$$\begin{aligned}\bar{D}_1(x) &= \frac{1}{1+h^\top h} \{ \tau W_c^\top \nabla \sigma \hat{W}_{m_2}^\top \rho(x) [\operatorname{sgn}(\mathfrak{A}_c) - \operatorname{sgn}(\mathfrak{A}_a)] - \varepsilon_{\text{HJB}} - \Theta_{\mathfrak{A}} \}, \\ \bar{D}_2(x) &= \tau \nabla \sigma \hat{W}_{m_2}^\top \rho(x) [\operatorname{sgn}(\mathfrak{A}_a) - \tanh(\mathfrak{A}_a)] \frac{\bar{h}^\top}{1+h^\top h} W_c.\end{aligned}$$

Using (8.2.34), (8.2.38), and (8.2.44), we have

$$\begin{aligned}\dot{L}(x) &< -Q(x) - \tilde{W}_c^\top \bar{h} \bar{h}^\top \tilde{W}_c + \tilde{W}_c^\top \bar{h} \bar{D}_1(x) + \tilde{W}_a^\top \bar{D}_2(x) + \beta \|x\| + \varepsilon_{\text{max}} \\ &\quad - \tilde{W}_a^\top \left[l_a^{-1} \dot{\tilde{W}}_a - \tau \nabla \sigma \hat{W}_{m_2}^\top \rho(x) [\tanh(\mathfrak{A}_a) - \operatorname{sgn}(\mathfrak{A}_a)] \frac{\bar{h}^\top}{1+h^\top h} \hat{W}_c \right].\end{aligned}\quad (8.2.45)$$

To guarantee $\dot{L}(x) < 0$, we use the weight update law for the action NN as in (8.2.33). Observe that

$$\begin{aligned}\tilde{W}_a^\top K_2 \hat{W}_a - \tilde{W}_a^\top K_1 \bar{h}^\top \hat{W}_c &= \tilde{W}_a^\top K_2 (W_c - \tilde{W}_a) - \tilde{W}_a^\top K_1 \bar{h}^\top (W_c - \tilde{W}_c) \\ &= \tilde{W}_a^\top K_2 W_c - \tilde{W}_a^\top K_2 \tilde{W}_a - \tilde{W}_a^\top K_1 \bar{h}^\top W_c \\ &\quad + \tilde{W}_a^\top K_1 \bar{h}^\top \tilde{W}_c.\end{aligned}\quad (8.2.46)$$

From (8.2.33), (8.2.45), and (8.2.46), we derive

$$\begin{aligned}\dot{L}(x) &< -Q(x) - \tilde{W}_c^\top \bar{h} \bar{h}^\top \tilde{W}_c - \tilde{W}_a^\top K_2 \tilde{W}_a + \tilde{W}_c^\top \bar{h} \bar{D}_1(x) + \tilde{W}_a^\top K_1 \bar{h}^\top \tilde{W}_c \\ &\quad + \beta \|x\| + \tilde{W}_a^\top (\bar{D}_2(x) + K_2 W_c - K_1 \bar{h}^\top W_c) + \varepsilon_{\text{max}}.\end{aligned}\quad (8.2.47)$$

Since $Q(x) > 0$, there exists a positive value $q \in \mathbb{R}$ such that $x^\top q x < Q(x)$. Let $\mathcal{Z}^\top = [x^\top, \tilde{W}_c^\top \bar{h}, \tilde{W}_a^\top]$. Then, (8.2.47) can be rewritten as

$$\dot{L}(x) < -\mathcal{Z}^T G \mathcal{Z} + \mathcal{Z}^T M + \varepsilon_{\max}, \quad (8.2.48)$$

where

$$G = \begin{bmatrix} qI & 0 & 0 \\ 0 & I & -\frac{K_1^T}{2} \\ 0 & -\frac{K_1}{2} & K_2 \end{bmatrix}, \quad M = \begin{bmatrix} \beta \\ \bar{D}_1(x) \\ \bar{D}_2(x) + K_2 W_c - K_1 \bar{h}^T W_c \end{bmatrix}.$$

Select K_1 and K_2 such that G is positive definite. Then, (8.2.48) implies

$$\dot{L}(x) < -\lambda_{\min}(G) \|\mathcal{Z}\|^2 + \|M\| \|\mathcal{Z}\| + \varepsilon_{\max}.$$

Therefore, $\dot{L}(x)$ is negative as long as the following condition holds:

$$\|\mathcal{Z}\| > \frac{1}{2\lambda_{\min}(G)} \left(\|M\| + \sqrt{\|M\|^2 + 4\lambda_{\min}^2(G)\varepsilon_{\max}} \right).$$

According to the standard Lyapunov's extension theorem [16, 17] (or the Lagrange stability result [20]), this demonstrates the uniform ultimate boundedness of \mathcal{Z} . Therefore, the system state $x(t)$, and actor-critic weight estimates errors $\tilde{W}_a(t)$, and $\tilde{W}_c(t)$ are guaranteed to be UUB. This completes the proof.

8.2.4 Simulation Study

Consider the continuous-time input-nonaffine nonlinear system described by

$$\begin{aligned} \dot{x}_1 &= -0.4 \sin x_1 + 0.6x_2, \\ \dot{x}_2 &= -\sin x_1 \cos x_2 - 0.5u - 0.2 \tan u. \end{aligned} \quad (8.2.49)$$

It is desired to control the system with constrained inputs $|u| \leq 0.45$. The non-quadratic value function is given as

$$V(x) = \int_0^\infty \left(x_1^2 + x_2^2 + 2\tau \int_0^u \tanh^{-T}(v/\tau) dv \right) dt.$$

The prior knowledge of system (8.2.49) is assumed to be unavailable. First, a dynamic NN given in (8.2.9) is employed for identification of the system dynamics. The identifier gains are selected as $\eta = 20$, $\Lambda_1 = [1, 0.5; 0.5, 1]$, $\Lambda_2 = [1, 0.2; 0.2, 1]$, and $\Lambda_3 = [1, 0.1; 0.1, 1]$, and the vector function $\phi(\hat{x})$ and $\rho_i(\zeta_i^T \hat{x})$ are chosen as hyperbolic tangent functions $\tanh(\hat{x})$ and $\tanh(\zeta_i^T \hat{x})$, respectively. Each ζ_i ($i = 1, 2$) is selected randomly within the interval of $[-1, 1]$ and held constant. The computer

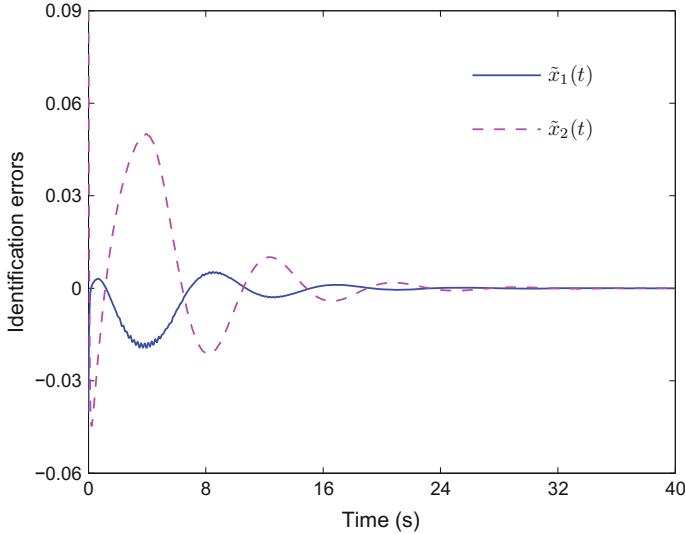


Fig. 8.1 Errors in estimating system state $x(t)$ by the dynamic NN identifier

simulation result of the system identification error is illustrated in Fig. 8.1. From Fig. 8.1, one observes that the dynamic NN identifier can ensure asymptotic identification of the unknown nonaffine nonlinear system (8.2.49). Then, the process of identifying the unknown system dynamics is finished and the identifier NN weights are kept unchanged.

The activation function of the critic NN is chosen with $N_0 = 24$ neurons as

$$\begin{aligned} \sigma(x) = & [x_1^2, x_2^2, x_1 x_2, x_1^4, x_2^4, x_1^3 x_2, x_1^2 x_2^2, x_1 x_2^3, x_1^6, x_2^6, x_1^5 x_2, x_1^4 x_2^2, \\ & x_1^3 x_2^3, x_1^2 x_2^4, x_1 x_2^5, x_1^8 x_2^8, x_1^7 x_2, x_1^6 x_2^2, x_1^5 x_2^3, x_1^4 x_2^4, x_1^3 x_2^5, x_1^2 x_2^6, x_1 x_2^7]^T. \end{aligned} \quad (8.2.50)$$

It should be mentioned that, in this example, the number of neurons is obtained by computer simulations. We find that selecting 24 neurons for the hidden layer can lead to satisfactory simulation results. The weights of the critic NN and the action NN are denoted as $\hat{W}_c = [\hat{W}_{c1}, \dots, \hat{W}_{c24}]^T$ and $\hat{W}_a = [\hat{W}_{a1}, \dots, \hat{W}_{a24}]^T$, respectively. The gains for the actor-critic learning laws are selected as $l_a = 0.7$, $l_c = 0.3$, and τ is given as 0.45. The initial state is chosen to be $x_0 = [0.2, 0.3]^T$, and the initial weights for both the action NN and the critic NN are selected randomly within the interval of $[-1, 1]$. To maintain the persistence of excitation condition, a small exploratory signal $\mathcal{N}(t) = 3 \sin^2(t) \cos(0.5t) - 0.6 \sin^3(0.1t) - 0.4 \sin(2.4t) \cos(2.4t)$ is added to the control $u(t)$ for the first 40 s. The present optimal control algorithm is implemented using (8.2.27) and (8.2.31)–(8.2.33).

Computer simulation results are shown in Figs. 8.2, 8.3, 8.4 and 8.5. Figure 8.2 shows the trajectories of system states $x_1(t)$ and $x_2(t)$, where the exploratory signal

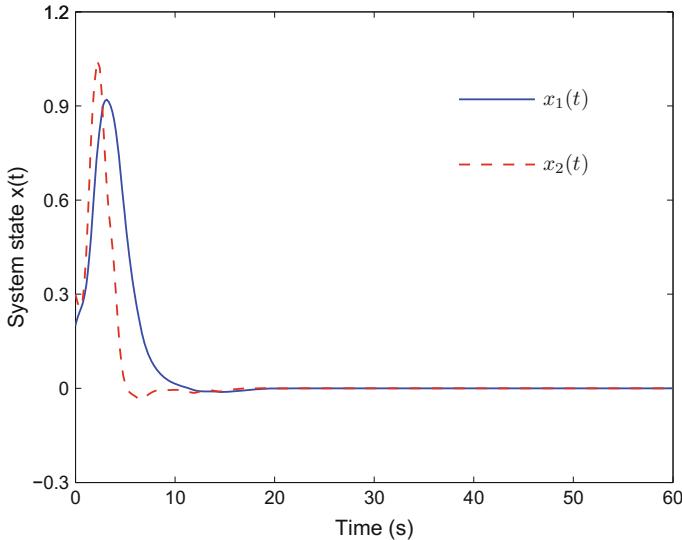


Fig. 8.2 Trajectories of system state $x_i(t)$ ($i = 1, 2$)

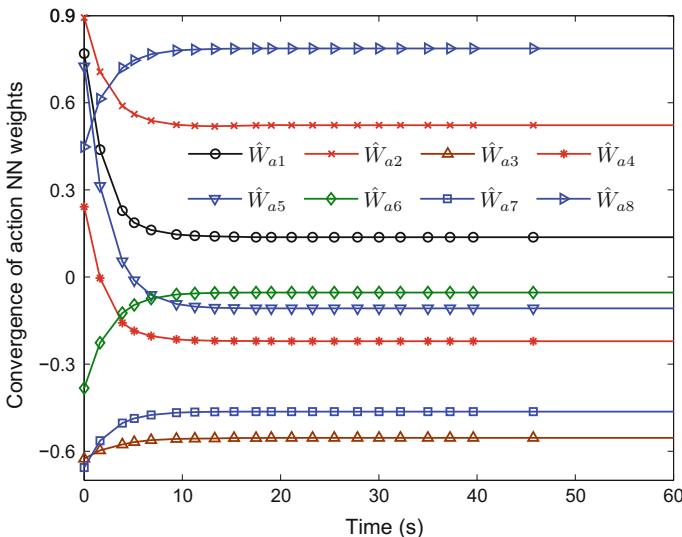


Fig. 8.3 Convergence of the first 8 weights of the action NN

is added to the control input during the first 40 s. Clearly, the learning process is completed before the end of the exploratory signal. Figure 8.3 indicates the convergence curves of the first 8 weights of the action NN. In fact, after 40 s the action NN weight vector converges to $\hat{W}_a = [0.1371, 0.5229, -0.5539, -0.2209, -0.1081, -0.0534,$

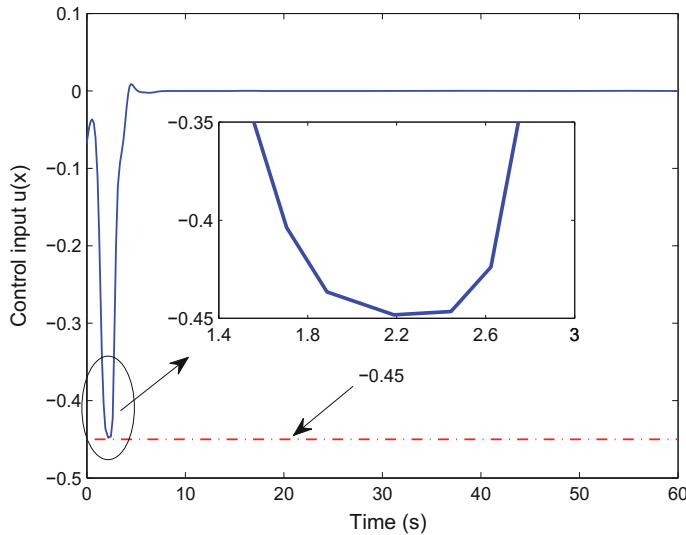


Fig. 8.4 Control input with constraints

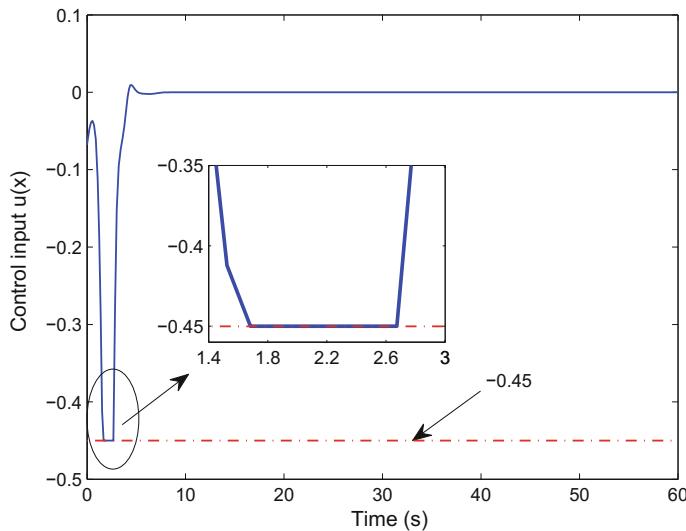


Fig. 8.5 Control input without constraints

$-0.4634, 0.7872, -0.8714, -0.0970, 0.8020, 0.8287, -0.6761, 0.1722, -0.1026, 0.9681, 0.5566, 0.9214, -0.5205, -0.8689, -0.5387, 0.5650, 0.9316, -0.5487]^T$. Meanwhile, after 40 s the critic NN weight vector converges to the same vector as the action NN. Figure 8.4 shows the optimal controller designed using the present algorithm with the consideration of control constraints. To make comparison,

we use Fig. 8.5 to illustrate the controller designed without considering control constraints, where the control signal is saturated for a short period of time.

From Figs. 8.2 and 8.3, it is observed that the system states and the estimated weights of the action NN are guaranteed to be UUB, while keeping closed-loop system stable. Meanwhile, one can find that persistence of excitation ensures the weights converge to their optimal values (\hat{W}_a^*) after 40 s. That is, the final weights \hat{W}_a^* are obtained. Under this circumstance, based on (8.2.31) and (8.2.50), we can derive the optimal control for system (8.2.49). In addition, comparing Fig. 8.4 with Fig. 8.5, we shall find that the restriction of control constraints has been successfully overcome.

8.3 Optimal Output Regulation of Unknown Nonaffine Nonlinear Systems

Consider the continuous-time nonlinear systems described by

$$\dot{x}(t) = F(x(t), u(t)), \quad y(t) = Cx(t), \quad (8.3.1)$$

where $x(t) = [x_1(t), \dots, x_n(t)]^\top \in \mathbb{R}^n$ is the state, $u(t) = [u_1(t), \dots, u_m(t)]^\top \in \mathbb{R}^m$ is the control input, $y(t) = [y_1(t), \dots, y_l(t)]^\top \in \mathbb{R}^l$ is the output, and $F(x, u)$ is an unknown nonaffine nonlinear smooth function with $F(0, 0) = 0$. The state of system (8.3.1) is not available, only the system output $y(t)$ can be measured. For convenience of later analysis, we provide an assumption as follows (for definition of various concepts, see [1, 15]).

Assumption 8.3.1 System (8.3.1) is observable and the system state is bounded in $\mathcal{L}_\infty(\Omega)$. In addition, $C \in \mathbb{R}^{l \times n}$ ($l \leq n$) is a full row rank matrix, i.e., $\text{rank}(C) = l$.

For optimal output regulator problems, the control objective is to find an admissible control for system (8.3.1) which minimizes the infinite-horizon value function

$$V(x(t)) = \int_t^\infty (y^\top(s) Q y(s) + u^\top(s) R u(s)) ds, \quad (8.3.2)$$

where Q and R are positive-definite matrices with appropriate dimensions. Noticing that $y(t) = Cx(t)$, (8.3.2) can be rewritten as

$$V(x(t)) = \int_t^\infty r(x(s), u(s)) ds, \quad (8.3.3)$$

where $r(x, u) = x^\top Q_c x + u^\top R u$ with $Q_c = C^\top Q C$, and Q_c is symmetric positive semidefinite since $l \leq n$.

8.3.1 Neural Network Observer

Due to the fact that the system dynamics and system states are completely unknown, we cannot directly apply existing ADP methods to system (8.3.1). In this section, we employ a multilayer feedforward NN state observer to obtain estimated states of system (8.3.1).

From (8.3.1), we have

$$\dot{x}(t) = Ax + G(x(t), u(t)), \quad y(t) = Cx(t), \quad (8.3.4)$$

where $G(x, u) = F(x, u) - Ax$, A is a Hurwitz matrix, and the pair (C, A) is observable. Then, the state observer for system (8.3.1) is given by

$$\dot{\hat{x}}(t) = A\hat{x}(t) + \hat{G}(\hat{x}(t), u(t)) + K(y(t) - C\hat{x}(t)), \quad \hat{y}(t) = C\hat{x}(t), \quad (8.3.5)$$

where $\hat{x}(t)$ and $\hat{y}(t)$ denote the state and output of the observer, respectively, and the observer gain $K \in \mathbb{R}^{n \times l}$ is selected such that $A - KC$ is a Hurwitz matrix.

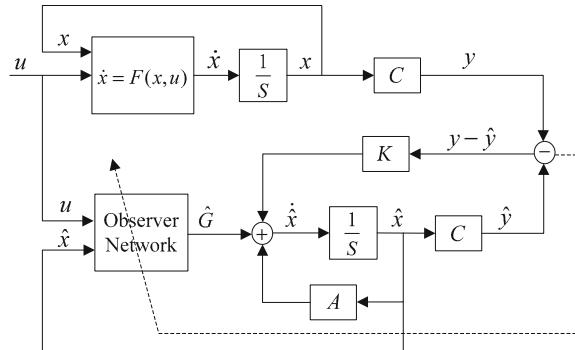
To design an NN state observer, one often uses an NN to identify the nonlinearity and a conventional observer to estimate system states [1, 3, 23]. The structure of the designed NN observer is shown in Fig. 8.6.

It has been proved that a three-layer NN with a single hidden layer can approximate nonlinear systems with any degree of accuracy [11, 12]. According to the universal approximation property of NNs, $G(x, u)$ can be represented on a compact set Ω as

$$G(x, u) = W_o^T \sigma(Y_o^T \bar{x}) + \varepsilon_o(x), \quad (8.3.6)$$

where $W_o \in \mathbb{R}^{k \times n}$ and $Y_o \in \mathbb{R}^{(n+m) \times k}$ are the ideal weight matrices for the hidden layer to the output layer and the input layer to the hidden layer, respectively, k is the number of neurons in the hidden layer, $\bar{x} = [x^T, u^T]^T$ is the NN input, and $\varepsilon_o(x)$ is the bounded NN functional approximation error. It is often assumed that there exists a constant $\varepsilon_M > 0$ such that $\|\varepsilon_o(x)\| \leq \varepsilon_M$. $\sigma(\cdot)$ is the NN activation function

Fig. 8.6 The structural diagram of the NN observer



which is componentwise and selected to be hyperbolic tangent function. Besides, NN activation functions are also bounded such that $\|\sigma(\cdot)\| \leq \sigma_M$ for a constant $\sigma_M > 0$.

For the weights of the three-layer NN, an assumption has been used often as follows [18, 33].

Assumption 8.3.2 The ideal NN weights W_o and Y_o are bounded by known positive constants \bar{W}_M and \bar{Y}_M , respectively. That is,

$$\|W_o\| \leq \bar{W}_M \text{ and } \|Y_o\| \leq \bar{Y}_M.$$

Note that $G(x, u)$ can be approximated by an NN on the compact set Ω as

$$\hat{G}(\hat{x}, u) = \hat{W}_o^T \sigma(\hat{Y}_o^T \hat{x}),$$

where \hat{x} is the estimated state vector, $\hat{x} = [\hat{x}^T, u^T]^T$, \hat{W}_o and \hat{Y}_o are the corresponding estimates of the ideal weight matrices. Then, the NN state observer (8.3.5) can be represented as

$$\dot{\hat{x}} = A\hat{x} + \hat{W}_o^T \sigma(\hat{Y}_o^T \hat{x}) + K(y - C\hat{x}), \quad \hat{y} = C\hat{x}. \quad (8.3.7)$$

Define the state and output estimation errors as $\tilde{x} = x - \hat{x}$ and $\tilde{y} = y - \hat{y}$. Then, using (8.3.4), (8.3.6) and (8.3.7), the error dynamics is obtained as

$$\dot{\tilde{x}} = (A - KC)\tilde{x} + W_o^T \sigma(Y_o^T \tilde{x}) - \hat{W}_o^T \sigma(\hat{Y}_o^T \hat{x}) + \varepsilon_o(x), \quad \tilde{y} = C\tilde{x}. \quad (8.3.8)$$

Adding and subtracting $W_o^T \sigma(\hat{Y}_o^T \hat{x})$ to (8.3.8), it follows

$$\dot{\tilde{x}} = A_o\tilde{x} + \tilde{W}_o^T \sigma(\hat{Y}_o^T \hat{x}) + \zeta(x), \quad \tilde{y} = C\tilde{x}, \quad (8.3.9)$$

where $\tilde{W}_o = W_o - \hat{W}_o$, $A_o = A - KC$, and $\zeta(x) = W_o^T [\sigma(Y_o^T \tilde{x}) - \sigma(\hat{Y}_o^T \hat{x})] + \varepsilon_o(x)$.

Remark 8.3.1 It is worth pointing out that $\zeta(x)$ is a bounded disturbance term. That is, there exists a known constant $\zeta_M > 0$ such that $\|\zeta(x)\| \leq \zeta_M$, because of the boundedness of the hyperbolic tangent function, the NN approximation error $\varepsilon_o(x)$, and the ideal NN weights W_o and Y_o .

To guarantee stability of the NN observer, a suitable tuning algorithm should be provided for the NN weights in the design. Inspired by [1], in what follows we design the weight tuning algorithm based on the error backpropagation algorithm plus some modification terms to guarantee the stability of the state observer and the NN weight estimation errors.

Theorem 8.3.1 Consider system (8.3.1) and the observer dynamics (8.3.7). Let Assumptions 8.3.1 and 8.3.2 hold. If the NN weight tuning algorithms are given as

$$\begin{aligned}\dot{\hat{W}}_o &= -\eta_1 \sigma(\hat{Y}_o^\top \hat{x}) \tilde{y}^\top C A_o^{-1} - \theta_1 \|\tilde{y}\| \hat{W}_o, \\ \dot{\hat{Y}}_o &= -\eta_2 \text{sgn}(\hat{x}) \tilde{y}^\top C A_o^{-1} \hat{W}_o^\top (I_k - \Gamma(\hat{Y}_o^\top \hat{x})) - \theta_2 \|\tilde{y}\| \hat{Y}_o,\end{aligned}\quad (8.3.10)$$

where

$$\Gamma(\hat{Y}_o^\top \hat{x}) = \text{diag}\{\sigma_1^2(\hat{Y}_o^\top \hat{x}), \dots, \sigma_k^2(\hat{Y}_o^\top \hat{x})\},$$

\hat{Y}_{oi} is the i th column of \hat{Y}_o , sgn is the componentwise sign function, $\eta_{\ell_1} > 0$ ($\ell_1 = 1, 2$) are learning rates, and $\theta_{\ell_2} > 0$ ($\ell_2 = 1, 2$) are design parameters, then the state estimation error \tilde{x} converges to the compact set

$$\Omega_{\tilde{x}} = \left\{ \tilde{x} : \|\tilde{x}\| \leq \frac{2d}{\rho \|C\| \lambda_{\min}[(C^+)^\top C^+]} \right\}, \quad (8.3.11)$$

where $d > 0$ is a constant to be determined later (see (8.3.24)), C^+ is the Moore-Penrose pseudoinverse [10] of the matrix C . In addition, the NN weight estimation errors $\tilde{W}_o = W_o - \hat{W}_o$ and $\tilde{Y}_o = Y_o - \hat{Y}_o$ are UUB.

Proof Consider the Lyapunov function candidate

$$L_o(x) = \frac{1}{2} \tilde{x}^\top P \tilde{x} + \frac{1}{2} \text{tr}\{\tilde{W}_o^\top \tilde{W}_o\} + \frac{1}{2} \text{tr}\{\tilde{Y}_o^\top \tilde{Y}_o\}, \quad (8.3.12)$$

where P is a symmetric positive-definite matrix satisfying

$$A_o^\top P + P A_o = -\rho I_n \quad (8.3.13)$$

for the Hurwitz matrix A_o and a positive constant ρ .

Taking the time derivative of L_o , it follows

$$\dot{L}_o(x) = \frac{1}{2} \dot{\tilde{x}}^\top P \tilde{x} + \frac{1}{2} \tilde{x}^\top P \dot{\tilde{x}} + \text{tr}(\tilde{W}_o^\top \dot{\tilde{W}}_o) + \text{tr}(\tilde{Y}_o^\top \dot{\tilde{Y}}_o). \quad (8.3.14)$$

Using (8.3.10), we obtain

$$\begin{aligned}\dot{\tilde{W}}_o &= \eta_1 \sigma(\hat{Y}_o^\top \hat{x}) \tilde{y}^\top C A_o^{-1} + \theta_1 \|\tilde{y}\| \hat{W}_o, \\ \dot{\tilde{Y}}_o &= \eta_2 \text{sgn}(\hat{x}) \tilde{y}^\top C A_o^{-1} \hat{W}_o^\top (I_k - \Gamma(\hat{Y}_o^\top \hat{x})) + \theta_2 \|\tilde{y}\| \hat{Y}_o.\end{aligned}\quad (8.3.15)$$

Substituting (8.3.9), (8.3.13), and (8.3.15) into (8.3.14), we obtain

$$\begin{aligned}\dot{L}_o(x) &= -\frac{\rho}{2} \tilde{x}^\top \tilde{x} + \tilde{x} P (\tilde{W}_o^\top \sigma(\hat{Y}_o^\top \hat{x}) + \zeta) \\ &\quad + \text{tr}\{\tilde{W}_o^\top \sigma(\hat{Y}_o^\top \hat{x}) \tilde{y}^\top l_1 + \tilde{W}_o^\top \theta_1 \|\tilde{y}\| (W_o - \tilde{W}_o)\} \\ &\quad + \text{tr}\{\tilde{Y}_o^\top \text{sgn}(\hat{x}) \tilde{y}^\top l_2 \hat{W}_o^\top (I_k - \Gamma(\hat{Y}_o^\top \hat{x})) + \tilde{Y}_o^\top \theta_2 \|\tilde{y}\| (Y_o - \tilde{Y}_o)\},\end{aligned}\quad (8.3.16)$$

where $l_1 = \eta_1 C A_o^{-1}$ and $l_2 = \eta_2 C A_o^{-1}$.

Replace \tilde{x} using $\tilde{x} = C^+ \tilde{y}$, where C^+ is the Moore–Penrose pseudoinverse of the matrix C [10]. Then, (8.3.16) can be represented as

$$\begin{aligned}\dot{L}_o(x) = & -\frac{\rho}{2} \tilde{y}^T [(C^+)^T C^+] \tilde{y} + C^+ \tilde{y} P \left(\tilde{W}_o^T \sigma \left(\hat{Y}_o^T \hat{x} \right) + \zeta \right) \\ & + \text{tr} \left\{ \tilde{W}_o^T \sigma \left(\hat{Y}_o^T \hat{x} \right) \tilde{y}^T l_1 + \tilde{W}_o^T \theta_1 \|\tilde{y}\| \left(W_o - \tilde{W}_o \right) \right\} \\ & + \text{tr} \left\{ \tilde{Y}_o^T \text{sgn}(\hat{x}) \tilde{y}^T l_2 \hat{W}_o^T (I_k - \Gamma(\hat{Y}_o^T \hat{x})) + \tilde{Y}_o^T \theta_2 \|\tilde{y}\| (Y_o - \tilde{Y}_o) \right\}.\end{aligned}$$

Before proceeding further, we provide the following inequalities:

$$\begin{aligned}\text{tr} \left\{ \tilde{W}_o^T (W_o - \tilde{W}_o) \right\} & \leq \bar{W}_M \|\tilde{W}_o\|_F - \|\tilde{W}_o\|_F^2, \\ \text{tr} \left\{ \tilde{Y}_o^T (Y_o - \tilde{Y}_o) \right\} & \leq \bar{Y}_M \|\tilde{Y}_o\|_F - \|\tilde{Y}_o\|_F^2, \\ \text{tr} \left\{ \tilde{W}_o^T \sigma \left(\hat{Y}_o^T \hat{x} \right) \tilde{y}^T l_1 \right\} & \leq \sigma_M \|l_1\| \|\tilde{y}\| \|\tilde{W}_o\|_F.\end{aligned}\quad (8.3.17)$$

Note that (8.3.17) is true for Frobenius matrix norm, but it is not true for other matrix norms in general. As we have declared earlier, Frobenius norm for matrices and Euclidean norm for vectors are used in this chapter. We do not use the subscript “F” for Frobenius matrix norm for convenience of presentation. The last inequality in (8.3.17) is obtained based on the fact that, for given matrices A and B , the following relationship holds:

$$\text{tr}(AB) = \text{tr}(BA). \quad (8.3.18)$$

Observing that

$$\|\hat{W}_o\| \leq \bar{W}_M + \|\tilde{W}_o\|, \quad 1 - \sigma_M^2 \leq 1,$$

and using (8.3.18), we get

$$\text{tr} \left\{ \tilde{Y}_o^T \text{sgn}(\hat{x}) \tilde{y}^T l_2 \hat{W}_o^T (I_k - \Gamma(\hat{Y}_o^T \hat{x})) \right\} \leq \|l_2\| \|\tilde{y}\| \|\tilde{Y}_o\| (\bar{W}_M + \|\tilde{W}_o\|). \quad (8.3.19)$$

Then, using (8.3.17) and (8.3.19), we have

$$\begin{aligned}\dot{L}_o(x) \leq & -\frac{\rho}{2} \lambda_{\min} [(C^+)^T C^+] \|\tilde{y}\|^2 + \|\tilde{y}\| \|P\| \|C^+\| \left(\sigma_M \|\tilde{W}_o\| + \zeta_M \right) \\ & + \|\tilde{y}\| \sigma_M \|l_1\| \|\tilde{W}_o\| + \|\tilde{y}\| \theta_1 \left(\bar{W}_M \|\tilde{W}_o\| - \|\tilde{W}_o\|^2 \right) \\ & + \|\tilde{y}\| \|l_2\| \|\tilde{Y}_o\| \left(\bar{W}_M + \|\tilde{W}_o\| \right) + \|\tilde{y}\| \theta_2 \left(\bar{Y}_M \|\tilde{Y}_o\| - \|\tilde{Y}_o\|^2 \right).\end{aligned}\quad (8.3.20)$$

Denote $K_1 = \|l_2\|/2$. Adding and subtracting $K_1^2 \|\tilde{W}_o\|^2 \|\tilde{y}\|$ and $\|\tilde{Y}_o\|^2 \|\tilde{y}\|$ to the right-hand side of (8.3.20), it follows

$$\begin{aligned}\dot{L}_o \leq & -\frac{\rho}{2}\lambda_{\min}[(C^+)^T C^+] \|\tilde{y}\|^2 + [\zeta_M \|P\| \|C^+\| - (\theta_1 - K_1^2)] \|\tilde{W}_o\|^2 \\ & + (\sigma_M \|P\| \|C^+\| + \sigma_M \|l_1\| + \theta_1 \bar{W}_M) \|\tilde{W}_o\| + (\theta_2 \bar{Y}_M + \|l_2\| \bar{W}_M) \|\tilde{Y}_o\| \\ & - (\theta_2 - 1) \|\tilde{Y}_o\|^2 - (K_1 \|\tilde{W}_o\| - \|\tilde{Y}_o\|)^2 \] \|\tilde{y}\|. \end{aligned} \quad (8.3.21)$$

Denote K_2 and K_3 as

$$K_2 = \frac{\sigma_M \|P\| \|C^+\| + \sigma_M \|l_1\| + \theta_1 \bar{W}_M}{2(\theta_1 - K_1^2)}, \quad K_3 = \frac{\theta_2 \bar{Y}_M + \|l_2\| \bar{W}_M}{2(\theta_2 - 1)}.$$

To complete the squares for the terms $\|\tilde{W}_o\|$ and $\|\tilde{Y}_o\|$, $K_2^2 \|\tilde{y}\|$ and $K_3^2 \|\tilde{y}\|$ are added to and subtracted from (8.3.21), and we obtain

$$\begin{aligned}\dot{L}_o(x) \leq & -\frac{\rho}{2}\lambda_{\min}[(C^+)^T C^+] \|\tilde{y}\|^2 + [\zeta_M \|P\| \|C^+\| + (\theta_1 - K_1^2) K_2^2 \\ & + (\theta_2 - 1) K_3^2 - (\theta_1 - K_1^2) (K_2 - \|\tilde{W}_o\|)^2 \\ & - (\theta_2 - 1) (K_3 - \|\tilde{Y}_o\|)^2 - (K_1 \|\tilde{W}_o\| - \|\tilde{Y}_o\|)^2] \|\tilde{y}\|. \end{aligned} \quad (8.3.22)$$

Select $\theta_1 \geq K_1^2$ and $\theta_2 \geq 1$. Then, (8.3.22) becomes

$$\dot{L}_o \leq -\frac{\rho}{2}\lambda_{\min}[(C^+)^T C^+] \|\tilde{y}\|^2 + d \|\tilde{y}\|, \quad (8.3.23)$$

where

$$d = \zeta_M \|P\| \|C^+\| + (\theta_1 - K_1^2) K_2^2 + (\theta_2 - 1) K_3^2. \quad (8.3.24)$$

Therefore, for guaranteeing $\dot{L}_o < 0$, the following condition should hold, i.e.,

$$\|\tilde{y}\| > \frac{2d}{\rho \lambda_{\min}[(C^+)^T C^+]} \quad (8.3.25)$$

Note that $\|\tilde{y}\| \leq \|C\| \|\tilde{x}\|$. Then, (8.3.25) implies

$$\|\tilde{x}\| > \frac{2d}{\rho \|C\| \lambda_{\min}[(C^+)^T C^+]}.$$

That is, the state estimation error \tilde{x} converges to $\mathcal{Q}_{\tilde{x}}$ defined as in (8.3.11). Meanwhile, by using the standard Lyapunov's extension theorem [16, 17] (or the Lagrange stability result [20]), we conclude that the weight estimation errors \tilde{W}_o and \tilde{Y}_o are UUB. This completes the proof.

Remark 8.3.2 By linear matrix theory [7, 10], we can obtain that $\text{rank}(C) = \text{rank}(C^+)$ and $\text{rank}(C^+) = \text{rank}[(C^+)^T C^+]$. Accordingly, using Assumption 8.3.1,

we derive $\text{rank}[(C^+)^T C^+] = \text{rank}(C) = l$. Noticing that $(C^+)^T C^+ \in \mathbb{R}^{l \times l}$ and $(C^+)^T C^+$ is a symmetric matrix, we can conclude that $(C^+)^T C^+$ is positive definite. Therefore, $\lambda_{\min}[(C^+)^T C^+] > 0$. This shows that the compact set $\mathcal{Q}_{\hat{x}}$ makes sense.

Remark 8.3.3 The explanation about selecting an NN observer rather than system identification technique is given here. In control engineering, a common approach is to start from the measurement of system behavior and external influences (inputs to the system) and try to determine a mathematical relationship between them without going into the details of what is actually happening inside the system [8, 27]. This approach is called system identification. So, we can conclude that based on system identification, we are generally able to obtain a “black box” model of the nonlinear system [14], but do not get any in depth knowledge about system states because they are the internal properties of the system. In most real cases, the state variables are unavailable for direct online measurements, and merely input and output of the system are measurable. Therefore, estimating the state variables by observers plays an important role in the control of processes to achieve better performances. Once the estimated states are obtained, we can directly design a state feedback controller to achieve the optimization of system performance [24]. In conclusion, here we employ an NN observer rather than system identification techniques.

8.3.2 *Observer-Based Optimal Control Scheme Using Critic Network*

By (8.3.1) and (8.3.3) and using the observed state \hat{x} to replace the system state x , we obtain the Hamiltonian as

$$H(\hat{x}, u, V_{\hat{x}}) = r(\hat{x}(t), u(t)) + V_{\hat{x}}^T F(\hat{x}(t), u(t)),$$

where

$$r(\hat{x}, u) = \hat{x}^T Q_c \hat{x} + u^T R u$$

as in (8.3.3) and $V_{\hat{x}} = \frac{\partial V(\hat{x})}{\partial \hat{x}}$. The optimal value function $V^*(\hat{x}(t))$ is

$$V^*(\hat{x}(t)) = \min_{u \in \mathcal{A}(\Omega)} \int_t^\infty r(\hat{x}(\tau), u(\tau)) d\tau,$$

and satisfies the HJB equation

$$\min_{u \in \mathcal{A}(\Omega)} H(\hat{x}, u, V_{\hat{x}}^*) = 0, \quad (8.3.26)$$

where $V_{\hat{x}}^* = \frac{\partial V^*(\hat{x})}{\partial \hat{x}}$. Assume that the minimum on the right-hand side of (8.3.26) exists and is unique. Then, by solving the equation $\frac{\partial H(\hat{x}, u, V_{\hat{x}}^*)}{\partial u} = 0$, the optimal control can be obtained as

$$u^* = -\frac{1}{2} R^{-1} \left(\frac{\partial F(\hat{x}, u)}{\partial u} \right)^T V_{\hat{x}}^*. \quad (8.3.27)$$

Substituting (8.3.27) into (8.3.26), it follows

$$\begin{aligned} 0 = & \hat{x}^T Q_c \hat{x} + \frac{1}{4} V_{\hat{x}}^{*\top} \frac{\partial F(\hat{x}, u)}{\partial u} R^{-1} \left(\frac{\partial F(\hat{x}, u)}{\partial u} \right)^T V_{\hat{x}}^* \\ & + V_{\hat{x}}^{*\top} F \left(\hat{x}, -\frac{1}{2} R^{-1} \left(\frac{\partial F(\hat{x}, u)}{\partial u} \right)^T V_{\hat{x}}^* \right). \end{aligned} \quad (8.3.28)$$

To obtain solutions of the optimal control problem, we only need to solve (8.3.28). However, due to the nonlinear nature of the HJB equation, finding its solutions is generally difficult or impossible. Therefore, a scheme shall be developed using NNs for solving the above optimal control problems. The structural diagram of the NN observer-based controller is shown in Fig. 8.7.

According to the universal approximation property of NNs, the value function $V^*(\hat{x})$ can be represented on a compact set Ω as

$$V^*(\hat{x}) = W_c^T \sigma(Y_c^T \hat{x}) + \varepsilon_c(\hat{x}), \quad (8.3.29)$$

where $W_c \in \mathbb{R}^{k_c}$ and $Y_c \in \mathbb{R}^{n \times k_c}$ are the ideal weight matrices for the hidden layer to the output layer and the input layer to the hidden layer, respectively, k_c is the number of neurons in the hidden layer, and ε_c is the bounded NN functional approximation error. In our design, based on [12], for both simplicity of learning and efficiency of approximation, the output layer weight W_c is adapted online, whereas the input layer weight Y_c is selected initially at random and held fixed during the entire learning process. It is demonstrated in [11, 12] that if the number of neurons k_c is sufficiently large, then the NN approximation error ε_c can be kept arbitrarily small.

The output of the critic NN is given as

$$\hat{V}(\hat{x}) = \hat{W}_c^T \sigma(Y_c^T \hat{x}) = \hat{W}_c^T \sigma(z),$$

where \hat{W}_c is the estimate of W_c . Since the hidden layer weight matrix Y_c is fixed, we write the activation function $\sigma(Y_c^T \hat{x})$ as $\sigma(z)$ with $z = Y_c^T \hat{x}$.

The derivative of the value function $V(\hat{x})$ with respect to \hat{x} is

$$V_{\hat{x}} = \nabla \sigma_c^T W_c + \nabla \varepsilon_c, \quad (8.3.30)$$

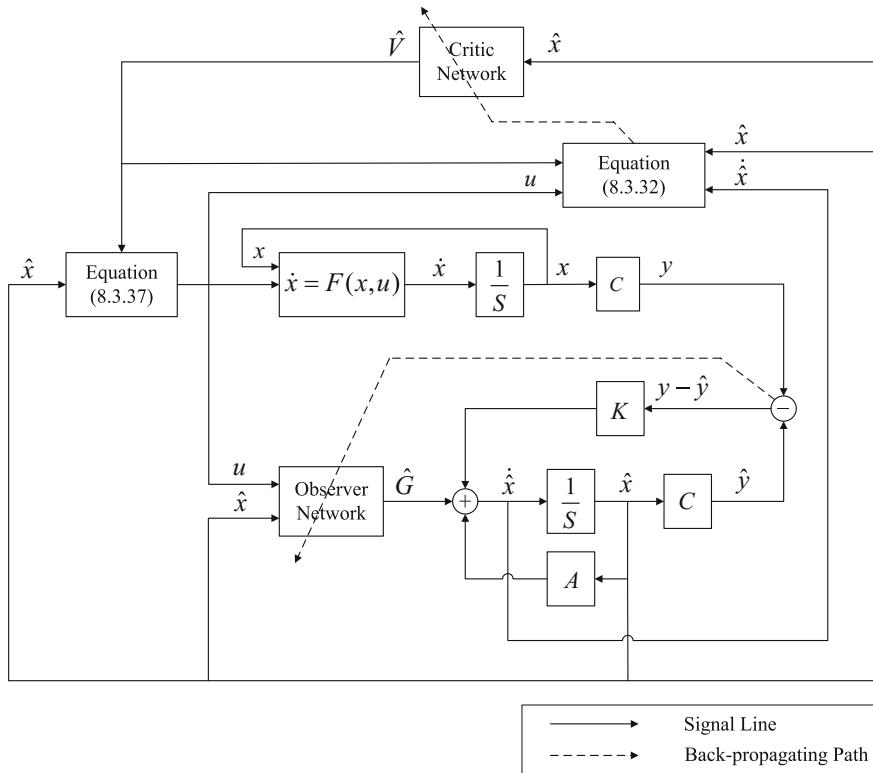


Fig. 8.7 The structural diagram of the NN observer-based controller

where $\nabla \sigma_c^\top = Y_c(\partial \sigma_c^\top(z)/\partial z)$ and $\nabla \varepsilon_c = \partial \varepsilon_c / \partial \hat{x}$. In addition, the derivative of $\hat{V}(\hat{x})$ with respect to \hat{x} is obtained as $\hat{V}_{\hat{x}} = \nabla \sigma_c^\top \hat{W}_c$. Then, the approximate Hamiltonian is derived as

$$H(\hat{x}, u, \hat{W}_c) = \hat{W}_c^\top \nabla \sigma_c F(\hat{x}, u) + r(\hat{x}, u) = e_c. \quad (8.3.31)$$

It is worth pointing out that, to get the error e_c , the knowledge of the system dynamics is required. To overcome this limitation, the NN observer developed in (8.3.7) is used to replace $F(\hat{x}, u)$. Then, (8.3.31) becomes

$$e_c = \hat{W}_c^\top \nabla \sigma_c \dot{\hat{x}} + r(\hat{x}, u). \quad (8.3.32)$$

Given that $u \in \mathcal{A}(\Omega)$, it is desired to select \hat{W}_c to minimize the squared residual error $E_c(\hat{W}_c)$ as

$$E_c(\hat{W}_c) = \frac{1}{2} e_c^2.$$

Using the normalized gradient descent algorithm, the weight update law for the critic NN is derived as

$$\dot{\hat{W}}_c = -\frac{\alpha}{(\psi^\top \psi + 1)^2} \frac{\partial E_c}{\partial \hat{W}_c} = -\alpha \frac{\psi}{(\psi^\top \psi + 1)^2} (\psi^\top \hat{W}_c + r(\hat{x}, u)), \quad (8.3.33)$$

where $\alpha > 0$ is the learning rate, and

$$\psi = \nabla \sigma_c \dot{\hat{x}}.$$

This is a modified Levenberg–Marquardt algorithm, and $(\psi^\top \psi + 1)^2$ is used for normalization. Define the weight estimation error of critic NN as $\tilde{W}_c = W_c - \hat{W}_c$.

By (8.3.26) and (8.3.29) and replacing $F(x, u)$ by $\dot{\hat{x}}$, we get

$$0 = r(\hat{x}, u) + W_c^\top \nabla \sigma_c \dot{\hat{x}} - \vartheta, \quad (8.3.34)$$

where $\vartheta = -\nabla \sigma_c \dot{\hat{x}}$ is the residual error due to the NN approximation. Substituting (8.3.34) into (8.3.33), and using the notation

$$\psi_1 = \frac{\psi}{(\psi^\top \psi + 1)}, \quad \psi_2 = \psi^\top \psi + 1,$$

the critic NN weight estimation error dynamics becomes

$$\dot{\tilde{W}}_c = -\alpha \psi_1 \psi_1^\top \tilde{W}_c + \alpha \psi_1 \frac{\vartheta}{\psi_2}.$$

Considering (8.3.34), we can further derive

$$\begin{aligned} \dot{\tilde{W}}_c &= -\dot{\tilde{W}}_c \\ &= \alpha \psi_1 \psi_1^\top \tilde{W}_c - \alpha \psi_1 \frac{\vartheta}{\psi_2} \\ &= -\alpha \frac{\psi_1}{\psi_2} (\psi^\top (\hat{W}_c - W_c) + \vartheta) \\ &= -\alpha \frac{\psi_1}{\psi^\top \psi + 1} (\psi^\top \hat{W}_c + r(\hat{x}, \hat{u})). \end{aligned} \quad (8.3.35)$$

From the expression ψ_1 , there exists a constant $\psi_{1M} > 1$ such that $\|\psi_1\| < \psi_{1M}$. It is important to note that the persistence of excitation (PE) condition is required for tuning the critic NN. To satisfy the PE condition, a small exploratory signal is added to the control input [25]. Furthermore, the PE condition ensures $\|\psi_1\| \geq \psi_{1m}$, with ψ_{1m} being a positive constant.

Using (8.3.27) and (8.3.30), the control u is given as

$$u = -\frac{1}{2} R^{-1} \left(\frac{\partial F(x, u)}{\partial u} \right)^\top (\nabla \sigma_c^\top W_c + \nabla \varepsilon_c). \quad (8.3.36)$$

The approximate expression of u is obtained as

$$\hat{u} = -\frac{1}{2} R^{-1} \left(\frac{\partial \hat{F}(\hat{x}, u)}{\partial u} \right)^\top \nabla \sigma_c^\top \hat{W}_c. \quad (8.3.37)$$

It should be mentioned that, to compute u in (8.3.36), the knowledge of $\partial F(x, u)/\partial u$ is required. However, for unknown nonlinear systems, this term cannot be obtained directly. In this chapter, using the observer NN, its estimates can be derived as

$$\frac{\partial \hat{F}(\hat{x}, u)}{\partial u} = \frac{\partial \hat{G}(\hat{x}, u)}{\partial u} = \frac{\partial (\hat{W}_o^\top \sigma(\hat{Y}_o^\top \hat{x}))}{\partial u} = \hat{W}_o^\top \frac{\partial \sigma(\hat{Y}_o^\top \hat{x})}{\partial (\hat{Y}_o^\top \hat{x})} \hat{Y}_o^\top \frac{\partial \hat{x}}{\partial u}.$$

Therefore, $\partial \hat{F}(\hat{x}, u)/\partial u$ can be obtained by the backpropagation from the output of the observer NN to its input u .

8.3.3 Stability Analysis of Closed-Loop System

Before proceeding further, we provide two assumptions as follows.

Assumption 8.3.3 The NN approximation error and its gradient are bounded on a compact set containing Ω . That is, $\|\varepsilon_c\| < \varepsilon_{cM}$ and $\|\nabla \varepsilon_c\| < \varepsilon_{dM}$, where ε_{cM} and ε_{dM} are known positive constants.

Assumption 8.3.4 The NN activation function and its gradient are bounded. That is, $\|\sigma\| < \sigma_{cM}$ and $\|\nabla \sigma_c\| < \sigma_{dM}$, where σ_{cM} and σ_{dM} are known positive constants.

Theorem 8.3.2 Consider the NN observer (8.3.7) and the feedback control given in (8.3.37). Let weight tuning laws for the observer NN be provided as in (8.3.10) and for the critic NN be provided as in (8.3.35). Then, x , \tilde{x} , \tilde{W}_o , \tilde{Y}_o and \tilde{W}_c in the NN observer-based control system are all UUB.

Proof Consider the Lyapunov function candidate

$$L(x) = L_o(x) + \underbrace{\frac{1}{2\alpha} \tilde{W}_c^\top \tilde{W}_c}_{L_{c1}} + \underbrace{\alpha(x^\top x + \gamma V(x))}_{L_{c2}},$$

where L_o is defined as in (8.3.12) and $\gamma > 0$.

Note the facts that

$$\psi_2 = \psi^T \psi + 1 \geq 1$$

and the residual error is upper bounded, i.e., there exists a constant $\vartheta_M > 0$ such that $\vartheta \leq \vartheta_M$. Taking the time derivative of L_{c1} , we obtain

$$\begin{aligned}\dot{L}_{c1} &= \frac{1}{\alpha} \tilde{W}_c^T \dot{\tilde{W}}_c \\ &= \frac{1}{\alpha} \tilde{W}_c^T \left(-\alpha \psi_1 \psi_1^T \tilde{W}_c + \alpha \psi_1 \frac{\vartheta}{\psi_2} \right) \\ &= -\|\tilde{W}_c^T \psi_1\|^2 + \tilde{W}_c^T \psi_1 \frac{\vartheta}{\psi_2} \\ &\leq -\|\tilde{W}_c^T \psi_1\|^2 + \|\tilde{W}_c^T \psi_1\| \left\| \frac{\vartheta}{\psi_2} \right\| \\ &\leq -\|\tilde{W}_c^T \psi_1\|^2 + \|\tilde{W}_c^T \psi_1\| \vartheta_M \\ &= -\left(\|\tilde{W}_c^T \psi_1\| - \frac{\vartheta_M}{2} \right)^2 + \frac{\vartheta_M^2}{4}.\end{aligned}$$

Using Assumptions 8.3.3 and 8.3.4, and

$$(a + b + c)^T (a + b + c) \leq 3(a^T a + b^T b + c^T c), \quad \forall a, b, c \in \mathbb{R}^n$$

we have

$$\begin{aligned}\dot{L}_{c2} &= 2\alpha x^T \dot{x} + \alpha \gamma (-x^T Q_c x - \hat{u}^T R \hat{u}) \\ &= 2\alpha x^T (Ax + W_o^T \sigma(Y_o^T \bar{x}) + \varepsilon) + \alpha \gamma (-x^T Q_c x - \hat{u}^T R \hat{u}) \\ &\leq \alpha [x^T x + (Ax + W_o^T \sigma(Y_o^T \bar{x}) + \varepsilon)^T (Ax + W_o^T \sigma(Y_o^T \bar{x}) + \varepsilon) \\ &\quad + \gamma (-x^T Q_c x - \hat{u}^T R \hat{u})] \\ &\leq \alpha [(1 + 3\|A\|^2) \|x\|^2 + 3\|W_o^T \sigma(Y_o^T \bar{x})\|^2 + 3\|\varepsilon\|^2 \\ &\quad - \gamma \lambda_{\min}(Q_c) \|x\|^2 - \gamma \lambda_{\min}(R) \|\hat{u}\|^2] \\ &\leq -\alpha [(\gamma \lambda_{\min}(Q_c) - 1 - 3\|A\|^2) \|x\|^2 \\ &\quad - \gamma \lambda_{\min}(R) \|\hat{u}\|^2 + 3\bar{W}_M^2 \sigma_M^2 + 3\varepsilon_M^2].\end{aligned}$$

Thus, we obtain

$$\begin{aligned}\dot{L}_{c1} + \dot{L}_{c2} &\leq -\alpha (\gamma \lambda_{\min}(Q_c) - 1 - 3\|A\|^2) \|x\|^2 \\ &\quad - \left(\|\tilde{W}_c^T \psi_1\| - \frac{\vartheta_M}{2} \right)^2 \\ &\quad - \alpha \gamma \lambda_{\min}(R) \|\hat{u}\|^2 + D_M,\end{aligned}\tag{8.3.38}$$

where

$$D_M = \frac{\vartheta_M^2}{4} + 3\alpha \bar{W}_M^2 \sigma_M^2 + 3\alpha \varepsilon_M^2.$$

Combining (8.3.23) and (8.3.38), it follows

$$\begin{aligned} \dot{L}(x) &\leq -\frac{\rho}{2} \lambda_{\min}[(C^+)^T C^+] \|C\tilde{x}\|^2 + d \|C\tilde{x}\| - \alpha (\gamma \lambda_{\min}(Q_c) - 1 - 3\|A\|^2) \|x\|^2 \\ &\quad - \left(\|\tilde{W}_c^T \psi_1\| - \frac{\vartheta_M}{2} \right)^2 - \alpha \gamma \lambda_{\min}(R) \|\hat{u}\|^2 + D_M \\ &\leq -\frac{\rho}{2} \lambda_{\min}[(C^+)^T C^+] \left[\|C\tilde{x}\| - \frac{d}{\rho \lambda_{\min}[(C^+)^T C^+]} \right]^2 \\ &\quad - \alpha (\gamma \lambda_{\min}(Q_c) - 1 - 3\|A\|^2) \|x\|^2 - \left(\|\tilde{W}_c^T \psi_1\| - \frac{\vartheta_M}{2} \right)^2 \\ &\quad - \alpha \gamma \lambda_{\min}(R) \|\hat{u}\|^2 + D_M + \underbrace{\frac{d^2}{2\rho \lambda_{\min}[(C^+)^T C^+]}}_{\tilde{D}_M}. \end{aligned}$$

Therefore, if $\theta_1, \theta_2, \gamma$, and α are selected to satisfy

$$\theta_1 \geq K_1^2, \quad \theta_2 \geq 1, \quad \gamma > \frac{3\|A\|^2 + 1}{\lambda_{\min}(Q_c)}, \quad (8.3.39)$$

and if one of the following inequalities holds

$$\|C\tilde{x}\| > \frac{d}{\rho \lambda_{\min}[(C^+)^T C^+]} + \sqrt{\frac{2\tilde{D}_M}{\rho \lambda_{\min}[(C^+)^T C^+]}} \triangleq \mathcal{B}_1 \quad (8.3.40)$$

or

$$\|\tilde{W}_c^T \psi_1\| > \sqrt{\tilde{D}_M} + \frac{\vartheta_M}{2} \triangleq \mathcal{B}_2,$$

then, we derive $\dot{L}(x) < 0$. By the dense property of \mathbb{R} [22], we can obtain a positive constant κ_1 ($0 < \kappa_1 \leq \|C\|$) such that $\|C\tilde{x}\| > \kappa_1 \|\tilde{x}\| > \mathcal{B}_1$. Similarly, we can derive that there exists a positive constant κ_2 ($0 < \kappa_2 \leq \|\psi_1\|$) such that $\|\tilde{W}_c^T \psi_1\| > \kappa_2 \|\tilde{W}_c\| > \mathcal{B}_2$. Then, it follows that $\dot{L}(x) < 0$, if (8.3.39) is true and if one of the following inequalities is true

$$\|\tilde{x}\| > \frac{\mathcal{B}_1}{\kappa_1} = \frac{d}{\kappa_1 \rho \lambda_{\min}[(C^+)^T C^+]} + \frac{1}{\kappa_1} \sqrt{\frac{2\tilde{D}_M}{\rho \lambda_{\min}[(C^+)^T C^+]}} \quad (8.3.41)$$

or

$$\|\tilde{W}_c\| > \frac{\mathcal{B}_2}{\kappa_2} = \frac{1}{\kappa_2} \left(\sqrt{\tilde{D}_M} + \frac{\vartheta_M}{2} \right). \quad (8.3.42)$$

By using Lyapunov's extension theorem [16, 17] (or the Lagrange stability result [20]), it can be concluded that the observer error \tilde{x} , the state x , and the NN weight estimation errors \tilde{W}_o , \tilde{Y}_o , and \tilde{W}_c are all UUB. This completes the proof.

Remark 8.3.4 It should be mentioned that, in (8.3.39) and (8.3.40), the constraints for θ_1 , θ_2 and \tilde{x} are the same as the NN observer designed earlier. In fact, a nonlinear separation principle is not applicable here. But for the proof of the NN observer-based control system, the closed-loop dynamics incorporates the observer dynamics, then we can develop simultaneous weight tuning laws for the observer NN and the critic NN.

8.3.4 Simulation Study

Consider the continuous-time input-nonaffine nonlinear system given by

$$\begin{aligned} \dot{x}_1 &= -x_1 + x_2, \\ \dot{x}_2 &= -x_1 - (1 - \sin^2(x_1)) x_2 + \sin(x_1)u + 0.1u^2, \\ y &= x_1, \end{aligned}$$

with initial conditions $x_1(0) = 1$ and $x_2(0) = -0.5$. The value function is given by (8.3.2), where Q and R are chosen as identity matrices with appropriate dimensions. It is assumed that the system dynamics is unknown, the system states are unavailable for measurements and only the input and output of the system are measurable. To estimate the system states, an NN observer is employed and the corresponding parameters are chosen as

$$A = \begin{bmatrix} 0 & 1 \\ -6 & -5 \end{bmatrix},$$

and

$$K = \begin{bmatrix} 10 \\ -2 \end{bmatrix}.$$

The observer NN is constructed by a three-layer NN with one hidden layer containing eight neurons. The input layer involves three neurons and the output layer contains two neurons. The activation function $\sigma(\cdot)$ is selected as hyperbolic tangent function $\tanh(\cdot)$. Let the design parameters be $\theta_1 = \theta_2 = 1.5$. The initial weights of W_o and Y_o are all set to be random within $[0.5, 1]$. From Figs. 8.8 and 8.9, it is clear that the errors between the actual states x_1 , x_2 and the observed states \hat{x}_1 , \hat{x}_2 quickly approach zero, which shows good the performance of the NN observer.

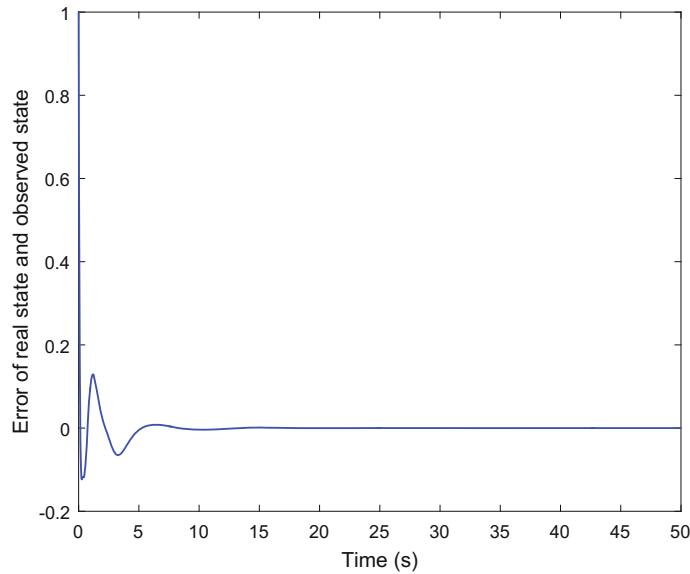


Fig. 8.8 The error between the actual state x_1 and observed state \hat{x}_1

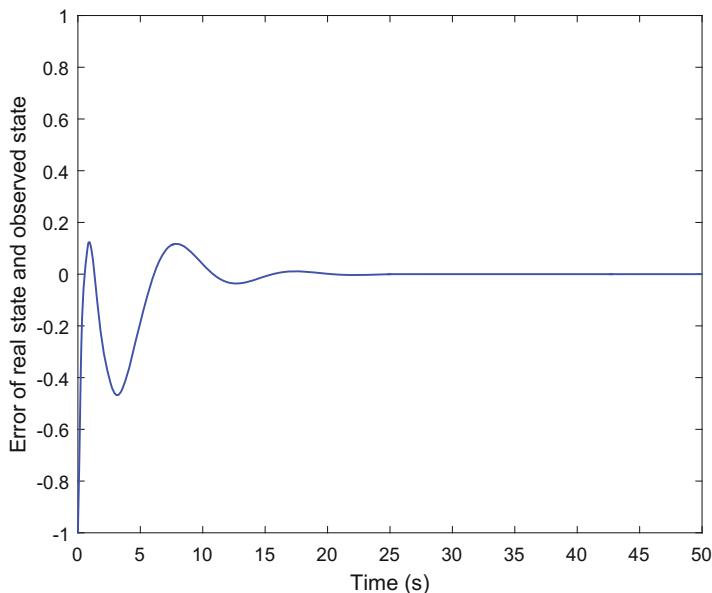


Fig. 8.9 The error between the actual state x_2 and observed state \hat{x}_2

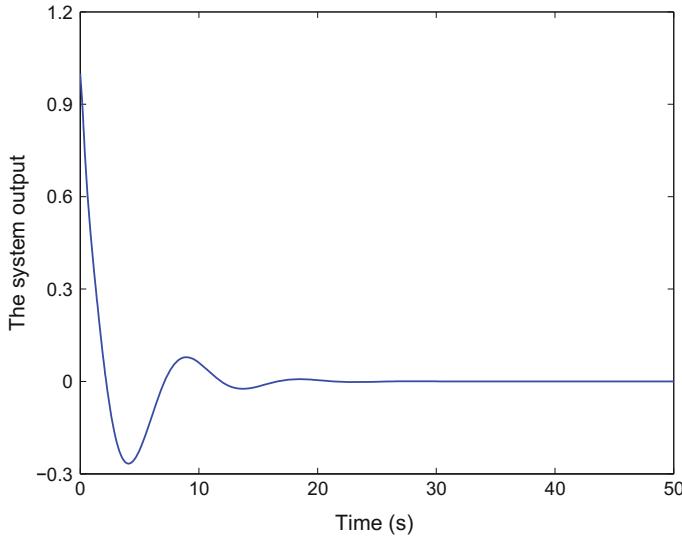


Fig. 8.10 System output y

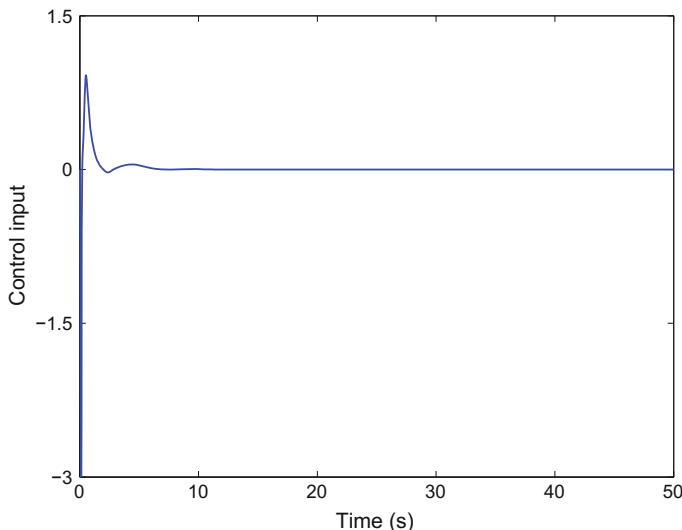


Fig. 8.11 Control input u

The activation functions of the critic NN are chosen from the sixth-order series expansion of the value function. Only polynomial terms of even order are considered, that is,

$$\sigma_c = [x_1^2, x_1 x_2, x_2^2, x_1^4, x_1^3 x_2, x_1^2 x_2^2, x_1 x_2^3, x_2^4, x_2^6, x_1^5 x_2, x_1^4 x_2^2, x_1^3 x_2^3, x_1^2 x_2^4, x_1 x_2^5, x_2^6]^T.$$

The critic NN weight vector is denoted as $\hat{W}_c = [\hat{W}_{c1}, \dots, \hat{W}_{c15}]^T$. The learning rate for the critic NN is selected as $\alpha = 0.5$. Moreover, the initial weight vector of \hat{W}_c is set to be $[1, \dots, 1]^T$. Moreover, in order to maintain the PE condition, a small exploratory signal

$$\mathcal{N}(t) = e^{-0.1t} [\sin^2 t \cos t + \sin^2(2t) \cos(0.1t)]$$

is added to the control input for the first 10 s as in [25].

Figures 8.10 and 8.11 depict the system output trajectory y and the nearly optimal control signal u , respectively. It can be seen from Figs. 8.10 and 8.11 that the present NN observer-based optimal control algorithm is effective for output regulation of unknown nonlinear systems.

8.4 Conclusions

In this chapter, using the ADP methods, an identifier–actor–critic architecture is presented to obtain the approximate optimal control for continuous-time unknown nonaffine nonlinear systems. Meanwhile, a novel ADP-based observer–critic architecture is developed to obtain the optimal output regulation for nonaffine nonlinear systems with unknown dynamics. A limitation of the present methods is that they do not consider the external disturbance of nonaffine nonlinear systems. Nevertheless, the disturbance might give rise to instability of the closed-loop system while designing optimal controllers. Accordingly, new ADP methods should be studied to derive the robust optimal control for continuous-time nonaffine nonlinear systems.

References

1. Abdollahi F, Talebi HA, Patel RV (2006) A stable neural network-based observer with application to flexible-joint manipulators. *IEEE Trans Neural Netw* 17(1):118–129
2. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791
3. Ahmed M, Riyaz S (2000) Dynamic observer-a neural net approach. *J Intell Fuzzy Syst* 9(1–2):113–127
4. Apostol TM (1974) Mathematical analysis. Addison-Wesley, Boston, MA
5. Bhasin S, Kamalapurkar R, Johnson M, Vamvoudakis KG, Lewis FL, Dixon WE (2013) A novel actor-critic-identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica* 49(1):82–92
6. Bian T, Jiang Y, Jiang ZP (2014) Adaptive dynamic programming and optimal control of nonlinear nonaffine systems. *Automatica* 50(10):2624–2632
7. Campbell SL, Meyer CD (1991) Generalized inverses of linear transformations. Dover Publications, New York
8. Goodwin GC, Payne RL (1977) Dynamic system identification: experiment design and data analysis. Academic Press, New York

9. Hayakawa T, Haddad WM, Hovakimyan N (2008) Neural network adaptive control for a class of nonlinear uncertain dynamical systems with asymptotic stability guarantees. *IEEE Trans Neural Netw* 19(1):80–89
10. Horn RA, Johnson CR (2012) Matrix analysis. Cambridge University Press, New York
11. Hornik K, Stinchcombe M, White H (1990) Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Netw* 3(5):551–560
12. Igelnik B, Pao YH (1995) Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Trans Neural Netw* 6(6):1320–1329
13. Ioannou P, Sun J (1996) Robust adaptive control. Prentice-Hall, Upper Saddle River, NJ
14. Jin G, Sain M, Pham K, Spencer B, Ramallo J (2001) Modeling MR-dampers: A nonlinear blackbox approach. In: Proceedings of the American control conference. pp 429–434
15. Khalil HK (2001) Nonlinear systems. Prentice-Hall, Upper Saddle River, NJ
16. LaSalle JP, Lefschetz S (1967) Stability by Liapunov's direct method with applications. Academic Press, New York
17. Lewis F, Jagannathan S, Yesildirak A (1999) Neural network control of robot manipulators and nonlinear systems. Taylor & Francis, London
18. Lewis FL, Vrabie D, Syrmos VL (2012) Optimal control. Wiley, Hoboken, NJ
19. Liu D, Huang Y, Wang D, Wei Q (2013) Neural-network-observer-based optimal control for unknown nonlinear systems using adaptive dynamic programming. *Int J Control* 86(9):1554–1566
20. Michel AN, Hou L, Liu D (2015) Stability of dynamical systems: on the role of monotonic and non-monotonic Lyapunov functions. Birkhäuser, Boston, MA
21. Modares H, Lewis F, Naghibi-Sistani MB (2014) Online solution of nonquadratic two-player zero-sum games arising in the H_∞ control of constrained input systems. *Int J Adap Control Signal Process* 28(3–5):232–254
22. Rudin W (1976) Principles of mathematical analysis. McGraw-Hill, New York
23. Selmic RR, Lewis FL (2001) Multimodel neural networks identification and failure detection of nonlinear systems. In: Proceedings of the IEEE conference on decision and control. pp 3128–3133
24. Theocaris J, Petridis V (1994) Neural network observer for induction motor control. *IEEE Control Syst Mag* 14(2):26–37
25. Vamvoudakis KG, Lewis FL (2010) Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* 46(5):878–888
26. Vrabie D, Lewis F (2009) Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Netw* 22(3):237–246
27. Walter E, Pronzato L (1997) Identification of parametric models from experimental data. Springer, Heidelberg
28. Yang X, Liu D, Wang D (2014) Reinforcement learning for adaptive optimal control of unknown continuous-time nonlinear systems with input constraints. *Int J Control* 87(3):553–566
29. Yang X, Liu D, Wang D, Wei Q (2014) Discrete-time online learning control for a class of unknown nonaffine nonlinear systems using reinforcement learning. *Neural Netw* 55:30–41
30. Yu W (2009) Recent advances in intelligent control systems. Springer, London
31. Yu W, Li X (2001) Some new results on system identification with dynamic neural networks. *IEEE Trans Neural Netw* 12(2):412–417
32. Zhang H, Cui L, Zhang X, Luo Y (2011) Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Trans Neural Netw* 22(12):2226–2236
33. Zhang H, Liu D, Luo Y, Wang D (2013) Adaptive dynamic programming for control: algorithms and stability. Springer, London

Chapter 9

Robust and Optimal Guaranteed Cost Control of Continuous-Time Nonlinear Systems

9.1 Introduction

Practical control systems are always subject to model uncertainties, exogenous disturbances, or other changes in their lifetime. They are often considered during the controller design process in order to avoid the performance deterioration of nominal closed-loop systems. A controller is said to be robust if it works even if the actual system deviates from its nominal model on which the controller design is based. The importance of the robust control problems is evident, and it has been recognized by control scientists for several decades [9–11, 19]. In [19], it was shown that the robust control problems could be solved by studying an optimal control problem of the nominal system. Hence, optimal control methods can be employed to design robust controllers. However, the results are restricted to a class of systems with special form of uncertainties. Though Adhyaru et al. [2, 3] proposed an HJB equation-based optimal control algorithm to deal with the nonlinear robust control problem, the algorithm was constructed using the least square method and performed offline, not to mention the stability analysis of the closed-loop optimal control system. On the other hand, when controlling a real plant, it is desirable to design a controller which guarantees not only the asymptotically stability of closed-loop system but also an adequate level of performance under uncertainties. The so-called guaranteed cost control approach [6] has the advantage of providing an upper bound on a given cost, and thus the system performance degradation incurred by the model parameter uncertainties is guaranteed to be less than this bound [58, 59]. The optimal robust guaranteed cost control problem arises when discussing the optimality of the guaranteed cost function. In this chapter, we study the robust control and optimal guaranteed cost control of uncertain nonlinear systems using the online ADP strategy.

The ADP algorithm was first proposed by Werbos [51, 52] as an effective method to solve optimization and optimal control problems. In general, it is implemented by solving the HJB equation based on function approximators, such as neural networks (NNs). It is one of the key directions for future researches in intelligent control and understanding brain intelligence [53, 54]. As a result, the ADP and related research have gained much attention from scholars across many disciplines; see, e.g., [14, 16, 24, 37, 39, 49, 61] and the numerous references therein. Significantly, the

ADP method has been often used in feedback control applications, both for discrete-time systems [8, 12, 17, 20, 22, 25, 34, 35, 43–46, 50, 57] and for continuous-time systems [1, 5, 7, 21, 23, 26, 27, 31, 36, 41, 42, 55, 60]. Besides, various traditional control problems, such as robust control [2, 3], decentralized control [29], networked control [56], power system control [18], are studied under the new framework, which greatly extends the application scope of ADP methods.

In this chapter, we first employ online policy iteration (PI) algorithm to tackle the robust control problem [47]. The robust control problem is transformed into an optimal control problem with the cost function modified to account for uncertainties. Then, an online PI algorithm is developed to solve the HJB equation by constructing and training a critic network. It is shown that an approximate closed-form expression of the optimal control law is available. Hence, there is no need to build an action network. The uniform ultimate boundedness of the closed-loop system is analyzed by using the Lyapunov approach. Since the ADP method is effective in solving optimal control problem and NNs can be constructed to facilitate the implementation process, it is convenient to employ the PI algorithm to handle robust control problem. Thus, the present robust control approach is easy to understand and implement. It can be used to solve a broad class of nonlinear robust control problems. Next, we investigate the optimal guaranteed cost control of continuous-time uncertain nonlinear systems using NN-based online solution of HJB equation [28]. The optimal robust guaranteed cost control problem is transformed into an optimal control problem by introducing an appropriate cost function. It can be proved that the optimal cost function of the nominal system is the optimal guaranteed cost of the uncertain system. Then, a critic network is constructed for facilitating the solution of modified HJB equation. Moreover, inspired by the work of [7, 36], an additional stabilizing term is introduced to ensure the stability, which relaxes the need for an initial stabilizing control. The uniform ultimate boundedness of the closed-loop system is also proved using Lyapunov's direct approach. Besides, it is shown that the approximate control input can converge to the optimal control within a small bound.

9.2 Robust Control of Uncertain Nonlinear Systems

Consider a class of continuous-time nonlinear systems described by

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) + \Delta f(x(t)), \quad (9.2.1)$$

where $x(t) \in \mathbb{R}^n$ is the state vector and $u(t) \in \mathbb{R}^m$ is the control vector, $f(\cdot)$ and $g(\cdot)$ are differentiable in their arguments with $f(0) = 0$, and $\Delta f(x(t))$ is the unknown perturbation. Here, we let $x(0) = x_0$ be the initial state.

Denote

$$\bar{f}(x) = f(x) + \Delta f(x).$$

Suppose that the function $\bar{f}(x)$ is known only up to an additive perturbation, which is bounded by a known function in the range of $g(x)$. Note that the condition for the unknown perturbation to be in the range space of $g(x)$ is called the matching condition. Thus, we write $\Delta f(x) = g(x)d(x)$ with $d(x) \in \mathbb{R}^m$, which represents the matched uncertainty of the system dynamics. Assume that the function $d(x)$ is bounded by a known function $d_M(x)$, i.e., $\|d(x)\| \leq d_M(x)$ with $d_M(0) = 0$. We also assume that $d(0) = 0$, so that $x = 0$ is an equilibrium.

For system (9.2.1), to deal with the robust control problem, we should find a feedback control law $u(x)$, such that the closed-loop system is globally asymptotically stable for all admissible uncertainties $d(x)$. Here, we will show that this problem can be converted into designing an optimal controller for the corresponding nominal system with appropriate cost function.

Consider the nominal system corresponding to (9.2.1)

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t), \quad (9.2.2)$$

where we assume that $f + gu$ is Lipschitz continuous on a set Ω in \mathbb{R}^n containing the origin, and that system (9.2.2) is controllable in the sense that there exists a continuous control law on Ω that asymptotically stabilizes the system. It is desired to find a control law $u(x)$ which minimizes the infinite horizon cost function given by

$$J(x_0, u) = \int_0^\infty \{\rho d_M^2(x(\tau)) + U(x(\tau), u(x(\tau)))\}d\tau, \quad (9.2.3)$$

where $\rho > 0$ is a constant, $U(x, u) \geq 0$ is the utility function with $U(0, 0) = 0$. In this section, the utility function is chosen as the quadratic form $U(x, u) = x^T Q x + u^T R u$, where Q and R are positive definite matrices with $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$. The cost function described in (9.2.3) gives a modification with respect to the ordinary optimal control problem, which appropriately reflects the uncertainties, regulation, and control simultaneously.

When dealing with the optimal control problem, the designed feedback control must be admissible. A control $u(x): \mathbb{R}^n \rightarrow \mathbb{R}^m$ is said to be admissible with respect to (9.2.2) on Ω , written as $u(x) \in \mathcal{A}(\Omega)$, if $u(x)$ is continuous on Ω , $u(0) = 0$, $u(x)$ stabilizes system (9.2.2) on Ω and $J(x_0, u)$ is finite for every $x_0 \in \Omega$. Given a control $\mu \in \mathcal{A}(\Omega)$, if the associated value function

$$V(x(t)) = \int_t^\infty \{\rho d_M^2(x(\tau)) + U(x(\tau), \mu(x(\tau)))\}d\tau \quad (9.2.4)$$

is continuously differentiable, then an infinitesimal version of (9.2.4) is the so-called nonlinear Lyapunov equation

$$\rho d_M^2(x) + U(x, \mu(x)) + (\nabla V(x))^T (f(x) + g(x)\mu(x)) = 0 \quad (9.2.5)$$

with $V(0) = 0$. In (9.2.5), the term $\nabla V(x)$ denotes the partial derivative of the value function $V(x)$ with respect to x , i.e., $\nabla V(x) = \frac{\partial V(x)}{\partial x}$.

Define the Hamiltonian of the problem and the optimal cost function as

$$H(x, \mu, \nabla V(x)) = \rho d_M^2(x) + U(x, \mu) + (\nabla V(x))^T(f(x) + g(x)\mu)$$

and

$$V^*(x) = \min_{\mu \in \mathcal{A}(\mathcal{Q})} \int_t^\infty \{\rho d_M^2(x(\tau)) + U(x(\tau), \mu(x(\tau)))\} d\tau. \quad (9.2.6)$$

The optimal value function defined above will be used interchangeably with optimal cost function $J^*(x_0)$. The optimal value function $V^*(x)$ satisfies the HJB equation

$$\min_{\mu \in \mathcal{A}(\mathcal{Q})} H(x, \mu, \nabla V^*(x)) = 0, \quad (9.2.7)$$

where $\nabla V^*(x) = \frac{\partial V^*(x)}{\partial x}$. Assume that the minimum on the right-hand side of (9.2.7) exists and is unique. Then, the optimal control law for the given problem is

$$u^*(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla V^*(x). \quad (9.2.8)$$

Substituting (9.2.8) into the nonlinear Lyapunov equation (9.2.5), we can obtain the HJB equation in terms of $\nabla V^*(x)$ as follows:

$$\begin{aligned} & \rho d_M^2(x) + x^T Q x + (\nabla V^*(x))^T f(x) \\ & - \frac{1}{4} (\nabla V^*(x))^T g(x) R^{-1} g^T(x) \nabla V^*(x) = 0 \end{aligned} \quad (9.2.9)$$

with $V^*(0) = 0$.

9.2.1 Equivalence Analysis and Problem Transformation

In this section, we establish the following theorem for the transformation between the robust control problem and the optimal control problem.

Theorem 9.2.1 *For nominal system (9.2.2) with the cost function (9.2.3), assume that the HJB equation (9.2.7) has a solution $V^*(x)$. Then, using this solution, the optimal control law obtained in (9.2.8) ensures closed-loop asymptotic stability of uncertain nonlinear system (9.2.1), provided that the following condition is satisfied:*

$$\rho d_M^2(x) \geq d^T(x) R d(x). \quad (9.2.10)$$

Proof Let $V^*(x)$ be the optimal solution of the HJB equation (9.2.7) and $u^*(x)$ be the optimal control law defined by (9.2.8). Now, we prove that $u^*(x)$ is a solution to the robust control problem, namely system (9.2.1) is asymptotically stable under the control $u^*(x)$ for all possible uncertainties $d(x)$. To do this, one needs to prove that $V^*(x)$ is a Lyapunov function of (9.2.1).

According to (9.2.6), $V^*(x) > 0$ for any $x \neq 0$ and $V^*(x) = 0$ when $x = 0$. This means that $V^*(x)$ is positive definite. Using (9.2.7), we have

$$(\nabla V^*(x))^\top (f(x) + g(x)u^*(x)) = -\rho d_M^2(x) - U(x, u^*(x)). \quad (9.2.11)$$

Using (9.2.8), it follows

$$2(u^*(x))^\top R = -(\nabla V^*(x))^\top g(x). \quad (9.2.12)$$

Considering (9.2.11) and (9.2.12), we find

$$\begin{aligned} \dot{V}_{(9.2.1)}^*(x) &\triangleq (\nabla V^*(x))^\top (f(x) + \Delta f(x) + g(x)u^*(x)) \\ &= -\rho d_M^2(x) - U(x, u^*(x)) - 2(u^*(x))^\top R d(x). \end{aligned} \quad (9.2.13)$$

By adding and subtracting the term $d^\top(x)Rd(x)$, (9.2.13) becomes

$$\begin{aligned} \dot{V}_{(9.2.1)}^*(x) &= -\rho d_M^2(x) - x^\top Q x + d^\top(x)Rd(x) \\ &\quad - (u^*(x) + d(x))^\top R(u^*(x) + d(x)) \\ &\leq -(\rho d_M^2(x) - d^\top(x)Rd(x)) - x^\top Q x. \end{aligned}$$

Using (9.2.10), we can conclude that $\dot{V}_{(9.2.1)}^*(x) \leq -x^\top Q x < 0$ for any $x \neq 0$. Then, the conditions for Lyapunov local stability theory are satisfied. Thus, there exists a neighborhood $\Phi = \{x: \|x(t)\| < c\}$ for some $c > 0$ such that if $x(t) \in \Phi$, then $\lim_{t \rightarrow \infty} x(t) = 0$.

However, $x(t)$ cannot remain forever outside Φ . Otherwise, $\|x(t)\| \geq c$ for all $t \geq 0$. Denote

$$q = \inf_{\|x(t)\| \geq c} \{x^\top Q x\} > 0.$$

Clearly,

$$\dot{V}_{(9.2.1)}^*(x) \leq -x^\top Q x \leq -q.$$

Then,

$$V^*(x(t)) - V^*(x(0)) = \int_0^t \dot{V}_{(9.2.1)}^*(x(\tau)) d\tau \leq -qt. \quad (9.2.14)$$

From (9.2.14), we obtain

$$V^*(x(t)) \leq V^*(x(0)) - qt \rightarrow -\infty \text{ as } t \rightarrow \infty.$$

This contradicts the fact that $V^*(x(t)) > 0$ for any $x \neq 0$. Therefore, $\lim_{t \rightarrow \infty} x(t) = 0$ no matter where the trajectory starts from in Ω . This completes the proof.

Remark 9.2.1 According to Theorem 9.2.1, if we set $\rho = 1$ and $R = I_m$, where I_m denotes the $m \times m$ identity matrix, then (9.2.10) becomes $d_M^2(x) \geq d^T(x) d(x) = \|d(x)\|^2$. Hence, in this special case, the robust control problem is equivalent to the optimal control problem without the need for any additional conditions. Otherwise, the formula (9.2.10) should be satisfied in order to ensure the equivalence of problem transformation.

In light of Theorem 9.2.1, by acquiring the solution of the HJB equation (9.2.9) and then deriving the optimal control law (9.2.8), we can obtain the robust control law for system (9.2.1) in the presence of matched uncertainties. However, due to the nonlinear nature of the HJB equation, finding its solution is often difficult. In what follows, we will introduce an online policy iteration (PI) algorithm to solve the problem based on NN techniques.

9.2.2 *Online Algorithm and Neural Network Implementation*

According to [40], PI algorithms consist of policy evaluation based on (9.2.5) and policy improvement based on (9.2.8). Its iteration procedure can be described as follows.

- Step 1: Choose a small number $\varepsilon > 0$. Let $i = 0$ and $V^{(0)} = 0$. Then, start with an initial admissible control law $\mu^{(0)}(x) \in \mathcal{A}(\Omega)$.
- Step 2: Let $i = i + 1$. Based on the control law $\mu^{(i-1)}(x)$, solve the nonlinear Lyapunov equation

$$\rho d_M^2(x) + U\left(x, \mu^{(i-1)}(x)\right) + (\nabla V^{(i)}(x))^T (f(x) + g(x)\mu^{(i-1)}(x)) = 0$$

$$\text{with } V^{(i)}(0) = 0.$$

- Step 3: Update the control law via

$$\mu^{(i)}(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla V^{(i)}(x).$$

- Step 4: If $|V^{(i)}(x) - V^{(i-1)}(x)| \leq \varepsilon$, stop and obtain the approximate optimal control law $\mu^{(i)}(x)$; else, go back to Step 2.

The algorithm will converge to the optimal value function and optimal control law, i.e., $V^{(i)}(x) \rightarrow V^*(x)$ and $\mu^{(i)}(x) \rightarrow u^*(x)$ as $i \rightarrow \infty$. The convergence proofs of the PI algorithm above have been given in [1, 4, 21, 32].

Next, we present the implementation process of the PI algorithm based on a critic NN. Assume that the value function $V(x)$ is continuously differentiable. According to the universal approximation property of NNs, $V(x)$ can be reconstructed by an NN on a compact set Ω as

$$V(x) = W_c^\top \sigma_c(x) + \varepsilon_c(x), \quad (9.2.15)$$

where $W_c \in \mathbb{R}^l$ is the ideal weight vector, $\sigma_c(x) \in \mathbb{R}^l$ is the activation function, l is the number of neurons in the hidden layer, and $\varepsilon_c(x)$ is the approximation error of the NN. Then,

$$\nabla V(x) = (\nabla \sigma_c(x))^\top W_c + \nabla \varepsilon_c(x), \quad (9.2.16)$$

where $\nabla V(x) = \partial V(x) / \partial x$, $\nabla \sigma_c(x) = \partial \sigma_c(x) / \partial x \in \mathbb{R}^{l \times n}$ and $\nabla \varepsilon_c(x) = \partial \varepsilon_c(x) / \partial x \in \mathbb{R}^n$ are the gradient of the activation function and NN approximation error, respectively.

Using (9.2.16), the Lyapunov equation (9.2.5) takes the following form:

$$\rho d_M^2(x) + U(x, \mu) + (W_c^\top \nabla \sigma_c(x) + (\nabla \varepsilon_c(x))^\top) \dot{x}_{(9.2.2)} = 0, \quad (9.2.17)$$

where we use $\dot{x}_{(9.2.2)}$ to indicate it is from (9.2.2). Assume that the weight vector W_c , the gradient $\nabla \sigma_c(x)$, and the approximation error $\varepsilon_c(x)$ and its derivative $\nabla \varepsilon_c(x)$ are all bounded on a compact set Ω . We also have $\varepsilon_c(x) \rightarrow 0$ and $\nabla \varepsilon_c(x) \rightarrow 0$ as $l \rightarrow \infty$ [41].

Since the ideal weights are unknown, a critic NN can be built in terms of the estimated weights as

$$\hat{V}(x) = \hat{W}_c^\top \sigma_c(x) \quad (9.2.18)$$

to approximate the value function. In (9.2.18), $\sigma_c(x)$ is selected such that $\hat{V}(x) > 0$ for any $x \neq 0$ and $\hat{V}(x) = 0$ when $x = 0$. Then,

$$\nabla \hat{V}(x) = (\nabla \sigma_c(x))^\top \hat{W}_c, \quad (9.2.19)$$

where $\nabla \hat{V}(x) = \partial \hat{V}(x) / \partial x$.

The approximate Hamiltonian can be derived as

$$H(x, \mu, \hat{W}_c) = \rho d_M^2(x) + U(x, \mu) + \hat{W}_c^\top \nabla \sigma_c(x) \dot{x}_{(9.2.2)} \triangleq e_c. \quad (9.2.20)$$

To train the critic NN, it is desired to design \hat{W}_c to minimize the objective function

$$E_c = \frac{1}{2} e_c^2.$$

We employ the standard steepest descent algorithm to tune the weights of the critic network, i.e.,

$$\dot{\hat{W}}_c = -\alpha_c \left[\frac{\partial E_c}{\partial \hat{W}_c} \right] = -\alpha_c e_c \left[\frac{\partial e_c}{\partial \hat{W}_c} \right], \quad (9.2.21)$$

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

Using (9.2.17), the Hamiltonian becomes

$$H(x, \mu, W_c) = \rho d_M^2(x) + U(x, \mu) + W_c^\top \nabla \sigma_c(x) \dot{x} \quad (9.2.22)$$

where $e_{cH} = -(\nabla \varepsilon_c(x))^\top \dot{x}$ is the residual error due to the NN approximation.

Denote $\theta = \nabla \sigma_c(x) \dot{x}$. Assume that there exists a constant $\theta_M > 0$ such that $\|\theta\| \leq \theta_M$, and let the weight estimation error of the critic NN be $\tilde{W}_c = W_c - \hat{W}_c$. Then, considering (9.2.20) and (9.2.22), we have $e_{cH} - e_c = \tilde{W}_c^\top \theta$. Therefore, the dynamics of weight estimation error is

$$\dot{\tilde{W}}_c = -\dot{\hat{W}}_c = \alpha_c (e_{cH} - \tilde{W}_c^\top \theta) \theta. \quad (9.2.23)$$

The PE condition is required to tune the critic network ensuring $\|\theta\| \geq \theta_m$, where $\theta_m > 0$ is a constant. A small exploratory signal will be added to the system in order to satisfy the PE condition.

When implementing the online PI algorithm, for the purpose of policy improvement, we should obtain the policy that can minimize the value function in (9.2.15). Hence, according to (9.2.8) and (9.2.16),

$$\mu(x) = -\frac{1}{2} R^{-1} g^\top(x) ((\nabla \sigma_c(x))^\top W_c + \nabla \varepsilon_c(x)). \quad (9.2.24)$$

The approximate control law can be formulated as

$$\hat{\mu}(x) = -\frac{1}{2} R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top \hat{W}_c. \quad (9.2.25)$$

Equation (9.2.25) implies that based on the trained critic NN, the approximate control law can be derived directly. The actor-critic architecture is maintained, but training of the action NN is not required in this case since we have closed-form solution available. The diagram of the online PI algorithm is depicted in Fig. 9.1.

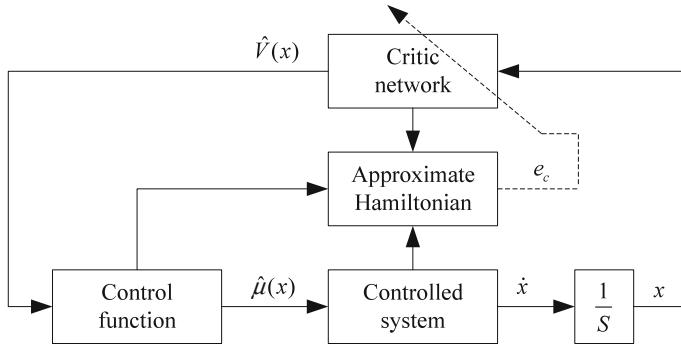


Fig. 9.1 Diagram of the online PI algorithm (solid lines represent signals and the dashed line represents the backpropagation path)

9.2.3 Stability Analysis of Closed-Loop System

The weight estimation dynamics and the closed-loop system based on the approximate optimal control law are uniformly ultimately bounded (UUB) as shown in the following two theorems.

Theorem 9.2.2 *For the controlled system (9.2.2), the weight update law for tuning the critic NN is given by (9.2.21). Then, the dynamics of the weight estimation error of the critic NN is UUB.*

Proof Select the Lyapunov function candidate as $L(x) = (1/\alpha_c) \tilde{W}_c^\top \tilde{W}_c$. Taking the time derivative of $L(x)$ along the trajectory of error dynamics (9.2.23) and considering the Cauchy–Schwarz inequality, it follows

$$\begin{aligned}
 \dot{L}(x) &= \frac{2}{\alpha_c} (\tilde{W}_c^\top \dot{\tilde{W}}_c) \\
 &= \frac{2}{\alpha_c} [\tilde{W}_c^\top \alpha_c (e_{cH} - \tilde{W}_c^\top \theta) \theta] \\
 &= \frac{2}{\alpha_c} [e_{cH} \alpha_c \tilde{W}_c^\top \theta - \alpha_c (\tilde{W}_c^\top \theta)^2] \\
 &\leq \frac{2}{\alpha_c} \left[\frac{1}{2} e_{cH}^2 + \frac{1}{2} \alpha_c^2 (\tilde{W}_c^\top \theta)^2 - \alpha_c (\tilde{W}_c^\top \theta)^2 \right] \\
 &= \frac{1}{\alpha_c} e_{cH}^2 - (2 - \alpha_c) (\tilde{W}_c^\top \theta)^2.
 \end{aligned}$$

We can conclude that $\dot{L}(x) < 0$ as long as $0 < \alpha_c < 2$ and

$$|\tilde{W}_c^\top \theta| > \sqrt{\frac{e_{cH}^2}{\alpha_c(2 - \alpha_c)}}.$$

By employing the dense property of real numbers [38], we derive that there exist a positive constant κ ($0 < \kappa \leq \theta_M$) such that

$$|\tilde{W}_c^\top \theta| > \kappa \|\tilde{W}_c\| > \sqrt{\frac{e_{cH}^2}{\alpha_c(2 - \alpha_c)}}. \quad (9.2.26)$$

By noticing (9.2.26), we can conclude that $\dot{L}(x) < 0$ as long as

$$\begin{cases} 0 < \alpha_c < 2, \\ \|\tilde{W}_c\| > \sqrt{\frac{e_{cH}^2}{\alpha_c \kappa^2 (2 - \alpha_c)}}. \end{cases} \quad (9.2.27)$$

By Lyapunov's extension theorem [13, 15] (or the Lagrange stability result [30]), the dynamics of the weight estimation error is UUB. The norm of the weight estimation error is bounded as well. This completes the proof.

Theorem 9.2.3 *For system (9.2.2), the weight update law of the critic NN given by (9.2.21) and the approximate optimal control law obtained by (9.2.25) ensure that, for any initial state x_0 , there exists a time $T(x_0, M)$ such that $x(t)$ is UUB. Here, the bound $M > 0$ is given by*

$$\|x(t)\| \leq \sqrt{\frac{\beta_M}{\rho \rho_0^2 + \lambda_{\min}(Q)}} \equiv M, \quad t \geq T,$$

where $\beta_M > 0$ and $\rho_0 > 0$ are constants to be determined later and $\lambda_{\min}(Q)$ is the smallest eigenvalue of Q .

Proof Taking the time derivative of $V(x)$ along the trajectory of (9.2.2) generated by the approximate control law $\hat{\mu}(x)$, we can obtain

$$\dot{V}_{(9.2.2)} = (\nabla V(x))^\top (f(x) + g(x)\hat{\mu}). \quad (9.2.28)$$

Using (9.2.9), we can find

$$0 = \rho d_M^2(x) + x^\top Q x + (\nabla V(x))^\top f(x) - \frac{1}{4} (\nabla V(x))^\top g(x) R^{-1} g^\top(x) \nabla V(x). \quad (9.2.29)$$

Considering (9.2.29), (9.2.28) becomes

$$\begin{aligned} \dot{V}_{(9.2.2)} = & -\rho d_M^2(x) - x^\top Q x + \frac{1}{4} (\nabla V(x))^\top g(x) R^{-1} g^\top(x) \nabla V(x) \\ & + (\nabla V(x))^\top g(x) \hat{\mu}. \end{aligned} \quad (9.2.30)$$

Adding and subtracting $(\nabla V(x))^T g(x) \mu$ to (9.2.30) and using (9.2.24) and (9.2.25), it follows

$$\begin{aligned}\dot{V}_{(9.2.2)} &= -\rho d_M^2(x) - x^T Q x + \frac{1}{4} (\nabla V(x))^T g(x) R^{-1} g^T(x) \nabla V(x) \\ &\quad + (\nabla V(x))^T g(x) \mu + (\nabla V(x))^T g(x) \hat{\mu} - (\nabla V(x))^T g(x) \mu \\ &= -\rho d_M^2(x) - x^T Q x - \frac{1}{4} (\nabla V(x))^T g(x) R^{-1} g^T(x) \nabla V(x) \\ &\quad + \frac{1}{2} (\nabla V(x))^T g(x) R^{-1} g^T(x) (\nabla V(x) - \nabla \hat{V}(x)).\end{aligned}\quad (9.2.31)$$

Substituting (9.2.16) and (9.2.19) into (9.2.31), we can further obtain

$$\begin{aligned}\dot{V}_{(9.2.2)} &= -\rho d_M^2(x) - x^T Q x - \frac{1}{4} (\nabla V(x))^T g(x) R^{-1} g^T(x) \nabla V(x) \\ &\quad + \frac{1}{2} (W_c^T \nabla \sigma_c(x) + (\nabla \varepsilon_c(x))^T) g(x) R^{-1} g^T(x) \\ &\quad \times ((\nabla \sigma_c(x))^T \tilde{W}_c + \nabla \varepsilon_c(x)).\end{aligned}$$

Here, we denote

$$\beta = \frac{1}{2} (W_c^T \nabla \sigma_c(x) + (\nabla \varepsilon_c(x))^T) g(x) R^{-1} g^T(x) ((\nabla \sigma_c(x))^T \tilde{W}_c + \nabla \varepsilon_c(x)).$$

Considering the fact that R^{-1} is positive definite, the assumption that W_c , $\nabla \sigma_c(x)$, and $\nabla \varepsilon_c(x)$ are bounded, and Theorem 9.2.2, we can conclude that β is upper bounded by $\beta \leq \beta_M$, where $\beta_M > 0$ is a constant. Therefore, $\dot{V}_{(9.2.2)}$ takes the following form:

$$\dot{V}_{(9.2.2)} \leq -\rho d_M^2(x) - x^T Q x + \beta_M. \quad (9.2.32)$$

In many cases, we can determine a quadratic bound of $d(x)$. Under such circumstances, we assume that $d_M(x) = \rho_0 \|x\|$, where $\rho_0 > 0$ is a constant. Then, (9.2.32) becomes

$$\dot{V}_{(9.2.2)} \leq -(\rho \rho_0^2 + \lambda_{\min}(Q)) \|x\|^2 + \beta_M.$$

Hence, we can observe that $\dot{V}_{(9.2.2)} < 0$ whenever $x(t)$ lies outside the compact set

$$\Omega_x = \left\{ x : \|x\| \leq \sqrt{\frac{\beta_M}{\rho \rho_0^2 + \lambda_{\min}(Q)}} \right\}.$$

Therefore, based on the approximate optimal control law, the state trajectories of the closed-loop system are UUB and $\|x(t)\| \leq M$. This completes the proof.

In the next theorem, the equivalence of the NN-based HJB solution of the optimal control problem and the solution of robust control problem is established.

Theorem 9.2.4 Assume that the NN-based HJB solution of the optimal control problem exists. Then, the control law defined by (9.2.25) ensures closed-loop asymptotic stability of uncertain nonlinear system (9.2.1) if the condition described in (9.2.10) is satisfied.

Proof Let $\hat{V}(x)$ be the solution of

$$\rho d_M^2(x) + U(x, \hat{\mu}(x)) + (\nabla \hat{V}(x))^T (f(x) + g(x)\hat{\mu}(x)) = 0$$

and $\hat{\mu}(x)$ be the approximate optimal control law defined by (9.2.25). Then,

$$2\hat{\mu}^T(x)R = -(\nabla \hat{V}(x))^T g(x).$$

Now, we show that with the approximate optimal control $\hat{\mu}(x)$, the closed-loop system remains asymptotically stable for all possible uncertainties $d(x)$. According to (9.2.18) and the selection of $\sigma_c(x)$, we have $\hat{V}(0) = 0$ and $\hat{V}(x) > 0$ when $x \neq 0$. Taking the manipulations similar to the proof of Theorem 9.2.1, we can easily obtain

$$\dot{\hat{V}}(x) \leq -x^T Q x,$$

which implies that $\dot{\hat{V}}(x) < 0$ for any $x \neq 0$. This completes the proof.

Remark 9.2.2 In [47], an iterative algorithm for online design of robust control for a class of continuous-time nonlinear systems was developed; however, the optimality of the robust controller with respect to a specified cost function was not discussed. In fact, recently, there are few results on robust optimal control of uncertain nonlinear systems based on ADP, not to mention the decentralized optimal control of large-scale systems. In [48], the robust optimal control scheme for a class of uncertain nonlinear systems via ADP technique and without using an initial admissible control was established. In addition, the developed results of [48] were also extended to deal with the decentralized optimal control for a class of continuous-time nonlinear interconnected systems.

9.2.4 Simulation Study

Consider the following continuous-time nonlinear system:

$$\begin{aligned} \dot{x} = & \begin{bmatrix} -x_1 + x_2 \\ -0.5x_1 - 0.5x_2(1 - (\cos(2x_1) + 2)^2) \end{bmatrix} \\ & + \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix}(u + 0.5px_1 \sin x_2), \end{aligned} \quad (9.2.33)$$

where $x = [x_1, x_2]^\top \in \mathbb{R}^2$ and $u \in \mathbb{R}$ are the state and control variables, respectively, and p is an unknown parameter. The term $d(x) = 0.5px_1 \sin x_2$ reflects the uncertainty of the control plant. For simplicity, we assume that $p \in [-1, 1]$. Here, we choose $d_M(x) = \|x\|$ and we select $\rho = 1$ for the purpose of simulation.

We aim at obtaining a robust control law that can stabilize system (9.2.33) for all possible p . This problem can be formulated as the following optimal control problem. For the nominal system, we need to find a feedback control law $u(x)$ that minimizes the cost function

$$J(x_0) = \int_0^\infty \{\|x\|^2 + x^\top Qx + u^\top Ru\} d\tau, \quad (9.2.34)$$

where $Q = I_2$ and $R = 2$. Based on the procedure proposed in [33], the optimal value function and the optimal control law of the problem are

$$V^*(x) = x_1^2 + 2x_2^2$$

and

$$u^*(x) = -(\cos(2x_1) + 2)x_2.$$

We adopt the online PI algorithm to tackle the optimal control problem for the nominal system, where a critic network is constructed to approximate the value function. We denote the weight vector of the critic network as $\hat{W}_c = [\hat{W}_{c1}, \hat{W}_{c2}, \hat{W}_{c3}]^\top$. During the simulation process, the initial weights of the critic network are chosen randomly in $[0, 2]$ and the weight normalization is not used. The activation function of the critic network is chosen as $\sigma_c(x) = [x_1^2, x_1x_2, x_2^2]^\top$, so the ideal weight is $[1, 0, 2]^\top$. Let the learning rate of the critic network be $\alpha_c = 0.1$ and the initial state of the controlled plant be $x_0 = [1, -1]^\top$.

During the implementation process of the PI algorithm, the following small sinusoidal exploratory signal with various frequencies will be added to satisfy the PE condition,

$$\begin{aligned} \mathcal{N}(t) = & \sin^2(t) \cos(t) + \sin^2(2t) \cos(0.1t) + \sin^2(-1.2t) \cos(0.5t) \\ & + \sin^5(t) + \sin^2(1.12t) + \cos(2.4t) \sin^3(2.4t). \end{aligned} \quad (9.2.35)$$

It is introduced into the control input and thus affect the system states. The weights of the critic network converge to $[0.9978, 0.0008, 1.9997]^\top$ as shown in Fig. 9.2a, which displays a good approximation of the ideal ones. Actually, we can observe that the convergence of the weight occurred after 750 s. Then, the exploratory signal is turned off. The evolution of the state trajectory is depicted in Fig. 9.2b. We see that the state converges to zero quickly after the exploratory signal is turned off.

Using (9.2.18) and (9.2.25), the value function and control law can be approximated as

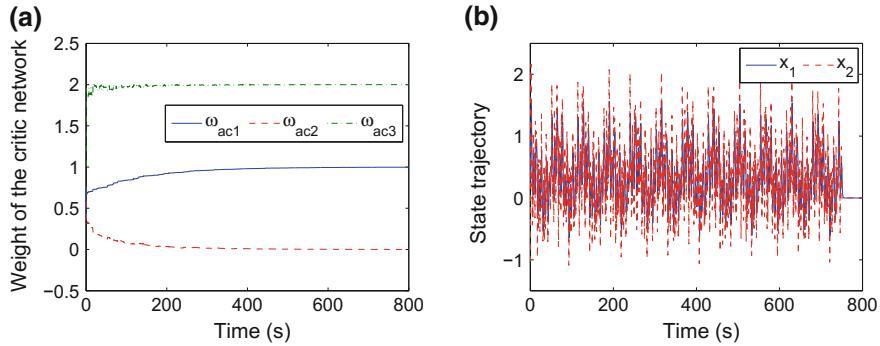


Fig. 9.2 Simulation results. **a** Convergence of the weight vector of the critic network (ω_{ac1} , ω_{ac2} , and ω_{ac3} represent \hat{W}_{c1} , \hat{W}_{c2} , and \hat{W}_{c3} , respectively). **b** Evolution of the state trajectory during the implementation process

$$\hat{V}(x) = \begin{bmatrix} 0.9978 \\ 0.0008 \\ 1.9997 \end{bmatrix}^T \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$$

and

$$\hat{\mu}(x) = -\frac{1}{2} R^{-1} \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 \\ x_2 & x_1 \\ 0 & 2x_2 \end{bmatrix}^T \begin{bmatrix} 0.9978 \\ 0.0008 \\ 1.9997 \end{bmatrix}, \quad (9.2.36)$$

respectively. The error between the optimal cost function and the approximate one is presented in Fig. 9.3a. The error between the optimal control law and the approximate version is displayed in Fig. 9.3b. Both approximation errors are close to zero, which verifies good performance of the learning algorithm.

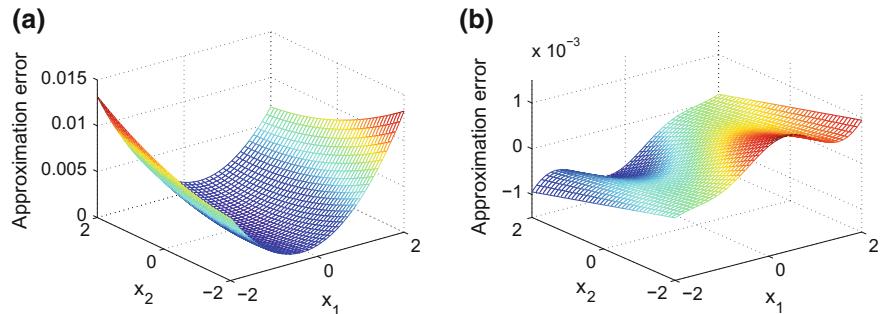


Fig. 9.3 Simulation results. **a** 3D plot of the approximation error of the cost function, i.e., $V^*(x) - \hat{V}(x)$. **b** 3D plot of the approximation error of the control law, i.e., $u^*(x) - \hat{\mu}(x)$

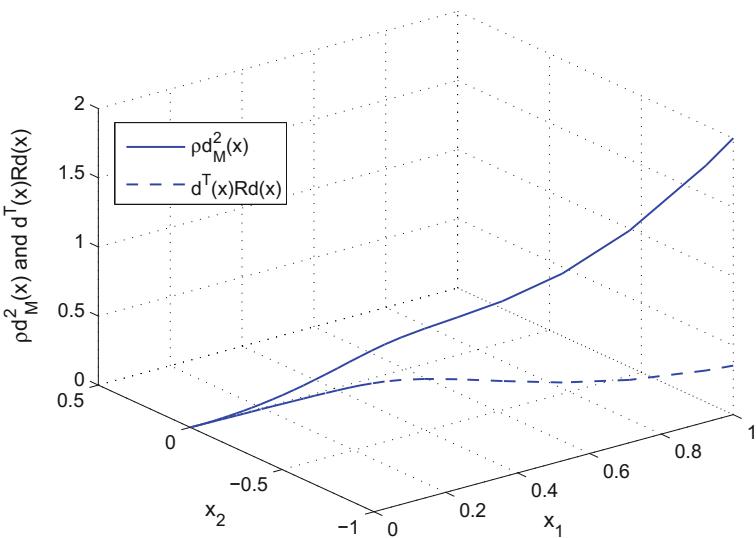


Fig. 9.4 Verification of condition (9.2.10)

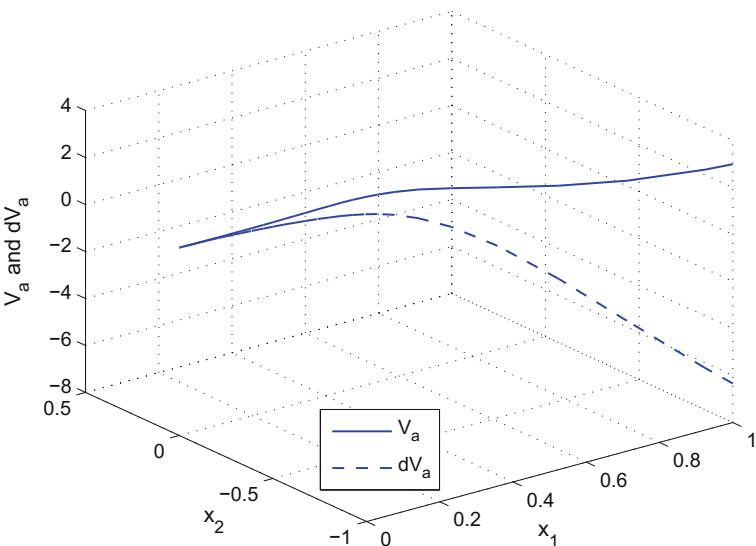


Fig. 9.5 The Lyapunov function and its derivative (V_a and dV_a represent \hat{V} and $\dot{\hat{V}}$, respectively)

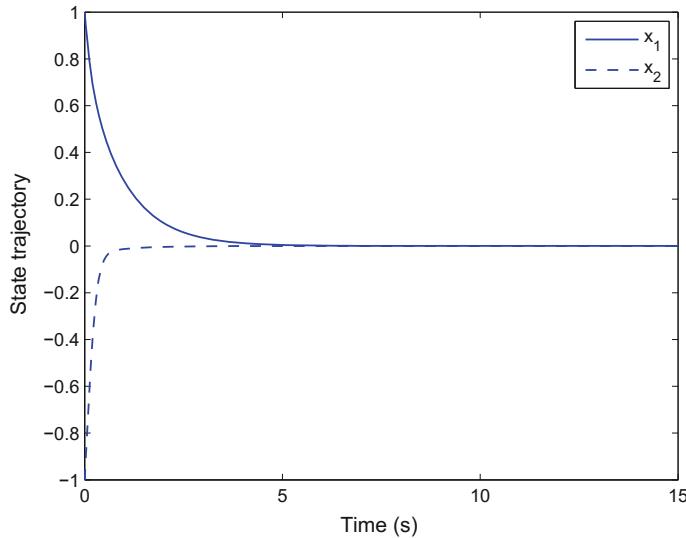


Fig. 9.6 The state trajectory under the robust control law $\hat{\mu}(x)$ when setting $p = 1$

Next, the scalar parameter $p = 1$ is chosen for evaluating the robust control performance. On the one hand, as shown in Fig. 9.4, the condition of Theorem 9.2.1 is satisfied. On the other hand, for all the values of x , the Lyapunov function $\hat{V}(x) \geq 0$ and its derivative $\dot{\hat{V}}(x) \leq 0$, which are described in Fig. 9.5. Therefore, the control law designed by solving the NN-based HJB equation is the robust control of the original uncertain nonlinear system (9.2.33). In other words, we can employ the control law (9.2.36) to stabilize system (9.2.33). In fact, under the action of the control law, the state vector converges to the equilibrium point. Figure 9.6 presents the evolution process of the state trajectory when applying the control law to system (9.2.33) for 15 s. These simulation results demonstrate the effectiveness of the robust control method established in this section.

9.3 Optimal Guaranteed Cost Control of Uncertain Nonlinear Systems

In this section, we study the optimal guaranteed cost control of uncertain nonlinear systems using the ADP approach [28]. Consider a class of continuous-time uncertain nonlinear systems given by

$$\begin{aligned}\dot{x}(t) &= \bar{F}(x(t), u(t)) \\ &= f(x(t)) + g(x(t))u(t) + \Delta f(x(t)),\end{aligned}\quad (9.3.1)$$

where $x(t) \in \mathbb{R}^n$ is the state vector and $u(t) \in \mathbb{R}^m$ is the control input. The known functions $f(\cdot)$ and $g(\cdot)$ are differentiable in their arguments with $f(0) = 0$, and $\Delta f(x(t))$ is the nonlinear perturbation of the corresponding nominal system

$$\dot{x}(t) = F(x(t), u(t)) = f(x(t)) + g(x(t))u(t). \quad (9.3.2)$$

Here, we let $x(0) = x_0$ be the initial state. In addition, we assume that $f + gu$ is Lipschitz continuous on a set Ω in \mathbb{R}^n containing the origin and that the system (9.3.2) is controllable on Ω .

Before proceeding further, we assign an explicit structure to the system uncertainty. The following assumption is given, which has been used in [9, 11].

Assumption 9.3.1 Assume that the uncertainty $\Delta f(x)$ has the form

$$\Delta f(x) = G(x) d(\varphi(x)), \quad (9.3.3)$$

where

$$d^\top(\varphi(x))d(\varphi(x)) \leq h^\top(\varphi(x))h(\varphi(x)). \quad (9.3.4)$$

In (9.3.3) and (9.3.4), $G(\cdot) \in \mathbb{R}^{n \times r}$ and $\varphi(\cdot)$ satisfying $\varphi(0) = 0$ are known functions denoting the structure of the uncertainty, $d(\cdot) \in \mathbb{R}^r$ is an uncertain function with $d(0) = 0$, and $h(\cdot) \in \mathbb{R}^r$ is a given function with $h(0) = 0$.

Consider system (9.3.1) with infinite horizon cost function

$$\bar{J}(x_0, u) = \int_0^\infty U(x(\tau), u(\tau))d\tau, \quad (9.3.5)$$

where

$$U(x, u) = Q(x) + u^\top R u,$$

$Q(x) \geq 0$, and $R = R^\top > 0$ is a constant matrix.

In this section, the aim of solving the robust guaranteed cost control problem is to find a feedback control function $u(x)$ and determine a finite upper bound function $\Phi(u)$, i.e., $\Phi(u) < +\infty$, such that the closed-loop system is robustly stable and the cost function (9.3.5) satisfies $\bar{J} \leq \Phi$. Here, the upper bound function $\Phi(u)$ is termed the robust guaranteed cost function. Only when $\Phi(u)$ is minimized, it is named the optimal robust guaranteed cost and is denoted as Φ^* , i.e., $\Phi^* = \inf_u \Phi(u)$. Additionally, the corresponding control function \bar{u}^* is called the optimal robust guaranteed cost control, i.e.,

$$\bar{u}^* = \arg \inf_u \Phi(u).$$

Next, we will prove that the optimal robust guaranteed cost control problem of system (9.3.1) can be transformed into an optimal control problem of the nominal system (9.3.2). The ADP technique can be employed to deal with the optimal control

problem of system (9.3.2). Note that in this section, the feedback control $u(x)$ is often written as u for simplicity.

9.3.1 Optimal Guaranteed Cost Controller Design

In this section, we show that the guaranteed cost of the uncertain nonlinear system is closely related to the modified cost function of the nominal system. The next theorem is from [10] with relaxed conditions.

Theorem 9.3.1 *Assume that there exist a continuously differentiable and radially unbounded value function $V(x)$ satisfying $V(x) > 0$ for all $x \neq 0$ and $V(0) = 0$, a bounded function $\Gamma(x)$ satisfying $\Gamma(x) \geq 0$, and a feedback control function $u(x)$ such that*

$$(\nabla V(x))^T \bar{F}(x, u) \leq (\nabla V(x))^T F(x, u) + \Gamma(x), \quad (9.3.6)$$

$$(\nabla V(x))^T F(x, u) + \Gamma(x) < 0, \quad x \neq 0, \quad (9.3.7)$$

$$U(x, u) + (\nabla V(x))^T F(x, u) + \Gamma(x) = 0, \quad (9.3.8)$$

where the symbol $\nabla V(x)$ denotes the partial derivative of the value function $V(x)$ with respect to x , i.e., $\nabla V(x) = \partial V(x)/\partial x$. Then, with the feedback control function $u(x)$, there exists a neighborhood of the origin such that system (9.3.1) is asymptotically stable. Furthermore,

$$\bar{J}(x_0, u) \leq J(x_0, u) = V(x_0), \quad (9.3.9)$$

where $J(x_0, u)$ is defined as

$$J(x_0, u) = \int_0^\infty \{U(x(\tau), u(x(\tau))) + \Gamma(x(\tau))\} d\tau, \quad (9.3.10)$$

and is called the modified cost function of system (9.3.2).

Proof First, we show the asymptotic stability of system (9.3.1) under the feedback control $u(x)$. Let

$$\dot{V}_{(9.3.1)}(x) = (\nabla V(x))^T \bar{F}(x, u).$$

Considering (9.3.6) and (9.3.7), we obtain $\dot{V}_{(9.3.1)}(x) < 0$ for any $x \neq 0$. This implies that $V(\cdot)$ is a Lyapunov function for system (9.3.1), which proves the asymptotic stability.

According to the definitions of $J(x_0, u)$ and $\bar{J}(x_0, u)$, we can see that

$$\bar{J}(x_0, u) \leq J(x_0, u), \quad (9.3.11)$$

since $\Gamma(x) \geq 0$.

When $\Delta f(x) = 0$, we can still find that (9.3.6)–(9.3.8) are true since $\Gamma(x) \geq 0$. In this case, we derive

$$\dot{V}_{(9.3.2)}(x) = (\nabla V(x))^T F(x, u).$$

Then,

$$U(x, u) + \Gamma(x) = -\dot{V}_{(9.3.2)}(x) + (\nabla V(x))^T F(x, u) + U(x, u) + \Gamma(x).$$

Based on (9.3.8), we obtain

$$U(x, u) + \Gamma(x) = -\dot{V}_{(9.3.2)}(x).$$

Similarly, by integrating over $[0, t]$, considering $V(x(t)) = 0$ as $t \rightarrow \infty$, we have

$$\int_0^t \{U(x, u) + \Gamma(x)\} d\tau = -V(x(t)) + V(x_0).$$

Here, letting $t \rightarrow \infty$ yields

$$J(x_0, u) = V(x_0). \quad (9.3.12)$$

Based on (9.3.11) and (9.3.12), we can easily find that (9.3.9) is true. This completes the proof.

Theorem 9.3.1 shows that the bounded function $\Gamma(x)$ takes an important role in deriving the guaranteed cost of the controlled system. The following lemma presents a specific form of $\Gamma(x)$.

Lemma 9.3.1 *For any continuously differentiable and radially unbounded function $V(x)$, define*

$$\Gamma(x) = h^T(\varphi(x)) h(\varphi(x)) + \frac{1}{4} (\nabla V(x))^T G(x) G^T(x) \nabla V(x). \quad (9.3.13)$$

Then,

$$(\nabla V(x))^T \Delta f(x) \leq \Gamma(x). \quad (9.3.14)$$

Proof Considering (9.3.3), (9.3.4), and (9.3.13), since

$$\begin{aligned}
0 &\leq \left(d(\varphi(x)) - \frac{1}{2} G^\top(x) \nabla V(x) \right)^\top \left(d(\varphi(x)) - \frac{1}{2} G^\top(x) \nabla V(x) \right) \\
&= d^\top(\varphi(x)) d(\varphi(x)) + \frac{1}{4} (\nabla V(x))^\top G(x) G^\top(x) \nabla V(x) - (\nabla V(x))^\top G(x) d(\varphi(x)) \\
&\leq h^\top(\varphi(x)) h(\varphi(x)) + \frac{1}{4} (\nabla V(x))^\top G(x) G^\top(x) \nabla V(x) - (\nabla V(x))^\top \Delta f(x) \\
&= \Gamma(x) - (\nabla V(x))^\top \Delta f(x),
\end{aligned}$$

we can see that (9.3.14) holds. This completes the proof.

Remark 9.3.1 For any continuously differentiable and radially unbounded function $V(x)$, since

$$(\nabla V(x))^\top \bar{F}(x, u) = (\nabla V(x))^\top F(x, u) + (\nabla V(x))^\top \Delta f(x), \quad (9.3.15)$$

we can easily find that the bounded function (9.3.13) satisfies (9.3.6). Note that Lemma 9.3.1 seems only to imply (9.3.6), but in fact, it presents a specific form of $\Gamma(x)$ satisfying (9.3.6), (9.3.7), and (9.3.8). The reason is that (9.3.7) and (9.3.8) are implicit assumptions of Theorem 9.3.1, noticing the framework of the generalized HJB equation [4] and the fact that $(\nabla V(x))^\top F(x, u) + \Gamma(x) = -U(x, u) < 0$ when $x \neq 0$. Hence, it can be used for problem transformation. In fact, based on (9.3.6) and (9.3.15), we can find that the positive semidefinite bounded function $\Gamma(x)$ gives an upper bound of the term $(\nabla V(x))^\top \Delta f(x)$, which facilitates us to solve the optimal robust guaranteed cost control problem of a class of nonlinear systems with uncertainties.

Remark 9.3.2 It is important to note that Theorem 9.3.1 indicates the existence of the guaranteed cost of the uncertain nonlinear system (9.3.1). In addition, in order to derive the optimal guaranteed cost controller, we should minimize the upper bound $J(x_0, u)$ with respect to u . Therefore, we should solve the optimal control problem of system (9.3.2) with $V(x_0)$ considered as the value function.

For system (9.3.2), observing that

$$\begin{aligned}
V(x_0) &= \int_0^\infty \{U(x, u) + \Gamma(x)\} d\tau \\
&= \int_0^T \{U(x, u) + \Gamma(x)\} d\tau + V(x(T)),
\end{aligned} \quad (9.3.16)$$

we have

$$\lim_{T \rightarrow 0} \frac{1}{T} \left(V(x(T)) - V(x_0) + \int_0^T \{U(x, u) + \Gamma(x)\} d\tau \right) = 0. \quad (9.3.17)$$

Clearly, (9.3.17) is equivalent to (9.3.8). Hence, (9.3.8) is an infinitesimal version of the modified value function (9.3.16) and is the so-called nonlinear Lyapunov equation.

For system (9.3.2) with modified value function (9.3.16), define the Hamiltonian of the optimal control problem as

$$H(x, u, \nabla V(x)) = U(x, u) + (\nabla V(x))^T F(x, u) + \Gamma(x). \quad (9.3.18)$$

Define the optimal cost function of system (9.3.2) as

$$V^*(x_0) = \min_{u \in \mathcal{A}(\Omega)} J(x_0, u),$$

where $J(x_0, u)$ is given in (9.3.10). Note that $V^*(x)$ satisfies the modified HJB equation

$$0 = \min_{u \in \mathcal{A}(\Omega)} H(x, u, \nabla V^*(x)), \quad (9.3.19)$$

where $\nabla V^*(x) = \partial V^*(x) / \partial x$. Assume that the minimum on the right-hand side of (9.3.19) exists and is unique. Then, the optimal control of system (9.3.2) is

$$\begin{aligned} u^*(x) &= \arg \min_{u \in \mathcal{A}(\Omega)} H(x, u, \nabla V^*(x)) \\ &= -\frac{1}{2} R^{-1} g^T(x) \nabla V^*(x). \end{aligned} \quad (9.3.20)$$

Hence, the modified HJB equation becomes

$$\begin{aligned} 0 &= U(x, u^*) + (\nabla V^*(x))^T F(x, u^*) + h^T(\varphi(x)) h(\varphi(x)) \\ &\quad + \frac{1}{4} (\nabla V^*(x))^T G(x) G^T(x) \nabla V^*(x) \end{aligned} \quad (9.3.21)$$

with $V^*(0) = 0$. Substituting (9.3.20) into (9.3.21), we can obtain the formulation of the modified HJB equation in terms of $\nabla V^*(x)$ as follows:

$$\begin{aligned} 0 &= Q(x) + (\nabla V^*(x))^T f(x) + h^T(\varphi(x)) h(\varphi(x)) \\ &\quad - \frac{1}{4} (\nabla V^*(x))^T g(x) R^{-1} g^T(x) \nabla V^*(x) \\ &\quad + \frac{1}{4} (\nabla V^*(x))^T G(x) G^T(x) \nabla V^*(x) \end{aligned} \quad (9.3.22)$$

with $V^*(0) = 0$.

Now, we give the following assumption, which is helpful to derive the optimal control with regard to system (9.3.2) and prove the stability of the closed-loop system.

Assumption 9.3.2 Consider system (9.3.2) with value function (9.3.16) and the optimal feedback control function (9.3.20). Let $L_s(x)$ be a continuously differentiable Lyapunov function candidate formed as a polynomial and satisfying

$$\dot{L}_s(x) = (\nabla L_s(x))^T \dot{x}_{(9.3.2)} = (\nabla L_s(x))^T (f(x) + g(x)u^*) < 0, \quad (9.3.23)$$

where $\nabla L_s(x) = \partial L_s(x)/\partial x$. Assume there exists a positive definite matrix $\Lambda(x)$ such that the following relation holds:

$$(\nabla L_s(x))^T (f(x) + g(x)u^*) = -(\nabla L_s(x))^T \Lambda(x) \nabla L_s(x).$$

Remark 9.3.3 This is a common assumption that has been used in the literature, for instance [7, 36, 60], to facilitate discussing the stability issue of closed-loop system. According to [7], we assume that the closed-loop dynamics with optimal control is bounded by a function of system state on the compact set of this section. Without loss of generality, we assume that

$$\|f(x) + g(x)u^*\| \leq \eta \|\nabla L_s(x)\|$$

with $\eta > 0$. Hence, we can further obtain

$$\|(\nabla L_s(x))^T (f(x) + g(x)u^*)\| \leq \eta \|\nabla L_s(x)\|^2.$$

Let λ_m and λ_M be the minimum and maximum eigenvalues of matrix $\Lambda(x)$, then

$$\lambda_m \|\nabla L_s(x)\|^2 \leq (\nabla L_s(x))^T \Lambda(x) \nabla L_s(x) \leq \lambda_M \|\nabla L_s(x)\|^2. \quad (9.3.24)$$

Therefore, by noticing (9.3.23) and (9.3.24), we can conclude that the Assumption 9.3.2 is reasonable. Specifically, in this section, $L_s(x)$ can be obtained by properly selecting a polynomial when implementing the ADP method.

The following theorem illustrates how to develop the optimal robust guaranteed cost control scheme for system (9.3.1).

Theorem 9.3.2 Consider system (9.3.1) with cost function (9.3.5). Suppose the modified HJB equation (9.3.22) has a continuously differentiable solution $V^*(x)$. Then, for any admissible control function u , the cost function (9.3.5) satisfies

$$\bar{J}(x_0, u) \leq \Phi(u),$$

where

$$\Phi(u) \triangleq V^*(x_0) + \int_0^\infty (u - u^*)^T R(u - u^*) d\tau.$$

Moreover, the optimal robust guaranteed cost of the controlled uncertain nonlinear system is given by

$$\Phi^* = \Phi(u^*) = V^*(x_0).$$

Accordingly, the optimal robust guaranteed cost control is given by $\bar{u}^* = u^*$.

Proof For any admissible control function $u(x)$, the cost function (9.3.5) can be written as the following form:

$$\bar{J}(x_0, u) = V^*(x_0) + \int_0^\infty \{U(x, u) + \dot{V}^*(x)\} d\tau. \quad (9.3.25)$$

Along the closed-loop trajectories of system (9.3.1) and according to (9.3.22), we find that

$$\begin{aligned} U(x, u) + \dot{V}_{(9.3.1)}^*(x) &= Q(x) + u^T R u + (\nabla V^*(x))^T (f(x) + g(x)u + \Delta f(x)) \\ &= u^T R u + (\nabla V^*(x))^T (g(x)u + \Delta f(x)) \\ &\quad - h^T(\varphi(x))h(\varphi(x)) - \frac{1}{4}(\nabla V^*(x))^T G(x)G^T(x)\nabla V^*(x) \\ &\quad + \frac{1}{4}(\nabla V^*(x))^T g(x)R^{-1}g^T(x)\nabla V^*(x). \end{aligned} \quad (9.3.26)$$

For the optimal cost function $V^*(x)$, in light of Lemma 9.3.1, the following inequality holds:

$$\begin{aligned} (\nabla V^*(x))^T \Delta f(x) &\leq h^T(\varphi(x))h(\varphi(x)) \\ &\quad + \frac{1}{4}(\nabla V^*(x))^T G(x)G^T(x)\nabla V^*(x). \end{aligned} \quad (9.3.27)$$

Substituting (9.3.27) into (9.3.26), we can further obtain

$$\begin{aligned} U(x, u) + \dot{V}^*(x) &\leq u^T R u + (\nabla V^*(x))^T g(x)u \\ &\quad + \frac{1}{4}(\nabla V^*(x))^T g(x)R^{-1}g^T(x)\nabla V^*(x). \end{aligned} \quad (9.3.28)$$

Considering the expression of the optimal control in (9.3.20), (9.3.28) is in fact

$$U(x, u) + \dot{V}^*(x) \leq (u - u^*)^T R (u - u^*). \quad (9.3.29)$$

Thus, combining (9.3.25) with (9.3.29), we can find

$$\bar{J}(x_0, u) \leq V^*(x_0) + \int_0^\infty (u - u^*)^T R (u - u^*) d\tau.$$

Clearly, the optimal robust guaranteed cost can be obtained when setting $u = u^*$, i.e., $\Phi(u^*) = V^*(x_0)$. Furthermore, we can derive that

$$\Phi^* = \inf_u \Phi(u) = V^*(x_0)$$

and

$$\bar{u}^* = \arg \inf_u \Phi(u) = u^*.$$

This completes the proof.

Remark 9.3.4 According to Theorem 9.3.2, the optimal robust guaranteed cost control of uncertain nonlinear system is transformed into the optimal control of nominal system, where the modified cost function is considered as the upper bound function. In other words, once the solution of the modified HJB equation (9.3.22) corresponding to nominal system (9.3.2) is derived, we can establish the optimal robust guaranteed cost control scheme of system (9.3.1).

9.3.2 Online Solution of Transformed Optimal Control Problem

In this section, inspired by the work of [5, 7, 41], an improved online technique without utilizing the iterative strategy and an initial stabilizing control is developed by constructing a single network, namely the critic network. Here, the ADP method is introduced to the framework of infinite horizon optimal robust guaranteed cost control of nonlinear systems with uncertainties.

Assume that the value function $V(x)$ is continuously differentiable. According to the universal approximation property of NNs, $V(x)$ can be reconstructed by a single-layer NN on a compact set \mathcal{Q} as

$$V(x) = W_c^\top \sigma_c(x) + \varepsilon_c(x), \quad (9.3.30)$$

where $W_c \in \mathbb{R}^l$ is the ideal weight, $\sigma_c(x) \in \mathbb{R}^l$ is the activation function, l is the number of neurons in the hidden layer, and $\varepsilon_c(x)$ is the unknown approximation error of NN. Then,

$$\nabla V(x) = (\nabla \sigma_c(x))^\top W_c + \nabla \varepsilon_c(x) \quad (9.3.31)$$

is also unknown, where

$$\nabla \sigma_c(x) = \frac{\partial \sigma_c(x)}{\partial x}, \quad \nabla \varepsilon_c(x) = \frac{\partial \varepsilon_c(x)}{\partial x}$$

are the gradients of the activation function and NN approximation error, respectively. Based on (9.3.31), the Lyapunov equation (9.3.8) takes the following form:

$$\begin{aligned}
0 = U(x, u) + & \left(W_c^\top \nabla \sigma_c(x) + (\nabla \varepsilon_c(x))^\top \right) F(x, u) \\
& + h^\top(\varphi(x)) h(\varphi(x)) + \frac{1}{4} \left(W_c^\top \nabla \sigma_c(x) + (\nabla \varepsilon_c(x))^\top \right) \\
& \times G(x) G^\top(x) \left((\nabla \sigma_c(x))^\top W_c + \nabla \varepsilon_c(x) \right).
\end{aligned}$$

Following the framework of [5, 7, 41], we assume that the weight vector W_c , the gradient $\nabla \sigma_c(x)$, and the approximation error $\varepsilon_c(x)$ and its derivative $\nabla \varepsilon_c(x)$ are all bounded on a compact set Ω .

Since the ideal weights are unknown, a critic NN can be built in terms of the estimated weights as

$$\hat{V}(x) = \hat{W}_c^\top \sigma_c(x) \quad (9.3.32)$$

to approximate the value function. Under the framework of ADP method, the selection of the activation function of the critic network is often a natural choice guided by engineering experience and intuition [1, 4]. Then,

$$\nabla \hat{V}(x) = (\nabla \sigma_c(x))^\top \hat{W}_c, \quad (9.3.33)$$

$$\text{where } \nabla \hat{V}(x) = \frac{\partial \hat{V}(x)}{\partial x}.$$

According to (9.3.20) and (9.3.31), we derive

$$u(x) = -\frac{1}{2} R^{-1} g^\top(x) \left((\nabla \sigma_c(x))^\top W_c + \nabla \varepsilon_c(x) \right), \quad (9.3.34)$$

which, in fact, represents the expression of optimal control $u^*(x)$ if the value function in (9.3.30) is considered as the optimal one $V^*(x)$. Besides, in light of (9.3.20) and (9.3.33), the approximate control function can be given as

$$\hat{u}(x) = -\frac{1}{2} R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top \hat{W}_c. \quad (9.3.35)$$

Applying (9.3.35) to system (9.3.2), the closed-loop system dynamics is expressed as

$$\dot{x} = f(x) - \frac{1}{2} g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top \hat{W}_c. \quad (9.3.36)$$

Recalling the definition of the Hamiltonian (9.3.18) and the modified HJB equation (9.3.19), we can easily obtain

$$H(x, u^*, \nabla V^*) = 0.$$

The NN expressions (9.3.31) and (9.3.34) imply that u^* and ∇V^* can be formulated based on the ideal weight of the critic network, i.e., W_c . As a result, the Hamiltonian becomes

$$H(x, W_c) = 0,$$

which specifically, can be written as

$$\begin{aligned} H(x, W_c) &= Q(x) + W_c^T \nabla \sigma_c(x) f(x) \\ &\quad - \frac{1}{4} W_c^T \nabla \sigma_c(x) g(x) R^{-1} g^T(x) (\nabla \sigma_c(x))^T W_c + h^T(\varphi(x)) h(\varphi(x)) \\ &\quad + \frac{1}{4} W_c^T \nabla \sigma_c(x) G(x) G^T(x) (\nabla \sigma_c(x))^T W_c + e_{cH} \\ &= 0, \end{aligned} \quad (9.3.37)$$

where

$$\begin{aligned} e_{cH} &= (\nabla \varepsilon_c(x))^T f(x) - \frac{1}{2} (\nabla \varepsilon_c(x))^T g(x) R^{-1} g^T(x) (\nabla \sigma_c(x))^T W_c \\ &\quad - \frac{1}{4} (\nabla \varepsilon_c(x))^T g(x) R^{-1} g^T(x) \nabla \varepsilon_c(x) \\ &\quad + \frac{1}{2} (\nabla \varepsilon_c(x))^T G(x) G^T(x) (\nabla \sigma_c(x))^T W_c \\ &\quad + \frac{1}{4} (\nabla \varepsilon_c(x))^T G(x) G^T(x) \nabla \varepsilon_c(x). \end{aligned} \quad (9.3.38)$$

In Eq. (9.3.38), e_{cH} denotes the residual error generated due to NN approximation.

Then, using the estimated weight vector, the approximate Hamiltonian can be derived as

$$\begin{aligned} \hat{H}(x, \hat{W}_c) &= Q(x) + \hat{W}_c^T \nabla \sigma_c(x) f(x) \\ &\quad - \frac{1}{4} \hat{W}_c^T \nabla \sigma_c(x) g(x) R^{-1} g^T(x) (\nabla \sigma_c(x))^T \hat{W}_c + h^T(\varphi(x)) h(\varphi(x)) \\ &\quad + \frac{1}{4} \hat{W}_c^T \nabla \sigma_c(x) G(x) G^T(x) (\nabla \sigma_c(x))^T \hat{W}_c. \end{aligned} \quad (9.3.39)$$

Let $e_c = \hat{H}(x, \hat{W}_c) - H(x, W_c)$. Considering (9.3.37), we have $e_c = \hat{H}(x, \hat{W}_c)$. Let the weight estimation error of the critic network be

$$\tilde{W}_c = W_c - \hat{W}_c. \quad (9.3.40)$$

Then, based on (9.3.37), (9.3.39), and (9.3.40), we can obtain the formulation of e_c in terms of \tilde{W}_c as follows:

$$\begin{aligned}
e_c &= \hat{H}(x, \hat{W}_c) - H(x, W_c) \\
&= -\tilde{W}_c^\top \nabla \sigma_c(x) f(x) - \frac{1}{4} \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c \\
&\quad + \frac{1}{2} \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top W_c \\
&\quad + \frac{1}{4} \tilde{W}_c^\top \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c \\
&\quad - \frac{1}{2} \tilde{W}_c^\top \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top W_c - e_{cH}.
\end{aligned} \tag{9.3.41}$$

For training the critic network, it is desired to design \hat{W}_c to minimize the objective function

$$E_c = \frac{1}{2} e_c^2. \tag{9.3.42}$$

Here, the weights of the critic network are tuned based on the standard steepest descent algorithm with an additional term introduced to ensure the boundedness of system state, i.e.,

$$\dot{\hat{W}}_c = -\alpha_c \left(\frac{\partial E_c}{\partial \hat{W}_c} \right) + \frac{1}{2} \alpha_s \Pi(x, \hat{u}) \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla L_s(x), \tag{9.3.43}$$

where $\alpha_c > 0$ is the learning rate of the critic network, $\alpha_s > 0$ is the learning rate of the additional term, and $L_s(x)$ is the Lyapunov function candidate given in Assumption 9.3.2. In (9.3.43), the $\Pi(x, \hat{u})$ is the additional stabilizing term defined as

$$\Pi(x, \hat{u}) = \begin{cases} 0, & \text{if } \dot{J}_s(x) = (\nabla L_s(x))^\top F(x, \hat{u}) < 0, \\ 1, & \text{else.} \end{cases}$$

The structural diagram of the implementation process using NN is displayed in Fig. 9.7.

Next, we will determine the dynamics of the weight estimation error \tilde{W}_c . According to (9.3.39), we have

$$\begin{aligned}
\frac{\partial e_c}{\partial \hat{W}_c} &= \nabla \sigma_c(x) f(x) - \frac{1}{2} \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top \hat{W}_c \\
&\quad + \frac{1}{2} \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top \hat{W}_c.
\end{aligned} \tag{9.3.44}$$

In light of (9.3.40), (9.3.42), and (9.3.43), the dynamics of the weight estimation error is

$$\dot{\tilde{W}}_c = \alpha_c e_c \left(\frac{\partial e_c}{\partial \hat{W}_c} \right) - \frac{1}{2} \alpha_s \Pi(x, \hat{u}) \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla L_s(x). \tag{9.3.45}$$

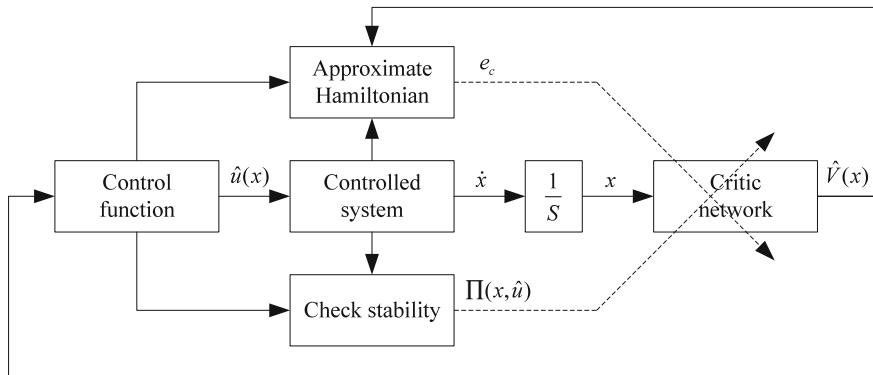


Fig. 9.7 Structural diagram of NN implementation (solid lines represent the signals and dashed lines represent the backpropagation paths)

Then, combining (9.3.40), (9.3.41), and (9.3.44), the error dynamics (9.3.45) becomes

$$\begin{aligned}
\dot{\tilde{W}}_c = & \alpha_c \left[-\tilde{W}_c^\top \nabla \sigma_c(x) f(x) - \frac{1}{4} \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c \right. \\
& + \frac{1}{2} \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top W_c \\
& + \frac{1}{4} \tilde{W}_c^\top \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c \\
& \left. - \frac{1}{2} \tilde{W}_c^\top \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top W_c - e_{cH} \right] \\
& \times \left[\nabla \sigma_c(x) f(x) - \frac{1}{2} \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top W_c \right. \\
& + \frac{1}{2} \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c \\
& + \frac{1}{2} \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top W_c \\
& \left. - \frac{1}{2} \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c \right] \\
& - \frac{1}{2} \alpha_s \Pi(x, \hat{u}) \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla L_s(x). \tag{9.3.46}
\end{aligned}$$

For continuous-time uncertain nonlinear systems (9.3.1) satisfying (9.3.3) and (9.3.4), we summarize the design procedure of optimal robust guaranteed cost control as follows.

Step 1: Select $G(x)$ and $\varphi(x)$, determine $h(\varphi(x))$, and conduct the problem transformation based on the bounded function $\Gamma(x)$ as in (9.3.13).

- Step 2: Choose the Lyapunov function candidate $L_s(x)$, construct a critic network as (9.3.32), and set its initial weights to zeros.
- Step 3: Solve the transformed optimal control problem via online solution of the modified HJB equation, using the expressions of approximate control function (9.3.35), approximate Hamiltonian (9.3.39), and weights update criterion (9.3.43).
- Step 4: Derive the optimal robust guaranteed cost and optimal robust guaranteed cost control of original uncertain nonlinear system based on the converged weights of critic network.

Remark 9.3.5 It is observed from (9.3.32) and (9.3.39), both the approximate value function and the approximate Hamiltonian become zero when $\|x\| = 0$. In this case, we can find that $\tilde{W}_c = 0$. Thus, when the system state converges to zero, the weights of the critic network are no longer updated. This can be viewed as a PE requirement of the NN inputs. In other words, the system state must be persistently exciting long enough in order to ensure the critic network to learn the optimal value function as accurately as possible. In this chapter, the PE condition is satisfied by adding a small exploratory signal to the control input. The condition can be removed once the weights of the critic network converge to their target values. Actually, it is for this reason that there always exists a trade-off between computational accuracy and time consumption for practical realization.

Next, the stability analysis of the NN-based feedback control system is presented using the Lyapunov theory.

9.3.3 Stability Analysis of Closed-Loop System

In this section, the error dynamics of the critic network and the closed-loop system based on the approximate optimal control are proved to be UUB.

Theorem 9.3.3 Consider the nonlinear system given by (9.3.2). Let the control input be provided by (9.3.35) and the weights of the critic network be tuned by (9.3.43). Then, the state x of the closed-loop system and the weight estimation error \tilde{W}_c of the critic network are UUB.

Proof We choose the following Lyapunov function candidate:

$$L(x) = \frac{1}{2\alpha_c} \tilde{W}_c^\top \tilde{W}_c + \frac{\alpha_s}{\alpha_c} L_s(x), \quad (9.3.47)$$

where $L_s(x)$ is presented in Assumption 9.3.2. The derivative of the Lyapunov function candidate (9.3.47) with respect to time along the solutions of (9.3.36) and (9.3.46) is

$$\dot{L}(x) = \frac{1}{\alpha_c} \tilde{W}_c^\top \dot{\tilde{W}}_c + \frac{\alpha_s}{\alpha_c} (\nabla L_s(x))^\top \dot{x}. \quad (9.3.48)$$

Substituting (9.3.36) and (9.3.46) into (9.3.48), we obtain

$$\begin{aligned} \dot{L}(x) = & \tilde{W}_c^\top \left[-\tilde{W}_c^\top \nabla \sigma_c(x) f(x) - \frac{1}{4} \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c \right. \\ & + \frac{1}{2} \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top W_c \\ & + \frac{1}{4} \tilde{W}_c^\top \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c \\ & \left. - \frac{1}{2} \tilde{W}_c^\top \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top W_c - e_{cH} \right] \\ & \times \left[\nabla \sigma_c(x) f(x) - \frac{1}{2} \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top W_c \right. \\ & + \frac{1}{2} \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c \\ & + \frac{1}{2} \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top W_c \\ & \left. - \frac{1}{2} \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c \right] \\ & - \frac{\alpha_s}{2\alpha_c} \Pi(x, \hat{u}) \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla L_s(x) + \frac{\alpha_s}{\alpha_c} (\nabla L_s(x))^\top \dot{x}. \end{aligned} \quad (9.3.49)$$

For simplicity, we denote

$$\begin{aligned} A &= \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top, \\ B &= \nabla \sigma_c(x) G(x) G^\top(x) (\nabla \sigma_c(x))^\top. \end{aligned}$$

Then, (9.3.49) becomes

$$\begin{aligned} \dot{L}(x) = & - \left[\tilde{W}_c^\top \nabla \sigma_c(x) f(x) + \frac{1}{4} \tilde{W}_c^\top A \tilde{W}_c - \frac{1}{2} \tilde{W}_c^\top A W_c - \frac{1}{4} \tilde{W}_c^\top B \tilde{W}_c + \frac{1}{2} \tilde{W}_c^\top B W_c + e_{cH} \right] \\ & \times \left[\tilde{W}_c^\top \nabla \sigma_c(x) f(x) + \frac{1}{2} \tilde{W}_c^\top A \tilde{W}_c - \frac{1}{2} \tilde{W}_c^\top A W_c - \frac{1}{2} \tilde{W}_c^\top B \tilde{W}_c + \frac{1}{2} \tilde{W}_c^\top B W_c \right] \\ & - \frac{\alpha_s}{2\alpha_c} \Pi(x, \hat{u}) \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla L_s(x) + \frac{\alpha_s}{\alpha_c} (\nabla L_s(x))^\top \dot{x}. \end{aligned}$$

Considering (9.3.36), we have

$$\begin{aligned}\dot{L}(x) = & - \left[\tilde{W}_c^\top \nabla \sigma_c(x) \dot{x} - \frac{1}{4} \tilde{W}_c^\top A \tilde{W}_c - \frac{1}{4} \tilde{W}_c^\top B \tilde{W}_c + \frac{1}{2} \tilde{W}_c^\top B W_c + e_{cH} \right] \\ & \times \left[\tilde{W}_c^\top \nabla \sigma_c(x) \dot{x} - \frac{1}{2} \tilde{W}_c^\top B \tilde{W}_c + \frac{1}{2} \tilde{W}_c^\top B W_c \right] \\ & - \frac{\alpha_s}{2\alpha_c} \Pi(x, \hat{u}) \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla L_s(x) \\ & + \frac{\alpha_s}{\alpha_c} (\nabla L_s(x))^\top \dot{x}.\end{aligned}$$

Using the formulations

$$\dot{x}^* = f(x) + g(x)u^*$$

and

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

we can obtain the relationship between \dot{x}^* and \dot{x} as

$$\dot{x} = \dot{x}^* + \frac{1}{2} g(x) R^{-1} g^\top(x) (\nabla \sigma_c(x))^\top \tilde{W}_c + \frac{1}{2} g(x) R^{-1} g^\top(x) \nabla \varepsilon_c(x).$$

Then, we have

$$\begin{aligned}\dot{L}(x) = & - \left[\tilde{W}_c^\top \nabla \sigma_c(x) \dot{x}^* + \frac{1}{4} \tilde{W}_c^\top A \tilde{W}_c \right. \\ & \left. + \frac{1}{2} \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla \varepsilon_c(x) - \frac{1}{4} \tilde{W}_c^\top B \tilde{W}_c + \frac{1}{2} \tilde{W}_c^\top B W_c + e_{cH} \right] \\ & \times \left[\tilde{W}_c^\top \nabla \sigma_c(x) \dot{x}^* + \frac{1}{2} \tilde{W}_c^\top A \tilde{W}_c \right. \\ & \left. + \frac{1}{2} \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla \varepsilon_c(x) - \frac{1}{2} \tilde{W}_c^\top B \tilde{W}_c + \frac{1}{2} \tilde{W}_c^\top B W_c \right] \\ & - \frac{\alpha_s}{2\alpha_c} \Pi(x, \hat{u}) \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla L_s(x) + \frac{\alpha_s}{\alpha_c} (\nabla L_s(x))^\top \dot{x}.\end{aligned}\tag{9.3.50}$$

As in the work of [7], we assume that $\lambda_{1m} > 0$ and $\lambda_{1M} > 0$ are the lower and upper bounds of the norm of matrix A . Similarly, assume that $\lambda_{2m} > 0$ and $\lambda_{2M} > 0$ are the lower and upper bounds of the norm of matrix B . Assume that $\|R^{-1}\| \leq R_M^{-1}$, $\|g(x)\| \leq g_M$, $\|\nabla \sigma(x)\| \leq \sigma_{dM}$, $\|BW_c\| \leq \lambda_4$, $\|\nabla \varepsilon_c(x)\| \leq \lambda_{10}$, and $\|e_{cH}\| \leq \lambda_{12}$, where R_M^{-1} , g_M , σ_{dM} , λ_4 , λ_{10} , and λ_{12} are positive constants. In addition, assume that $\|\nabla \sigma_c(x) \dot{x}^*\| \leq \lambda_3$, where λ_3 is a positive constant. Let $\lambda_5 = (\sqrt{6}/2)\lambda_{12}$, $\lambda_9 = g_M^2 R_M^{-1}$, and $\lambda_{11} = \sigma_{dM} g_M^2 R_M^{-1} \lambda_{10}$, then $\|g(x) R^{-1} g^\top(x)\| \leq \lambda_9$ and $\|\nabla \sigma(x) g(x) R^{-1} g^\top(x) \nabla \varepsilon_c(x)\| \leq \lambda_{11}$. Using the relations

$$ab = \frac{1}{2} \left[- \left(\phi_+ a - \frac{b}{\phi_+} \right)^2 + \phi_+^2 a^2 + \frac{b^2}{\phi_+^2} \right],$$

$$-ab = -\frac{1}{2} \left[\left(\phi_- a + \frac{b}{\phi_-} \right)^2 - \phi_-^2 a^2 - \frac{b^2}{\phi_-^2} \right],$$

we have

$$\begin{aligned} -\frac{3}{4} (\tilde{W}_c^\top \nabla \sigma_c(x) \dot{x}^*) (\tilde{W}_c^\top A \tilde{W}_c) \\ = -\frac{3}{8} \left[\left(\phi_1 \tilde{W}_c^\top \nabla \sigma_c(x) \dot{x}^* + \frac{\tilde{W}_c^\top A \tilde{W}_c}{\phi_1} \right)^2 - \phi_1^2 (\tilde{W}_c^\top \nabla \sigma_c(x) \dot{x}^*)^2 - \frac{(\tilde{W}_c^\top A \tilde{W}_c)^2}{\phi_1^2} \right] \\ \leq \frac{3}{8} \left[\phi_1^2 (\tilde{W}_c^\top \nabla \sigma_c(x) \dot{x}^*)^2 + \frac{(\tilde{W}_c^\top A \tilde{W}_c)^2}{\phi_1^2} \right] \\ \leq \frac{3}{8\phi_1^2} \lambda_{1M}^2 \|\tilde{W}_c\|^4 + \frac{3}{8} \phi_1^2 \lambda_3^2 \|\tilde{W}_c\|^2, \end{aligned}$$

where ϕ_+ , ϕ_- , and ϕ_1 are nonzero constants. Other terms in the expression of (9.3.50) can be handled the same way. Then, we can find that

$$\begin{aligned} \dot{L}(x) \leq & -\lambda_7 \|\tilde{W}_c\|^4 + \lambda_8 \|\tilde{W}_c\|^2 + \lambda_5^2 \\ & - \frac{\alpha_s}{2\alpha_c} \Pi(x, \hat{u}) \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla L_s(x) + \frac{\alpha_s}{\alpha_c} (\nabla L_s(x))^\top \dot{x}, \end{aligned} \quad (9.3.51)$$

where

$$\begin{aligned} \lambda_7 = & \frac{1}{8} \lambda_{1m}^2 + \frac{1}{8} \lambda_{2m}^2 - \frac{3}{8\phi_1^2} \lambda_{1M}^2 - \frac{3}{8\phi_2^2} \lambda_{2M}^2 \\ & - \frac{3}{16} \phi_3^2 \lambda_{1M}^2 - \frac{3}{16} \phi_4^2 \lambda_{1M}^2 - \frac{3}{16} \phi_5^2 \lambda_{11}^2 - \frac{3}{16} \phi_6^2 \lambda_{2M}^2, \\ \lambda_8 = & \frac{3}{8} \phi_1^2 \lambda_3^2 + \frac{3}{8} \phi_2^2 \lambda_3^2 + \frac{3}{16\phi_3^2} \lambda_{11}^2 + \frac{3}{16\phi_4^2} \lambda_4^2 \\ & + \frac{3}{16\phi_5^2} \lambda_{11}^2 + \frac{3}{16\phi_6^2} \lambda_4^2, \end{aligned}$$

and ϕ_i , $i = 1, 2, \dots, 6$, are nonzero design constants. Note that with proper choices of ϕ_i , $i = 1, 2, \dots, 6$, the relation $\lambda_7 > 0$ can be guaranteed.

We will consider next the cases of $\Pi(x, \hat{u}) = 0$ and $\Pi(x, \hat{u}) = 1$, respectively.

Case 1: $\Pi(x, \hat{u}) = 0$. Since $(\nabla L_s(x))^\top \dot{x} < 0$, we have $-(\nabla L_s(x))^\top \dot{x} > 0$. According to the density property of real numbers [38], there exists a positive constant λ_6 such that $0 < \lambda_6 \|\nabla L_s(x)\| \leq -(\nabla L_s(x))^\top \dot{x}$ holds for all $x \in \Omega$, i.e.,

$$(\nabla L_s(x))^\top \dot{x} \leq -\lambda_6 \|\nabla L_s(x)\|.$$

Hence, the inequality (9.3.51) becomes

$$\begin{aligned}\dot{L}(x) &\leq -\lambda_7 \|\tilde{W}_c\|^4 + \lambda_8 \|\tilde{W}_c\|^2 + \lambda_5^2 + \frac{\alpha_s}{\alpha_c} (\nabla L_s(x))^\top \dot{x} \\ &\leq -\lambda_7 \|\tilde{W}_c\|^4 + \lambda_8 \|\tilde{W}_c\|^2 + \lambda_5^2 - \frac{\alpha_s}{\alpha_c} \lambda_6 \|\nabla L_s(x)\|.\end{aligned}$$

Therefore, given the following inequality:

$$\|\tilde{W}_c\| \geq \sqrt{\frac{\lambda_8 + \sqrt{4\lambda_5^2\lambda_7 + \lambda_8^2}}{2\lambda_7}} \triangleq \mathcal{A}_1$$

or

$$\|\nabla L_s(x)\| \geq \frac{\alpha_c(4\lambda_5^2\lambda_7 + \lambda_8^2)}{4\alpha_s\lambda_6\lambda_7} \triangleq \mathcal{B}_1 \quad (9.3.52)$$

holds, we conclude $\dot{L}(x) < 0$. Note that (9.3.52) is derived from

$$\frac{\alpha_s}{\alpha_c} \lambda_6 \|\nabla L_s(x)\| \geq \max_{\|\tilde{W}_c\|^2} \{-\lambda_7 \|\tilde{W}_c\|^4 + \lambda_8 \|\tilde{W}_c\|^2 + \lambda_5^2\} = \lambda_5^2 + \frac{\lambda_8^2}{4\lambda_7}.$$

Case 2: $\Pi(x, \hat{u}) = 1$. Subtracting and adding

$$\frac{\alpha_s (\nabla L_s(x))^\top g(x) R^{-1} g^\top(x) \nabla \varepsilon_c(x)}{2\alpha_c}$$

to the right-hand side of (9.3.51) and taking Assumption 9.3.2 into consideration yield

$$\begin{aligned}\dot{L}(x) &\leq -\lambda_7 \|\tilde{W}_c\|^4 + \lambda_8 \|\tilde{W}_c\|^2 + \lambda_5^2 \\ &\quad - \frac{\alpha_s}{2\alpha_c} \tilde{W}_c^\top \nabla \sigma_c(x) g(x) R^{-1} g^\top(x) \nabla L_s(x) + \frac{\alpha_s}{\alpha_c} (\nabla L_s(x))^\top (f(x) + g(x)\hat{u}) \\ &= -\lambda_7 \|\tilde{W}_c\|^4 + \lambda_8 \|\tilde{W}_c\|^2 + \lambda_5^2 \\ &\quad + \frac{\alpha_s}{\alpha_c} (\nabla L_s(x))^\top (f(x) + g(x)u^*) + \frac{\alpha_s}{2\alpha_c} (\nabla L_s(x))^\top g(x) R^{-1} g^\top(x) \nabla \varepsilon_c(x) \\ &\leq -\lambda_7 \|\tilde{W}_c\|^4 + \lambda_8 \|\tilde{W}_c\|^2 + \lambda_5^2 \\ &\quad - \frac{\alpha_s}{\alpha_c} \lambda_m \|\nabla L_s(x)\|^2 + \frac{\alpha_s}{2\alpha_c} \lambda_9 \lambda_{10} \|\nabla L_s(x)\| \\ &= -\lambda_7 \|\tilde{W}_c\|^4 + \lambda_8 \|\tilde{W}_c\|^2 + \lambda_5^2 \\ &\quad - \frac{\alpha_s}{\alpha_c} \lambda_m \left[\left(\|\nabla L_s(x)\| - \frac{\lambda_9 \lambda_{10}}{4\lambda_m} \right)^2 - \frac{\lambda_9^2 \lambda_{10}^2}{16\lambda_m^2} \right].\end{aligned}$$

Therefore, given the following inequality:

$$\|\tilde{W}_c\| \geq \sqrt{\frac{\lambda_8}{2\lambda_7} + \sqrt{\frac{\lambda_5^2}{\lambda_7} + \frac{\lambda_8^2}{4\lambda_7^2} + \frac{\alpha_s \lambda_9^2 \lambda_{10}^2}{16\alpha_c \lambda_m \lambda_7}}} \triangleq \mathcal{A}_2$$

or

$$\|\nabla L_s(x)\| \geq \frac{\lambda_9 \lambda_{10}}{4\lambda_m} + \sqrt{\frac{\alpha_c(4\lambda_5^2 \lambda_7 + \lambda_8^2)}{4\alpha_s \lambda_m \lambda_7} + \frac{\lambda_9^2 \lambda_{10}^2}{16\lambda_m^2}} \triangleq \mathcal{B}_2$$

holds, we obtain $\dot{L}(x) < 0$.

To summarize, if the inequality $\|\tilde{W}_c\| > \max(\mathcal{A}_1, \mathcal{A}_2) = \mathcal{A}$ or $\|\nabla L_s(x)\| > \max(\mathcal{B}_1, \mathcal{B}_2) = \mathcal{B}$ holds, then $\dot{L}(x) < 0$. Considering the fact that $L_s(x)$ is chosen as a polynomial and in accordance with the standard Lyapunov's extension theorem [13, 15] (or the Lagrange stability result [30]), we can derive the conclusion that the state x and the weight estimation error \tilde{W}_c are UUB. This completes the proof.

Corollary 9.3.1 *The approximate control input \hat{u} in (9.3.35) converges to a neighborhood of optimal control input u^* with finite bound.*

Proof According to (9.3.34) and (9.3.35), we have

$$u^* - \hat{u} = -\frac{1}{2} R^{-1} g^T(x) (\nabla \sigma(x))^T \tilde{W}_c - \frac{1}{2} R^{-1} g^T(x) \nabla \varepsilon_c(x).$$

In light of Theorem 9.3.3, we have $\|\tilde{W}_c\| < \mathcal{A}$, where \mathcal{A} is defined in the proof above. Then, the terms $R^{-1} g^T(x) (\nabla \sigma(x))^T \tilde{W}_c$ and $R^{-1} g^T(x) \nabla \varepsilon_c(x)$ are both bounded. Thus, we can further determine

$$\|u^* - \hat{u}\| \leq \frac{1}{2} R_M^{-1} g_M \sigma_{dM} \mathcal{A} + \frac{1}{2} R_M^{-1} g_M \lambda_{10} \triangleq \varepsilon_u,$$

where λ_{10} is given in the proof of Theorem 9.3.3 and ε_u is the finite bound. This completes the proof.

9.3.4 Simulation Studies

In this section, two simulation examples are provided to demonstrate the effectiveness of the optimal robust guaranteed cost control strategy derived based on the online HJB solution. We first consider a continuous-time linear system and then a nonlinear system, both with system uncertainty.

Example 9.3.1 Consider the continuous-time linear system

$$\dot{x} = \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix}x + \begin{bmatrix} 1 \\ -3 \end{bmatrix}u + \Delta f(x),$$

where $x = [x_1, x_2]^\top$ and $\Delta f(x) = [px_1 \sin x_2, 0]^\top$ with $p \in [-0.5, 0.5]$. According to the form of system uncertainty, we choose $G(x) = [1, 0]^\top$ and $\varphi(x) = x$. Then, $d(\varphi(x)) = px_1 \sin x_2$. Besides, we select $h(\varphi(x)) = 0.5x_1 \sin x_2$.

In this example, we first choose $Q(x) = x^\top x$, $R = I$, where I is an identity matrix with suitable dimension. In order to solve the transformed optimal control problem, a critic network is constructed to approximate the value function as

$$\hat{V}(x) = \hat{W}_{c1}x_1^2 + \hat{W}_{c2}x_1x_2 + \hat{W}_{c3}x_2^2.$$

Let the initial state of the controlled plant be $x_0 = [1, -1]^\top$. Select the Lyapunov function candidate of the weights tuning criterion as $L_s(x) = (1/2)x^\top x$. Let the learning rate of the critic network and the additional term be $\alpha_c = 0.8$ and $\alpha_s = 0.5$, respectively. During the NN implementation process, the exploratory signal $\mathcal{N}(t)$ given in (9.2.35) is added to the control input to satisfy the PE condition. It is introduced into the control input and thus affects the system state. After a learning session, the weights of the critic network converge to $[0.3461, -0.1330, 0.1338]^\top$ as shown in Fig. 9.8. Here, it is important to note that the initial weights of the critic network are all set to zeros, which implies that no initial stabilizing control is needed

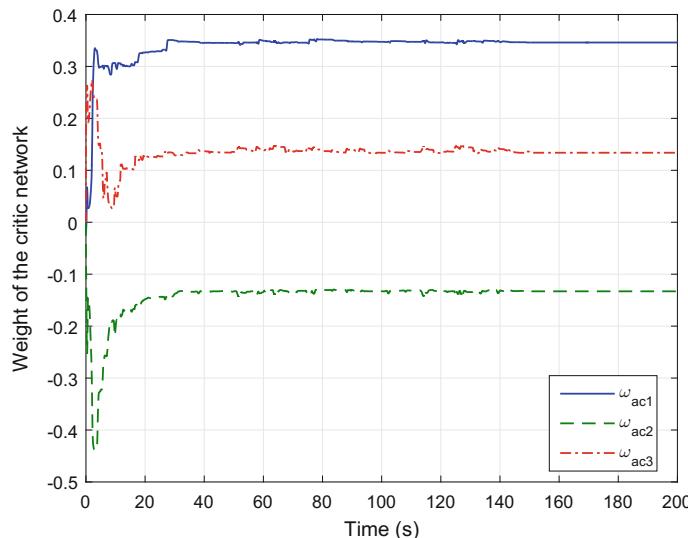


Fig. 9.8 Convergence of weight vector of the critic network (ω_{ac1} , ω_{ac2} , and ω_{ac3} represent \hat{W}_{c1} , \hat{W}_{c2} , and \hat{W}_{c3} , respectively)

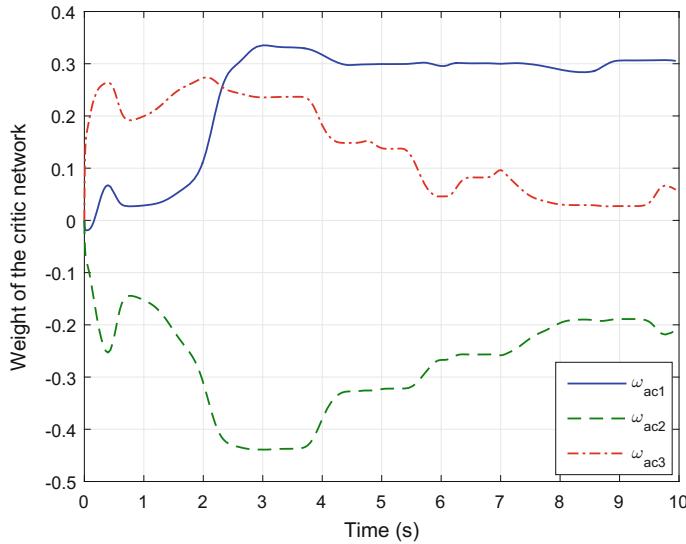


Fig. 9.9 Updating process of weight vector during the first 10 s (ω_{ac1} , ω_{ac2} , and ω_{ac3} represent \hat{W}_{c1} , \hat{W}_{c2} , and \hat{W}_{c3} , respectively)

for implementing the control strategy. This can be verified by observing Fig. 9.9, which displays the updating process of weight vector during the first 10 s.

Based on the converged weight vector, the optimal robust guaranteed cost of the controlled system is $\Phi(u^*) = V^*(x_0) = 0.6129$. Next, the scalar parameter $p = 0.5$ is chosen for evaluating the control performance. Under the action of the obtained control function, the system trajectory during the first 20 s is presented in Fig. 9.10, which shows good performance of the control approach.

Next, we set $Q(x) = 8x^T x$, $R = 5I$, and conduct the NN implementation again by increasing the learning rates of the critic network and the additional term properly. In this case, the weights of the critic network converge to $[5.4209, -3.5088, 1.2605]^T$, which is depicted in Fig. 9.11. Similarly, the system trajectory during the first 20 s when choosing $p = 0.5$ is displayed in Fig. 9.12. These simulation results show that the parameters $Q(x)$ and R play an important role in the design process. In addition, the power of the present control technique is demonstrated again.

Example 9.3.2 Consider the following continuous-time nonlinear system:

$$\dot{x} = \begin{bmatrix} -x_1 + x_2 \\ 0.1x_1 - x_2 - x_1x_3 \\ x_1x_2 - x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} u + \Delta f(x), \quad (9.3.53)$$

where $x = [x_1, x_2, x_3]^T$,

$$\Delta f(x) = [0, 0, px_1 \sin x_2 \cos x_3]^T,$$

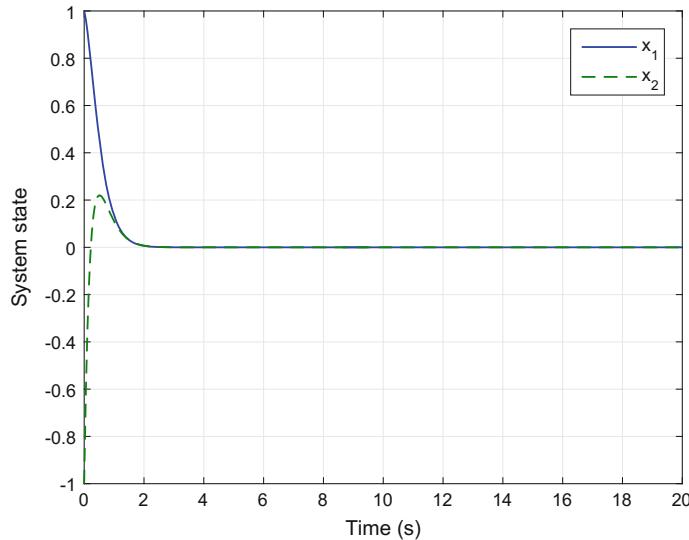


Fig. 9.10 The system state ($p = 0.5$)

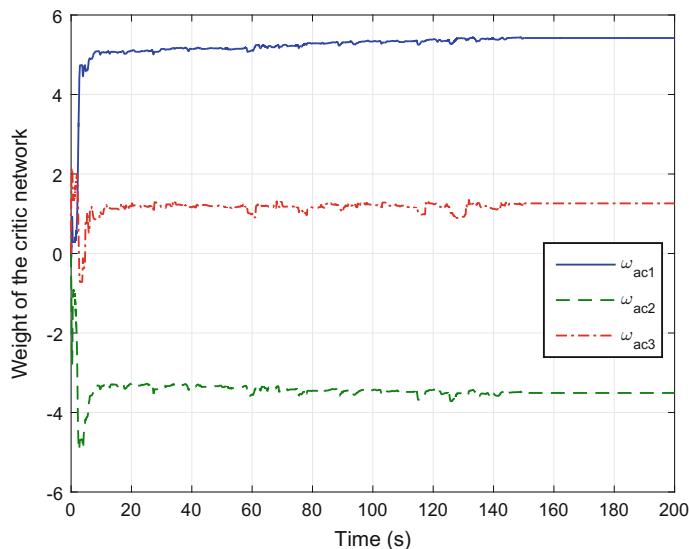


Fig. 9.11 Convergence of weight vector of the critic network (ω_{ac1} , ω_{ac2} , and ω_{ac3} represent \hat{W}_{c1} , \hat{W}_{c2} , and \hat{W}_{c3} , respectively)

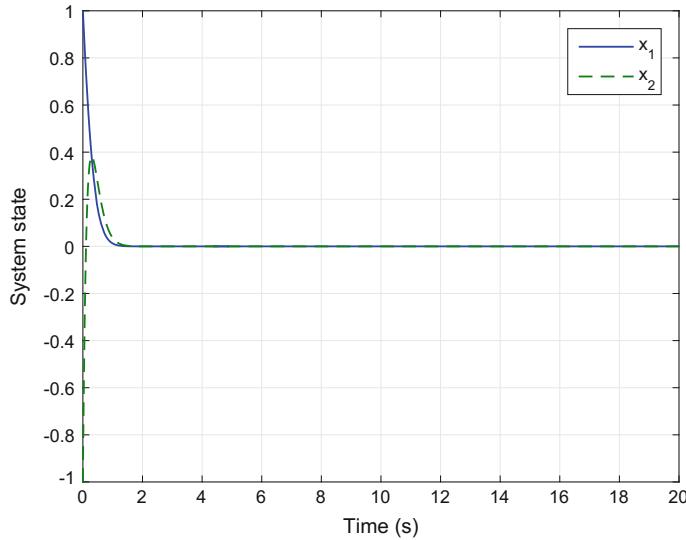


Fig. 9.12 The system state ($p = 0.5$)

and $p \in [-1, 1]$. Similarly, if we choose $G(x) = [0, 0, 1]^\top$ and $\varphi(x) = x$ based on the form of system uncertainty, then $d(\varphi(x)) = px_1 \sin x_2 \cos x_3$. Clearly, we can select $h(\varphi(x)) = x_1 \sin x_2 \cos x_3$.

In this example, $Q(x)$ and R are chosen the same as the first case of Example 9.3.1. However, the critic network is constructed by using the following equation:

$$\begin{aligned}\hat{V}(x) = & \hat{W}_{c1}x_1^2 + \hat{W}_{c2}x_2^2 + \hat{W}_{c3}x_3^2 + \hat{W}_{c4}x_1x_2 + \hat{W}_{c5}x_1x_3 + \hat{W}_{c6}x_2x_3 + \hat{W}_{c7}x_1^4 \\ & + \hat{W}_{c8}x_2^4 + \hat{W}_{c9}x_3^4 + \hat{W}_{c10}x_1^2x_2^2 + \hat{W}_{c11}x_1^2x_3^2 + \hat{W}_{c12}x_2^2x_3^2 + \hat{W}_{c13}x_1^2x_2x_3 \\ & + \hat{W}_{c14}x_1x_2^2x_3 + \hat{W}_{c15}x_1x_2x_3^2 + \hat{W}_{c16}x_1^3x_2 + \hat{W}_{c17}x_1^3x_3 + \hat{W}_{c18}x_1x_2^3 \\ & + \hat{W}_{c19}x_2^3x_3 + \hat{W}_{c20}x_1x_3^3 + \hat{W}_{c21}x_2x_3^3.\end{aligned}$$

Here, let the initial state of the controlled system be $x_0 = [1, -1, 0.5]^\top$. Besides, let the learning rate of the critic network and the additional term be $\alpha_c = 0.3$ and $\alpha_s = 0.5$, respectively. Same as earlier, a small exploratory signal $\mathcal{N}(t)$ (cf. (9.2.35)) is added to satisfy the PE condition during the NN implementation process. Besides, all the elements of the weight vector of critic network are initialized to zero. After a sufficient learning session, the weights of the critic network converge to $[0.4759, 0.5663, 0.1552, 0.4214, 0.0911, 0.0375, 0.0886, -0.0099, 0.0986, 0.1539, 0.0780, -0.0192, -0.1335, -0.0052, -0.0639, -0.1583, 0.0456, 0.0576, -0.0535, 0.0885, -0.0227]^\top$.

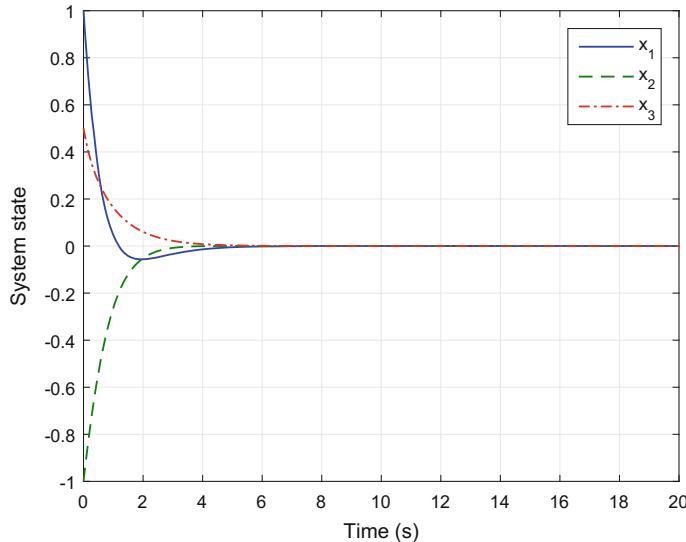


Fig. 9.13 The system state ($p = -1$)

Similarly, the optimal robust guaranteed cost of the nonlinear system is $\Phi(u^*) = V^*(x_0) = 1.1841$. In this example, the scalar parameter $p = -1$ is chosen for evaluating the robust control performance. The system trajectory is depicted in Fig. 9.13 when applying the obtained control to system (9.3.53) for 20 s. These simulation results verify the effectiveness of the present control approach.

9.4 Conclusions

In this chapter, a novel strategy is developed to solve the robust control problem of a class of uncertain nonlinear systems. The robust control problem is transformed into an optimal control problem with appropriate cost function. The online PI algorithm is presented to solve the HJB equation by constructing a critic network. Then, a strategy is developed to derive the optimal guaranteed cost control of uncertain nonlinear systems. This is accomplished by properly modifying the cost function to account for system uncertainty, so that the solution of the transformed optimal control problem also solves the optimal robust guaranteed cost control problem of the original system. A critic network is constructed to solve the modified HJB equation online. Several simulation examples are presented to reinforce the theoretical results.

References

1. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791
2. Adhyaru DM, Kar IN, Gopal M (2009) Fixed final time optimal control approach for bounded robust controller design using Hamilton-Jacobi-Bellman solution. *IET Control Theory Appl* 3(9):1183–1195
3. Adhyaru DM, Kar IN, Gopal M (2011) Bounded robust control of nonlinear systems using neural network-based HJB solution. *Neural Comput Appl* 20(1):91–103
4. Beard RW, Saridis GN, Wen JT (1997) Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation. *Automatica* 33(12):2159–2177
5. Bhasin S, Kamalapurkar R, Johnson M, Vamvoudakis KG, Lewis FL, Dixon WE (2013) A novel actor-critic-identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica* 49(1):82–92
6. Chang SSL, Peng TKC (1972) Adaptive guaranteed cost control of systems with uncertain parameters. *IEEE Trans Autom Control* 17(4):474–483
7. Dierks T, Jagannathan S (2010) Optimal control of affine nonlinear continuous-time systems. In: *Proceedings of the American Control Conference*, pp 1568–1573
8. Dierks T, Jagannathan S (2012) Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update. *IEEE Trans Neural Netw Learn Syst* 23(7):1118–1129
9. Haddad WM, Chellaboina V (2008) *Nonlinear dynamical systems and control: a Lyapunov-based approach*. Princeton University Press, Princeton
10. Haddad WM, Chellaboina V, Fausz JL (1998) Robust nonlinear feedback control for uncertain linear systems with nonquadratic performance criteria. *Syst Control Lett* 33(5):327–338
11. Haddad WM, Chellaboina V, Fausz JL, Leonessa A (2000) Optimal non-linear robust control for nonlinear uncertain systems. *Int J Control* 73(4):329–342
12. Heydari A, Balakrishnan SN (2013) Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics. *IEEE Trans Neural Netw Learn Syst* 24(1):145–157
13. LaSalle JP, Lefschetz S (1967) *Stability by Liapunov's direct method with applications*. Academic Press, New York
14. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst Mag* 9(3):32–50
15. Lewis FL, Jagannathan S, Yesildirek A (1999) *Neural network control of robot manipulators and nonlinear systems*. Taylor & Francis, London
16. Lewis FL, Vrabie D, Vamvoudakis KG (2012) Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst Mag* 32(6):76–105
17. Li H, Liu D (2012) Optimal control for discrete-time affine nonlinear systems using general value iteration. *IET Control Theory Appl* 6(18):2725–2736
18. Liang J, Venayagamoorthy GK, Harley RG (2012) Wide-area measurement based dynamic stochastic optimal power flow control for smart grids with high variability and uncertainty. *IEEE Trans Smart Grid* 3(1):59–69
19. Lin F, Brand RD, Sun J (1992) Robust control of nonlinear systems: compensating for uncertainty. *Int J Control* 56(6):1453–1459
20. Liu D, Wang D, Zhao D, Wei Q, Jin N (2012) Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming. *IEEE Trans Autom Sci Eng* 9(3):628–634
21. Liu D, Yang X, Li H (2013) Adaptive optimal control for a class of continuous-time affine nonlinear systems with unknown internal dynamics. *Neural Comput Appl* 23:1843–1850
22. Liu D, Wang D, Yang X (2013) An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs. *Inf Sci* 220:331–342

23. Liu D, Huang Y, Wang D, Wei Q (2013) Neural-network-observer-based optimal control for unknown nonlinear systems using adaptive dynamic programming. *Int J Control* 86(9):1554–1566
24. Liu D, Li H, Wang D (2013) Data-based self-learning optimal control: Research progress and prospects. *Acta Autom Sinica* 39(11):1858–1870
25. Liu D, Li H, Wang D (2013) Neural-network-based zero-sum game for discrete-time nonlinear systems via iterative adaptive dynamic programming algorithm. *Neurocomputing* 110:92–100
26. Liu D, Li H, Wang D (2014) Online synchronous approximate optimal learning algorithm for multi-player non-zero-sum games with unknown dynamics. *IEEE Trans Syst Man Cybern Syst* 44(8):1015–1027
27. Liu D, Wang D, Li H (2014) Decentralized stabilization for a class of continuous-time nonlinear interconnected systems using online learning optimal control approach. *IEEE Trans Neural Netw Learn Syst* 25(2):418–428
28. Liu D, Wang D, Wang FY, Li H, Yang X (2014) Neural-network-based online HJB solution for optimal robust guaranteed cost control of continuous-time uncertain nonlinear systems. *IEEE Trans Cybern* 44(12):2834–2847
29. Mehraeen S, Jagannathan S (2011) Decentralized optimal control of a class of interconnected nonlinear discrete-time systems by using online Hamilton-Jacobi-Bellman formulation. *IEEE Trans Neural Netw* 22(11):1757–1769
30. Michel AN, Hou L, Liu D (2015) Stability of dynamical systems: On the role of monotonic and non-monotonic Lyapunov functions. Birkhäuser, Boston
31. Modares H, Lewis FL, Naghibi-Sistani MB (2013) Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks. *IEEE Trans Neural Netw Learn Syst* 24(10):1513–1525
32. Murray JJ, Cox CJ, Lendaris GG, Saeks R (2002) Adaptive dynamic programming. *IEEE Trans Syst Man Cybern Part C Appl Rev* 32(2):140–153
33. Nevistic V, Primbs JA (1996) Constrained nonlinear optimal control: a converse HJB approach. Technical Memorandum No. CIT-CDS, California Institute of Technology, Pasadena, CA, pp 96–021
34. Ni Z, He H, Wen J (2013) Adaptive learning in tracking control based on the dual critic network design. *IEEE Trans Neural Netw Learn Syst* 24(6):913–928
35. Ni Z, He H, Wen J, Xu X (2013) Goal representation heuristic dynamic programming on maze navigation. *IEEE Trans Neural Netw Learn Syst* 24(12):2038–2050
36. Nodland D, Zargarzadeh H, Jagannathan S (2013) Neural network-based optimal adaptive output feedback control of a helicopter UAV. *IEEE Trans Neural Netw Learn Syst* 24(7):1061–1073
37. Prokhorov DV, Wunsch DC (1997) Adaptive critic designs. *IEEE Trans Neural Netw* 8(5):997–1007
38. Rudin W (1976) Principles of mathematical analysis. McGraw-Hill, New York
39. Si J, Wang YT (2001) On-line learning control by association and reinforcement. *IEEE Trans Neural Netw* 12(2):264–276
40. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
41. Vamvoudakis KG, Lewis FL (2010) Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* 46(5):878–888
42. Vrabie D, Lewis FL (2009) Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Netw* 22(3):237–246
43. Wang D, Liu D (2013) Neuro-optimal control for a class of unknown nonlinear dynamic systems using SN-DHP technique. *Neurocomputing* 121:218–225
44. Wang D, Liu D, Wei Q (2012) Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach. *Neurocomputing* 78(1):14–22
45. Wang D, Liu D, Wei Q, Zhao D, Jin N (2012) Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. *Automatica* 48(8):1825–1832

46. Wang D, Liu D, Zhao D, Huang Y, Zhang D (2013) A neural-network-based iterative GDHP approach for solving a class of nonlinear optimal control problems with control constraints. *Neural Comput Appl* 22(2):219–227
47. Wang D, Liu D, Li H (2014) Policy iteration algorithm for online design of robust control for a class of continuous-time nonlinear systems. *IEEE Trans Autom Sci Eng* 11(2):627–632
48. Wang D, Liu D, Li H, Ma H (2014) Neural-network-based robust optimal control design for a class of uncertain nonlinear systems via adaptive dynamic programming. *Inf Sci* 282:167–179
49. Wang FY, Zhang H, Liu D (2009) Adaptive dynamic programming: an introduction. *IEEE Comput Intell Mag* 4(2):39–47
50. Wang FY, Jin N, Liu D, Wei Q (2011) Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε -error bound. *IEEE Trans Neural Netw* 22(1):24–36
51. Werbos PJ (1977) Advanced forecasting methods for global crisis warning and models of intelligence. *General Syst Yearb* 22:25–38
52. Werbos PJ (1992) Approximate dynamic programming for real-time control and neural modeling. In: White DA, Sofge DA (eds) *Handbook of intelligent control: neural, fuzzy, and adaptive approaches* (Chapter 13). Van Nostrand Reinhold, New York
53. Werbos PJ (2008) ADP: The key direction for future research in intelligent control and understanding brain intelligence. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):898–900
54. Werbos PJ (2009) Intelligence in the brain: a theory of how it works and how to build it. *Neural Netw* 22(3):200–212
55. Wu HN, Luo B (2012) Neural network based online simultaneous policy update algorithm for solving the HJI equation in nonlinear H_∞ control. *IEEE Trans Neural Netw Learn Syst* 23(12):1884–1895
56. Xu H, Jagannathan S, Lewis FL (2012) Stochastic optimal control of unknown linear networked control system in the presence of random delays and packet losses. *Automatica* 48(6):1017–1030
57. Xu X, Lian C, Zuo L, He H (2014) Kernel-based approximate dynamic programming for real-time online learning control: an experimental study. *IEEE Trans Control Syst Technol* 22(1):146–156
58. Yu L, Chu J (1999) An LMI approach to guaranteed cost control of linear uncertain time-delay systems. *Automatica* 35(6):1155–1159
59. Yu L, Han QL, Sun MX (2005) Optimal guaranteed cost control of linear uncertain systems with input constraints. *Int J Control Autom Syst* 3(3):397–402
60. Zhang H, Cui L, Luo Y (2013) Near-optimal control for nonzero-sum differential games of continuous-time nonlinear systems using single-network ADP. *IEEE Trans Cybern* 43(1):206–216
61. Zhang H, Zhang X, Luo Y, Yang J (2013) An overview of research on adaptive dynamic programming. *Acta Autom Sinica* 39(4):303–311

Chapter 10

Decentralized Control of Continuous-Time Interconnected Nonlinear Systems

10.1 Introduction

Various complex systems in social and engineering areas, such as ecosystems, transportation systems, and power systems, are considered as large-scale systems. Generally speaking, a large-scale system is comprised of several subsystems with obvious interconnections, which leads to the increasing difficulty of analysis and synthesis when using classical centralized control techniques. Therefore, it is necessary to partition the design issue of the overall system into manageable subproblems, as Bakule pointed out in [2]. Then, the overall plant is no longer controlled by a single controller but by an array of independent controllers which together represent a decentralized controller. Consequently, the decentralized control has been a control of choice for large-scale systems since it is computationally efficient to formulate control laws that use only locally available subsystem states or outputs [20]. Actually, considerable attention has been paid to the decentralized stabilization of large-scale systems during the last several decades. A decentralized strategy consists of some noninteracting local controllers corresponding to the isolated subsystems, not the overall system. Thus, in many situations, the design of isolated subsystems is a matter of great significance. In [18], it was shown that the decentralized control of the interconnected system is related to the optimal control of the isolated subsystems. Therefore, the optimal control method can be employed to facilitate the design process of the decentralized control strategy. However, in that work, the cost functions of the isolated subsystems were not chosen as the general forms, not to mention that the detailed procedure was not given. For this reason, in this chapter, we will investigate the decentralized stabilization problem using neural network (NN)-based optimal learning control approach.

The optimal control of nonlinear system often leads to solving the Hamilton–Jacobi–Bellman (HJB) equation instead of the Riccati equation as in the case of linear systems. Adaptive dynamic programming (ADP) was first proposed by Werbos as an alternative method to solve the optimal control problems

forward-in-time [27, 28]. In recent years, great efforts have been made to ADP-related research in theory and applications [1, 5–8, 10–12, 15, 16, 23–25]. In light of [9, 10, 29], the ADP technique is closely related to reinforcement learning when engaging in the research of feedback control. Policy iteration (PI) is a fundamental algorithm for reinforcement learning-based ADP in optimal control. Vrabie and Lewis [24] derived an integral reinforcement learning method to obtain direct adaptive optimal control for nonlinear input-affine continuous-time systems with partially unknown dynamics. Jiang and Jiang [5] presented a novel PI approach for continuous-time linear systems with complete unknown dynamics. Lee et al. [8] presented an integral Q-learning algorithm for continuous-time systems without the exact knowledge of system dynamics.

In this chapter, we first employ the online PI algorithm to tackle the decentralized control problem for a class of large-scale interconnected nonlinear systems [13]. The decentralized control strategy can be established by adding appropriate feedback gains to the local optimal control laws. The online PI algorithm is developed to solve the HJB equations related to the optimal control problem by constructing and training some critic neural networks. The uniform ultimate boundedness (UUB) of the dynamics of the NN weight estimation errors is analyzed by using the Lyapunov approach. Since it is difficult to obtain the exact knowledge of the system dynamics for large-scale nonlinear systems, such as chemical engineering processes, transportation systems, and power systems, we relax the assumptions of exact knowledge of the system dynamics required in the optimal controller design presented in [13]. We use an online model-free integral PI to solve the decentralized control of a class of continuous-time large-scale interconnected nonlinear systems [14]. The optimal control problems for the isolated subsystems with unknown dynamics are related to the development of decentralized control laws. To implement this algorithm, a critic NN and an action NN are used to approximate the value function and control law of each isolated subsystem, respectively.

10.2 Decentralized Control of Interconnected Nonlinear Systems

In this section, we study the decentralized control of interconnected nonlinear systems based on ADP technique [13]. Consider a class of continuous-time large-scale nonlinear systems composed of N interconnected subsystems described by

$$\dot{x}_i(t) = f_i(x_i(t)) + g_i(x_i(t))(\bar{u}_i(t) + \tilde{Z}_i(x(t))), \quad i = 1, \dots, N, \quad (10.2.1)$$

where $x_i(t) \in \mathbb{R}^{n_i}$ and $\bar{u}_i(t) \in \mathbb{R}^{m_i}$ are the state vector and the control vector of the i th subsystem, respectively. In large-scale system (10.2.1), $x = [x_1^\top, x_2^\top, \dots, x_N^\top]^\top \in \mathbb{R}^n$ denotes the overall state, where $n = \sum_{i=1}^N n_i$. Correspondingly, x_1, x_2, \dots, x_N are called local states while $\bar{u}_1, \bar{u}_2, \dots, \bar{u}_N$ are local controls. Note that for subsystem

$i, f_i(x_i)$, $g_i(x_i)$, and $g_i(x_i)\bar{Z}_i(x)$ represent the nonlinear internal dynamics, the input gain matrix, and the interconnected term, respectively.

Let $x_i(0) = x_{i0}$ be the initial state of the i th subsystem, $i = 1, \dots, N$. Additionally, we let the following assumptions hold throughout this chapter.

Assumption 10.2.1 The state vector $x_i = 0$ is the equilibrium of the i th subsystem, $i = 1, \dots, N$.

Assumption 10.2.2 The functions $f_i(\cdot)$ and $g_i(\cdot)$ are differentiable in their arguments with $f_i(0) = 0$, where $i = 1, \dots, N$.

Assumption 10.2.3 The feedback control vector $\bar{u}_i(x_i) = 0$ when $x_i = 0$, where $i = 1, \dots, N$.

Let $R_i \in \mathbb{R}^{m_i \times m_i}$, $i = 1, \dots, N$, be symmetric positive-definite matrices. Then, we denote $Z_i(x) = R_i^{1/2}\bar{Z}_i(x)$, where $Z_i(x) \in \mathbb{R}^{m_i}$, $i = 1, \dots, N$, are bounded as follows:

$$\|Z_i(x)\| \leq \sum_{j=1}^N \rho_{ij} h_{ij}(x_j), \quad i = 1, \dots, N. \quad (10.2.2)$$

In (10.2.2), ρ_{ij} are nonnegative constant parameters and $h_{ij}(x_j)$ are positive-semidefinite functions with $i, j = 1, \dots, N$.

If we define $h_i(x_i) = \max\{h_{1i}(x_i), h_{2i}(x_i), \dots, h_{Ni}(x_i)\}$, $i = 1, \dots, N$, then (10.2.2) can be formulated as

$$\|Z_i(x)\| \leq \sum_{j=1}^N \lambda_{ij} h_j(x_j), \quad i = 1, \dots, N, \quad (10.2.3)$$

where

$$\lambda_{ij} \geq \rho_{ij} \frac{h_{ij}(x_j)}{h_j(x_j)}, \quad i, j = 1, \dots, N,$$

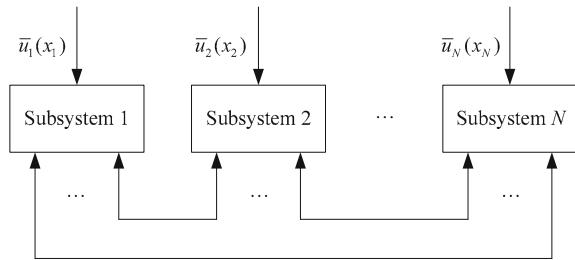
are nonnegative constant parameters.

When dealing with the decentralized control problem, we aim at finding N control laws $\bar{u}_1(x_1), \bar{u}_2(x_2), \dots, \bar{u}_N(x_N)$ to stabilize the large-scale system (10.2.1). It is important to note that in the control vector $(\bar{u}_1(x_1), \bar{u}_2(x_2), \dots, \bar{u}_N(x_N))$, $\bar{u}_i(x_i)$ is only a function of the corresponding local state, namely x_i , where $i = 1, \dots, N$. The schematic diagram of the decentralized control problem is displayed in Fig. 10.1.

10.2.1 Decentralized Stabilization via Optimal Control Approach

In this section, we investigate the methodology for decentralized controller design. The optimal control of the isolated subsystems is described under the framework of

Fig. 10.1 The schematic diagram of the decentralized control problem of the interconnected system



HJB equations. Then, the decentralized control strategy can be constructed based on the optimal control laws.

Consider the N isolated subsystems corresponding to (10.2.1) given by

$$\dot{x}_i(t) = f_i(x_i(t)) + g_i(x_i(t))u_i(x_i(t)), \quad i = 1, \dots, N. \quad (10.2.4)$$

For the i th isolated subsystem, we further assume that $f_i + g_i u_i$ is Lipschitz continuous on a set Ω_i in \mathbb{R}^{n_i} containing the origin, and the subsystem is controllable in the sense that there exists a continuous control policy on Ω_i that asymptotically stabilizes the subsystem.

In this section, in order to deal with the infinite-horizon optimal control problem, we need to find the control laws $u_i(x_i)$, $i = 1, \dots, N$, which minimize the cost functions

$$J_i(x_{i0}) = \int_0^\infty \{Q_i^2(x_i(\tau)) + u_i^\top(x_i(\tau))R_iu_i(x_i(\tau))\}d\tau, \quad i = 1, \dots, N, \quad (10.2.5)$$

where $Q_i(x_i)$, $i = 1, \dots, N$, are positive-definite functions satisfying

$$h_i(x_i) \leq Q_i(x_i), \quad i = 1, \dots, N. \quad (10.2.6)$$

Based on the theory of optimal control, the designed feedback controls must not only stabilize the subsystems on Ω_i , $i = 1, \dots, N$, but also guarantee that the cost functions (10.2.5) are finite. In other words, the control laws must be admissible. We use $\mathcal{A}_i(\Omega_i)$ to denote the set of admissible controls of subsystem i on Ω_i .

For any admissible control laws $\mu_i \in \mathcal{A}_i(\Omega_i)$, $i = 1, \dots, N$, if the associated value functions

$$V_i(x_i) = \int_t^\infty \{Q_i^2(x_i(\tau)) + \mu_i^\top(x_i(\tau))R_i\mu_i(x_i(\tau))\}d\tau, \quad i = 1, \dots, N, \quad (10.2.7)$$

are continuously differentiable, then the infinitesimal versions of (10.2.7) are the so-called nonlinear Lyapunov equations

$$0 = Q_i^2(x_i) + \mu_i^\top(x_i)R_i\mu_i(x_i) + \nabla V_i^\top(x_i)(f_i(x_i) + g_i(x_i)\mu_i(x_i)), \\ i = 1, \dots, N, \quad (10.2.8)$$

with $V_i(0) = 0$, $i = 1, \dots, N$. In (10.2.8), the terms $\nabla V_i(x_i)$, $i = 1, \dots, N$, denote the partial derivatives of the value functions $V_i(x_i)$ with respect to local states x_i , i.e., $\nabla V_i(x_i) = \partial V_i(x_i) / \partial x_i$, where $i = 1, \dots, N$.

Define the Hamiltonians of the N isolated subsystems as

$$H_i(x_i, \mu_i, \nabla V_i(x_i)) = Q_i^2(x_i) + \mu_i^\top(x_i)R_i\mu_i(x_i) + \nabla V_i^\top(x_i)(f_i(x_i) + g_i(x_i)\mu_i(x_i)),$$

where $i = 1, \dots, N$.

The optimal cost functions of the N isolated subsystems can be formulated as

$$J_i^*(x_i) = \min_{\mu_i \in \mathcal{A}_i(\Omega_i)} \int_t^\infty \{Q_i^2(x_i(\tau)) + \mu_i^\top(x_i(\tau))R_i\mu_i(x_i(\tau))\} d\tau, \quad i = 1, \dots, N. \quad (10.2.9)$$

In view of the theory of optimal control, the optimal cost functions $J_i^*(x_i)$, $i = 1, \dots, N$, satisfy the HJB equations

$$\min_{\mu_i \in \mathcal{A}_i(\Omega_i)} H_i(x_i, \mu_i, \nabla J_i^*(x_i)) = 0, \quad i = 1, \dots, N, \quad (10.2.10)$$

where

$$\nabla J_i^*(x_i) = \frac{\partial J_i^*(x_i)}{\partial x_i}, \quad i = 1, \dots, N.$$

Assume that the minima on the left-hand side of (10.2.10) exist and are unique. Then, the optimal control laws for the N isolated subsystems are

$$u_i^*(x_i) = \arg \min_{\mu_i \in \mathcal{A}_i(\Omega_i)} H_i(x_i, \mu_i, \nabla J_i^*(x_i)) \\ = -\frac{1}{2}R_i^{-1}g_i^\top(x_i)\nabla J_i^*(x_i), \quad i = 1, \dots, N. \quad (10.2.11)$$

Substituting the optimal control laws (10.2.11) into the nonlinear Lyapunov equations (10.2.8), we can obtain the formulation of the HJB equations in terms of $\nabla J_i^*(x_i)$, $i = 1, \dots, N$, as

$$Q_i^2(x_i) + (\nabla J_i^*(x_i))^\top f_i(x_i) - \frac{1}{4}(\nabla J_i^*(x_i))^\top g_i(x)R_i^{-1}g_i^\top(x_i)\nabla J_i^*(x_i) = 0 \quad (10.2.12)$$

with $J_i^*(0) = 0$ and $i = 1, \dots, N$.

According to (10.2.11), we have expressed the optimal control policies, i.e., $u_1^*(x_1), u_2^*(x_2), \dots, u_N^*(x_N)$, for the N isolated subsystems (10.2.4). We will show that by proportionally increasing some local feedback gains, a stabilizing decentralized control scheme can be established for the interconnected system (10.2.1). Now,

we give the following lemma, indicating how the feedback gains can be added, in order to guarantee the asymptotic stability of the isolated subsystems.

Lemma 10.2.1 *Consider the isolated subsystems (10.2.4), the feedback controls*

$$\begin{aligned}\bar{u}_i(x_i) &= \pi_i u_i^*(x_i) \\ &= -\frac{1}{2}\pi_i R_i^{-1} g_i^\top(x_i) \nabla J_i^*(x_i), \quad i = 1, \dots, N,\end{aligned}\quad (10.2.13)$$

ensure that the N closed-loop isolated subsystems are asymptotically stable for all $\pi_i \geq 1/2$, where $i = 1, \dots, N$.

Proof The lemma can be proved by showing that $J_i^*(x_i)$, $i = 1, \dots, N$, are Lyapunov functions. First of all, in light of (10.2.9), we can find that $J_i^*(x_i) > 0$ for any $x_i \neq 0$ and $J_i^*(x_i) = 0$ when $x_i = 0$, which implies that $J_i^*(x_i)$, $i = 1, \dots, N$, are positive-definite functions. Next, the derivatives of $J_i^*(x_i)$, $i = 1, \dots, N$, along the corresponding trajectories of the closed-loop isolated subsystems are given by

$$\begin{aligned}\dot{J}_i^*(x_i) &= (\nabla J_i^*(x_i))^\top \dot{x}_i \\ &= (\nabla J_i^*(x_i))^\top (f_i(x_i) + g_i(x_i) \bar{u}_i(x_i)),\end{aligned}\quad (10.2.14)$$

where $i = 1, \dots, N$. Then, by adding and subtracting $(1/4)(\nabla J_i^*(x_i))^\top g_i(x_i) u_i^*(x_i)$ to (10.2.14) and considering (10.2.11)–(10.2.13), we have

$$\begin{aligned}\dot{J}_i^*(x_i) &= (\nabla J_i^*(x_i))^\top f_i(x_i) - \frac{1}{4}(\nabla J_i^*(x_i))^\top g_i(x_i) R_i^{-1} g_i^\top(x_i) \nabla J_i^*(x_i) \\ &\quad - \frac{1}{2}\left(\pi_i - \frac{1}{2}\right)(\nabla J_i^*(x_i))^\top g_i(x_i) R_i^{-1} g_i^\top(x_i) \nabla J_i^*(x_i) \\ &= -Q_i^2(x_i) - \frac{1}{2}\left(\pi_i - \frac{1}{2}\right)\|R_i^{-1/2} g_i^\top(x_i) \nabla J_i^*(x_i)\|^2,\end{aligned}\quad (10.2.15)$$

where $i = 1, \dots, N$. Observing (10.2.15), we can obtain $\dot{J}_i^*(x_i) < 0$ for all $\pi_i \geq 1/2$ and $x_i \neq 0$, where $i = 1, \dots, N$. Therefore, the conditions for Lyapunov local stability theory are satisfied and the proof is complete.

Remark 10.2.1 Lemma 10.2.1 reveals that any feedback controls $\bar{u}_i(x_i)$, $i = 1, \dots, N$, can ensure the asymptotic stability of the closed-loop isolated subsystems as long as $\pi_i \geq 1/2$, $i = 1, \dots, N$. However, only when $\pi_i = 1$, $i = 1, \dots, N$, the feedback controls are optimal. In fact, similar results have been given in [3, 4, 22], showing that the optimal controls $u_i^*(x_i)$, $i = 1, \dots, N$, are robust in the sense that they have infinite gain margins.

Now, we present the main theorem of this section, based on which the acquired decentralized control strategy can be established.

Theorem 10.2.1 For the interconnected system (10.2.1), there exist N positive numbers $\pi_i^* > 0$, $i = 1, 2, \dots, N$, such that for any $\pi_i \geq \pi_i^*$, $i = 1, 2, \dots, N$, the feedback controls developed by (10.2.13) ensure that the closed-loop interconnected system is asymptotically stable. In other words, the control vector $(\bar{u}_1(x_1), \bar{u}_2(x_2), \dots, \bar{u}_N(x_N))$ is the decentralized control strategy of large-scale system (10.2.1).

Proof In accordance with Lemma 10.2.1, we observe that $J_i^*(x_i)$, $i = 1, \dots, N$, are all Lyapunov functions. Here, we choose a composite Lyapunov function given by

$$L(x) = \sum_{i=1}^N \theta_i J_i^*(x_i),$$

where θ_i , $i = 1, \dots, N$, are arbitrary positive constants.

Taking the time derivative of $L(x)$ along the trajectories of the closed-loop interconnected system, we obtain

$$\dot{L}(x) = \sum_{i=1}^N \theta_i \{ (\nabla J_i^*(x_i))^T (f_i(x_i) + g_i(x_i) \bar{u}_i(x_i)) + (\nabla J_i^*(x_i))^T g_i(x_i) \bar{Z}_i(x) \}. \quad (10.2.16)$$

Using (10.2.3), (10.2.6), and (10.2.15), from (10.2.16), it follows

$$\begin{aligned} \dot{L}(x) &\leq - \sum_{i=1}^N \theta_i \left\{ Q_i^2(x_i) + \frac{1}{2} \left(\pi_i - \frac{1}{2} \right) \| R_i^{-1/2} g_i^T(x_i) \nabla J_i^*(x_i) \|^2 \right. \\ &\quad \left. - \| (\nabla J_i^*(x_i))^T g_i(x_i) R_i^{-1/2} \| \| Z_i(x) \| \right\} \\ &\leq - \sum_{i=1}^N \theta_i \left\{ Q_i^2(x_i) + \frac{1}{2} \left(\pi_i - \frac{1}{2} \right) \| (\nabla J_i^*(x_i))^T g_i(x_i) R_i^{-1/2} \|^2 \right. \\ &\quad \left. - \| (\nabla J_i^*(x_i))^T g_i(x_i) R_i^{-1/2} \| \sum_{j=1}^N \lambda_{ij} Q_j(x_j) \right\}. \end{aligned} \quad (10.2.17)$$

For convenience, we denote

$$\Theta = \text{diag}\{\theta_1, \theta_2, \dots, \theta_N\}, \quad \Lambda = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1N} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{N1} & \lambda_{N2} & \cdots & \lambda_{NN} \end{bmatrix},$$

and

$$\Pi = \text{diag} \left\{ \frac{1}{2} \left(\pi_1 - \frac{1}{2} \right), \frac{1}{2} \left(\pi_2 - \frac{1}{2} \right), \dots, \frac{1}{2} \left(\pi_N - \frac{1}{2} \right) \right\}.$$

Then, by introducing a $2N$ -dimensional vector

$$\xi = \begin{bmatrix} Q_1(x_1) \\ Q_2(x_2) \\ \vdots \\ Q_N(x_N) \\ \hline \|\nabla J_1^*(x_1))^T g_1(x_1) R_1^{-1/2}\| \\ \|\nabla J_2^*(x_2))^T g_2(x_2) R_2^{-1/2}\| \\ \vdots \\ \|\nabla J_N^*(x_N))^T g_N(x_N) R_N^{-1/2}\| \end{bmatrix},$$

we can transform (10.2.17) into compact form as

$$\begin{aligned} \dot{L}(x) &\leq -\xi^T \begin{bmatrix} \Theta & -\frac{1}{2}\Lambda^T\Theta \\ \hline -\frac{1}{2}\Theta\Lambda & \Theta\Pi \end{bmatrix} \xi \\ &\triangleq -\xi^T \mathcal{E} \xi. \end{aligned} \quad (10.2.18)$$

According to (10.2.18), sufficiently large $\pi_i, i = 1, \dots, N$, can be chosen such that the matrix \mathcal{E} is positive definite. That is to say, there exist $\pi_i^*, i = 1, \dots, N$, such that any $\pi_i \geq \pi_i^*, i = 1, \dots, N$, are large enough to guarantee the positive definiteness of \mathcal{E} . Then, we have $\dot{L}(x) < 0$. Therefore, the closed-loop interconnected system is asymptotically stable under the action of the control vector $(\bar{u}_1(x_1), \bar{u}_2(x_2), \dots, \bar{u}_N(x_N))$. The proof is complete.

Clearly, the focal point of designing the decentralized control strategy becomes to derive the optimal controllers for the N isolated subsystems on the basis of Theorem 10.2.1. Then, we should put our emphasis on solving the HJB equations, which yet is regarded as a difficult task [10, 25]. Hence, in what follows we shall employ a more pragmatic approach to obtain the approximate solutions based on online PI algorithm and NN techniques.

10.2.2 Optimal Controller Design of Isolated Subsystems

In this section, the online PI algorithm is introduced to solve the HJB equations. The PI algorithm consists of policy evaluation based on (10.2.8) and policy improvement based on (10.2.11) [21]. Specifically, its iteration procedure can be described as follows.

Step 1: Choose a small positive number ε . Let $p = 0$ and $V_i^{(0)}(x_i) = 0$, where $i = 1, \dots, N$. Then, start with N initial admissible control laws $\mu_1^{(0)}(x_1), \mu_2^{(0)}(x_2), \dots, \mu_N^{(0)}(x_N)$.

Step 2: Let $p = p + 1$. Based on the control laws $\mu_i^{(p-1)}(x_i), i = 1, \dots, N$, solve the following nonlinear Lyapunov equations

$$0 = Q_i^2(x_i) + (\mu_i^{(p-1)}(x_i))^T R_i \mu_i^{(p-1)}(x_i) + (\nabla V_i^{(p)}(x_i))^T (f_i(x_i) + g_i(x_i) \mu_i^{(p-1)}(x_i)) \quad (10.2.19)$$

with $V_i^{(p)}(0) = 0$ and $i = 1, \dots, N$.

Step 3: Update the control laws via

$$\mu_i^{(p)}(x_i) = -\frac{1}{2} R_i^{-1} g_i^T(x_i) \nabla V_i^{(p)}(x_i), \quad (10.2.20)$$

where $i = 1, \dots, N$.

Step 4: If $|V_i^{(p)}(x_i) - V_i^{(p-1)}(x_i)| \leq \varepsilon, i = 1, \dots, N$, stop and obtain the approximate optimal controls of the N isolated subsystems; else, go back to Step 2.

Note that N initial admissible control laws are required in the above algorithm. In the following theorems, we present the convergence analysis of the online PI algorithm for the isolated subsystems.

Theorem 10.2.2 Consider the N isolated subsystems (10.2.4), given N initial admissible control laws $\mu_1^{(0)}(x_1), \mu_2^{(0)}(x_2), \dots, \mu_N^{(0)}(x_N)$. Then, using the PI algorithm established in (10.2.19) and (10.2.20), the value functions and control laws converge to the optimal ones as $p \rightarrow \infty$, i.e., $V_i^{(p)}(x_i) \rightarrow J_i^*(x_i)$ and $\mu_i^{(p)}(x_i) \rightarrow u_i^*(x_i)$ as $p \rightarrow \infty$, where $i = 1, \dots, N$.

Proof First, we consider the subsystem i . According to [1], when given an initial admissible control law $\mu_i^{(0)}(x_i)$, we have $\mu_i^{(p)}(x_i) \in \mathcal{A}_i(\Omega_i)$ for any $p \geq 0$. Additionally, for any $\zeta > 0$, there exists an integer p_{0i} , such that for any $p \geq p_{0i}$, the inequalities

$$\sup_{x_i \in \Omega_i} |V_i^{(p)}(x_i) - J_i^*(x_i)| < \zeta \quad (10.2.21)$$

and

$$\sup_{x_i \in \Omega_i} \|\mu_i^{(p)}(x_i) - u_i^*(x_i)\| < \zeta \quad (10.2.22)$$

hold simultaneously.

Next, we consider the N isolated subsystems. When given $\mu_1^{(0)}(x_1), \mu_2^{(0)}(x_2), \dots, \mu_N^{(0)}(x_N)$, where $\mu_i^{(0)}(x_i)$ is the initial admissible control law corresponding to the i th subsystem, we can acquire that $\mu_i^{(p)}(x_i) \in \mathcal{A}_i(\Omega_i)$ for any $p \geq 0$, where $i = 1, \dots, N$. Moreover, we denote $p_0 = \max\{p_{01}, p_{02}, \dots, p_{0N}\}$. Thus, we can conclude that for any $\zeta > 0$, there exists an integer p_0 , such that for any $p \geq p_0$, (10.2.21) and (10.2.22)

are true with $i = 1, \dots, N$. In other words, the algorithm will converge to the optimal cost functions and optimal control laws of the N isolated subsystems. The proof is complete.

Next, we describe the NN implementation process of the online PI algorithm. For the N isolated subsystems, assume that the value functions $V_i(x_i)$, $i = 1, \dots, N$, are continuously differentiable. Then, according to the universal approximation property of NNs, $V_i(x_i)$ can be reconstructed by an NN on a compact set Ω_i as

$$V_i(x_i) = W_c^{i\top} \sigma_c^i(x_i) + \varepsilon_c^i(x_i), \quad i = 1, \dots, N,$$

where $W_c^i \in \mathbb{R}^{l_i}$ is the ideal weight, $\sigma_c^i(x_i) \in \mathbb{R}^{l_i}$ is the activation function, l_i is the number of neurons in the hidden layer, and $\varepsilon_c^i(x_i) \in \mathbb{R}$ is the approximation error of the i th NN, $i = 1, \dots, N$.

The derivatives of the value functions with respect to their state vectors are formulated as

$$\nabla V_i(x_i) = \nabla \sigma_c^{i\top}(x_i) W_c^i + \nabla \varepsilon_c^i(x_i), \quad i = 1, \dots, N, \quad (10.2.23)$$

where $\nabla \sigma_c^i(x_i) = \partial \sigma_c^i(x_i) / \partial x_i \in \mathbb{R}^{l_i \times n_i}$ and $\nabla \varepsilon_c^i(x_i) = \partial \varepsilon_c^i(x_i) / \partial x_i \in \mathbb{R}^{n_i}$ are the gradients of the activation function and approximation error of the i th NN, respectively, $i = 1, \dots, N$. Based on (10.2.23), the Lyapunov equations (10.2.8) become

$$0 = Q_i^2(x_i) + \mu_i^\top R_i \mu_i + (W_c^{i\top} \nabla \sigma_c^i(x_i) + \nabla \varepsilon_c^{i\top}(x_i)) \dot{x}_i,$$

where $i = 1, \dots, N$ and \dot{x}_i is given in (10.2.4).

For the i th NN, $i = 1, \dots, N$, assume that the NN weight vector W_c^i , the gradient $\nabla \sigma_c^i(x_i)$, and the approximation error $\varepsilon_c^i(x_i)$, and its derivative $\nabla \varepsilon_c^i(x_i)$ are all bounded on the compact set Ω_i . Moreover, according to [23], we have $\varepsilon_c^i(x_i) \rightarrow 0$ and $\nabla \varepsilon_c^i(x_i) \rightarrow 0$ as $l_i \rightarrow \infty$, where $i = 1, \dots, N$.

Because the ideal weights are unknown, N critic NNs can be built to approximate the value functions as

$$\hat{V}_i(x_i) = \hat{W}_c^{i\top} \sigma_c^i(x_i), \quad i = 1, \dots, N,$$

where \hat{W}_c^i , $i = 1, \dots, N$, are the estimated weights. Here, $\sigma_c^i(x_i)$, $i = 1, \dots, N$, are selected such that $\hat{V}_i(x_i) > 0$ for any $x_i \neq 0$ and $\hat{V}_i(x_i) = 0$ when $x_i = 0$.

Similarly, the derivatives of the approximate value functions with respect to the state vectors can be expressed as

$$\nabla \hat{V}_i(x_i) = \nabla \sigma_c^{i\top}(x_i) \hat{W}_c^i, \quad i = 1, \dots, N,$$

where $\nabla \hat{V}_i(x_i) = \partial \hat{V}_i(x_i) / \partial x_i$, $i = 1, \dots, N$. Then, the approximate Hamiltonians can be obtained as

$$\begin{aligned} H_i(x_i, \mu_i, \hat{W}_c^i) &= Q_i^2(x_i) + \mu_i^T R_i \mu_i + \hat{W}_c^{i\top} \nabla \sigma_c^i(x_i) \dot{x}_i \\ &= e_{ci}, \quad i = 1, \dots, N. \end{aligned} \quad (10.2.24)$$

For the purpose of training the critic networks of the isolated subsystems, it is desired to obtain $\hat{W}_c^i, i = 1, \dots, N$, to minimize the following objective functions:

$$E_{ci} = \frac{1}{2} e_{ci}^2, \quad i = 1, \dots, N.$$

The standard steepest descent algorithm is introduced to tune the critic networks. Then, their weights are updated through

$$\dot{\hat{W}}_c^i = -\alpha_{ci} \left[\frac{\partial E_{ci}}{\partial \hat{W}_c^i} \right], \quad i = 1, \dots, N, \quad (10.2.25)$$

where $\alpha_{ci} > 0, i = 1, \dots, N$, are the learning rates of the critic networks.

On the other hand, based on (10.2.23), the Hamiltonians take the following forms:

$$\begin{aligned} H_i(x_i, \mu_i, W_c^i) &= Q_i^2(x_i) + \mu_i^T R_i \mu_i + W_c^{i\top} \nabla \sigma_c^i(x_i) \dot{x}_i \\ &= e_{cHi}, \quad i = 1, \dots, N, \end{aligned} \quad (10.2.26)$$

where

$$e_{cHi} = -(\nabla \varepsilon_c^i(x_i))^T \dot{x}_i, \quad i = 1, \dots, N,$$

are the residual errors due to the NN approximation.

Denote

$$\delta_i = \nabla \sigma_c^i(x_i) \dot{x}_i, \quad i = 1, \dots, N.$$

We assume that there exist N positive constants $\delta_{Mi}, i = 1, \dots, N$, such that

$$\|\delta_i\| \leq \delta_{Mi}, \quad i = 1, \dots, N. \quad (10.2.27)$$

Moreover, we define the weight estimation errors of the critic networks as $\tilde{W}_c^i = W_c^i - \hat{W}_c^i$, where $i = 1, \dots, N$. Then, combining (10.2.24) with (10.2.26) yields

$$e_{cHi} - e_{ci} = \tilde{W}_c^{i\top} \delta_i, \quad i = 1, \dots, N.$$

Therefore, the dynamics of the weight estimation errors can be given as

$$\dot{\tilde{W}}_c^i = \alpha_{ci} (e_{cHi} - \tilde{W}_c^{i\top} \delta_i) \delta_i, \quad i = 1, \dots, N. \quad (10.2.28)$$

Incidentally, the persistency of excitation condition is required to tune the i th critic network to guarantee that $\|\delta_i\| \geq \delta_{mi}$, where $\delta_{mi}, i = 1, \dots, N$, are positive constants.

Thus, a set small exploratory signals will be added to the isolated subsystems in order to satisfy the condition in practice.

When implementing the online PI algorithm, in order to accomplish the policy improvement, we should obtain the control polices that minimize the value functions. Hence, according to (10.2.11) and (10.2.23), we have

$$\begin{aligned}\mu_i(x_i) &= -\frac{1}{2}R_i^{-1}g_i^\top(x_i)\nabla V_i(x_i) \\ &= -\frac{1}{2}R_i^{-1}g_i^\top(x_i)\left(\nabla\sigma_c^{i\top}(x_i)W_c^i + \nabla\varepsilon_c^i(x_i)\right),\end{aligned}$$

where $i = 1, \dots, N$. Hence, the approximate control policies can be obtained as

$$\begin{aligned}\hat{\mu}_i(x_i) &= -\frac{1}{2}R_i^{-1}g_i^\top(x_i)\nabla\hat{V}_i(x_i) \\ &= -\frac{1}{2}R_i^{-1}g_i^\top(x_i)\nabla\sigma_c^{i\top}(x_i)\hat{W}_c^i,\end{aligned}\quad (10.2.29)$$

where $i = 1, \dots, N$.

Remark 10.2.2 According to (10.2.29), it is obvious to observe that the approximate control laws of the N isolated subsystems can be derived directly based on the trained critic networks. Consequently, unlike the traditional actor–critic architecture, the action NNs are not required any more.

When considering the critic networks, the weight estimation dynamics are uniformly ultimately bounded (UUB) as described in the following theorem.

Theorem 10.2.3 *For the N isolated subsystems (10.2.4), the weight update laws for tuning the critic networks are given by (10.2.25). Then, the dynamics of the weight estimation errors of the critic networks are UUB.*

Proof Choose N Lyapunov function candidates described as follows:

$$L_i(x) = \frac{1}{\alpha_{ci}}\text{tr}(\tilde{W}_c^{i\top}\tilde{W}_c^i) = \frac{1}{\alpha_{ci}}\tilde{W}_c^{i\top}\tilde{W}_c^i, \quad i = 1, \dots, N.$$

The time derivatives of the Lyapunov functions $L_i(x)$, $i = 1, \dots, N$, along the trajectories of the error dynamics (10.2.28) are

$$\dot{L}_i(x) = \frac{2}{\alpha_{ci}}\tilde{W}_c^{i\top}\dot{\tilde{W}}_{ci} = \frac{2}{\alpha_{ci}}[\tilde{W}_c^{i\top}\alpha_{ci}(e_{cHi} - \tilde{W}_c^{i\top}\delta_i)\delta_i],$$

where $i = 1, \dots, N$. After some basic mathematical manipulations and considering the Cauchy–Schwarz inequality, it yields

$$\begin{aligned}\dot{L}_i(x) &= \frac{2}{\alpha_{ci}} [e_{cHi} \alpha_{ci} \tilde{W}_c^{i\top} \delta_i - \alpha_{ci} (\tilde{W}_c^{i\top} \delta_i)^2] \\ &\leq \frac{1}{\alpha_{ci}} e_{cHi}^2 - (2 - \alpha_{ci}) (\tilde{W}_c^{i\top} \delta_i)^2,\end{aligned}$$

where $i = 1, \dots, N$. It can be found that $\dot{L}_i(x) < 0$ as long as

$$\begin{cases} 0 < \alpha_{ci} < 2, \\ |\tilde{W}_c^{i\top} \delta_i| > \sqrt{\frac{e_{cHi}^2}{\alpha_{ci}(2 - \alpha_{ci})}}, \end{cases} \quad (10.2.30)$$

where $i = 1, \dots, N$. By employing the dense property of real numbers [17], we derive that there exist some positive constants κ_i ($0 < \kappa_i \leq \delta_{Mi}$) such that

$$|\tilde{W}_c^{i\top} \delta_i| > \kappa_i \|\tilde{W}_c^i\| > \sqrt{\frac{e_{cHi}^2}{\alpha_{ci}(2 - \alpha_{ci})}}. \quad (10.2.31)$$

By noticing (10.2.31), we can conclude that $\dot{L}_i(x) < 0$ as long as

$$\begin{cases} 0 < \alpha_{ci} < 2, \\ \|\tilde{W}_c^i\| > \sqrt{\frac{e_{cHi}^2}{\alpha_{ci}\kappa_i^2(2 - \alpha_{ci})}}, \end{cases} \quad (10.2.32)$$

where $i = 1, \dots, N$. In accordance with the Lyapunov stability theory, we conclude that the dynamics of the weight estimation errors of the critic networks are all UUB. Meanwhile, the norms of the weight estimation errors are bounded as well. The proof is complete.

Remark 10.2.3 Let

$$\hat{u}_i(x_i) = \pi_i \hat{\mu}_i(x_i),$$

where $\hat{\mu}_i(x_i)$, $i = 1, \dots, N$, are obtained by (10.2.29). According to the selections of the activation functions of the critic networks, we can easily find that the approximate value functions $\hat{V}_i(x_i)$, $i = 1, \dots, N$, are also Lyapunov functions. Furthermore, similar to the proof of Theorem 10.2.1, we have

$$\dot{L}(x) \leq -\xi^\top \Xi \xi + \Sigma_e,$$

where Σ_e is the sum of the approximation errors. Hence, we can conclude that based on the approximate optimal control laws $\hat{\mu}_i(x_i)$, $i = 1, \dots, N$, the present control vector $(\hat{u}_1(x_1), \hat{u}_2(x_2), \dots, \hat{u}_N(x_N))$ can ensure the uniform ultimate boundedness of the state trajectories of the closed-loop interconnected system. It is in this sense

that we accomplish the design of the decentralized control scheme by adopting the learning optimal control approach based on online PI algorithm.

Remark 10.2.4 Note that the controller presented here is a decentralized stabilizing one. Though the optimal decentralized controller of interconnected systems has been studied before [19], in this section, we aim at developing a novel decentralized control strategy based on ADP. How to extend the present results to the design of optimal decentralized control for interconnected nonlinear systems is of our future research. In fact, in the recent work, Wang et al. [26] has provided some preliminary results on this topic.

10.2.3 Generalization to Model-Free Decentralized Control

Now, we generalize the above results to model-free decentralized control of interconnected nonlinear systems [14]. We develop an online model-free integral PI algorithm for optimal control problems with completely unknown system dynamics. To deal with exploration which relaxes the assumptions of exact knowledge on $f_i(x_i)$ and $g_i(x_i)$, we consider the following nonlinear subsystem explored by a known bounded piecewise continuous signal $e_i(t)$

$$\dot{x}_i(t) = f_i(x_i(t)) + g_i(x_i(t))[u_i(x_i(t)) + e_i(t)]. \quad (10.2.33)$$

The derivative of the value function with respect to time along the trajectory of the subsystem (10.2.33) is calculated using (10.2.8) as

$$\begin{aligned} \dot{V}_i(x_i) &= \nabla V_i^T(x_i)(f_i(x_i) + g_i(x_i)[\mu_i(x_i) + e_i]) \\ &= -r_i(x_i, \mu_i) + \nabla V_i^T(x_i)g_i(x_i)e_i, \end{aligned} \quad (10.2.34)$$

where

$$r_i(x_i, \mu_i) = Q_i^2(x_i) + \mu_i^T(x_i)R_i\mu_i(x_i).$$

We present a lemma which is essential to prove the convergence of the model-free PI algorithm for the isolated subsystems.

Lemma 10.2.2 *Solving for $V_i(x_i)$ in the following equation*

$$V_i(x_i(t+T)) - V_i(x_i(t)) = \int_t^{t+T} \nabla V_i^T(x_i)g_i(x_i)e_i d\tau - \int_t^{t+T} r_i(x_i, \mu_i(x_i)) d\tau \quad (10.2.35)$$

is equivalent to finding the solution of (10.2.34).

Proof Since $\mu_i(x_i) \in \mathcal{A}_i(\Omega_i)$, the value function $V_i(x_i)$ is a Lyapunov function for the subsystem (10.2.33), and it satisfies (10.2.34) with $r_i(x_i, \mu_i) > 0$, $x_i \neq 0$. We

integrate (10.2.34) over the interval $[t, t + T]$ to obtain (10.2.35). This means that the unique solution of (10.2.34), $V_i(x_i)$, also satisfies (10.2.35). To complete the proof, we show that (10.2.35) has a unique solution by contradiction.

Thus, we assume that there exists another value function $\bar{V}_i(x_i)$ which satisfies (10.2.35) with the end condition (boundary condition) $\bar{V}_i(0) = 0$. This value function also satisfies

$$\dot{\bar{V}}_i(x_i) = -r_i(x_i, \mu_i) + \nabla \bar{V}_i^\top(x_i) g_i(x_i) e_i.$$

Subtracting this from (10.2.34), we obtain

$$\dot{V}_i(x_i) - \dot{\bar{V}}_i(x_i) = \left(\frac{d[\bar{V}_i(x_i) - V_i(x_i)]}{dx_i} \right)^\top \dot{x}_i = (\nabla V_i(x_i) - \nabla \bar{V}_i(x_i))^\top g_i(x_i) e_i,$$

or,

$$\begin{aligned} 0 &= \left(\frac{d[\bar{V}_i(x_i) - V_i(x_i)]}{dx_i} \right)^\top (\dot{x}_i - g_i(x_i) e_i) \\ &= \left(\frac{d[\bar{V}_i(x_i) - V_i(x_i)]}{dx_i} \right)^\top (f_i(x_i) + g_i(x_i) \mu_i(x_i)), \end{aligned} \quad (10.2.36)$$

which must hold for any x_i on the system trajectories generated by the stabilizing policy $\mu_i(x_i)$. According to (10.2.36), we have $\bar{V}_i(x_i) = V_i(x_i) + c$. As this relation must hold for $x_i(t) = 0$, we have $\bar{V}_i(0) = V_i(0) + c \Rightarrow c = 0$. Thus, $\bar{V}_i(x_i) = V_i(x_i)$, i.e., (10.2.35) has a unique solution which is equal to the solution of (10.2.34). The proof is complete.

Integrating (10.2.34) from t to $t + T$ with any time interval $T > 0$, and considering (10.2.19) and (10.2.20), we have

$$\begin{aligned} V_i^{(p)}(x_i(t + T)) - V_i^{(p)}(x_i(t)) \\ = -2 \int_t^{t+T} (\mu_i^{(p)}(x_i))^\top R_i e_i d\tau \\ - \int_t^{t+T} \{Q_i^2(x_i) + (\mu_i^{(p)}(x_i))^\top R_i \mu_i^{(p)}(x_i)\} d\tau. \end{aligned} \quad (10.2.37)$$

Equation (10.2.37) which is derived by (10.2.20) and (10.2.35) plays an important role in relaxing the knowledge of the system dynamics, since $f_i(x_i)$ and $g_i(x_i)$ do not appear in the equation. It means that the iteration can be done without knowing the system dynamics. Thus, we obtain the online model-free integral PI algorithm as in Algorithm 10.2.1.

Algorithm 10.2.1 Online model-free integral PI algorithm

Step 1. Give a small positive real number ε . Let $p = 0$ and start with an initial admissible control law $\mu_i^{(0)}(x_i)$.

Step 2. Policy Evaluation and Improvement:

Based on the control law $\mu_i^{(p)}(x_i)$, solve the following nonlinear Lyapunov equations for $V_i^{(p)}(x_i)$ and $\mu_i^{(p+1)}(x_i)$

$$V_i^{(p)}(x_i(t)) = \int_t^{t+T} \{Q_i^2(x_i) + (\mu_i^{(p)}(x_i))^T R_i \mu_i^{(p)}(x_i)\} dt$$

$$+ 2 \int_t^{t+T} (\mu_i^{(p+1)}(x_i))^T R_i e_i dt + V_i^{(p)}(x_i(t+T)). \quad (10.2.38)$$

Step 3. If $|V_i^{(p)}(x_i) - V_i^{(p-1)}(x_i)| \leq \varepsilon$, stop and obtain the approximate optimal control law of the i th isolated subsystem; else, set $p = p + 1$ and go to Step 2.

Note that N initial admissible control laws are required in this algorithm, and we let $V_i^{(p)}(x_i) = 0$, when $p = 0$.

Theorem 10.2.4 *Considering the isolated subsystems (10.2.4), we give N initial admissible control laws $\mu_1^{(0)}(x_1), \mu_2^{(0)}(x_2), \dots, \mu_N^{(0)}(x_N)$. Then, using the PI algorithm established in (10.2.38), the value functions and control laws converge to the optimal ones as $p \rightarrow \infty$, i.e., $V_i^{(p)}(x_i) \rightarrow J_i^*(x_i)$, $\mu_i^{(p)}(x_i) \rightarrow u_i^*(x_i)$.*

Proof In [1], it was shown that in the iteration process of (10.2.20) and (10.2.34), if the initial control policy $\mu_i^{(0)}(x_i)$ is admissible, all the subsequent control laws will be admissible. Moreover, the iteration result will converge to the solution of the HJB equation. Based on (10.2.37) and the proven equivalence between (10.2.34) and (10.2.35), we can conclude that the present online model-free PI algorithm will converge to the solution of the optimal control problem for subsystem (10.2.33) without using the knowledge of system dynamics. The proof is complete.

Now, we discuss the NN-based implementation method of the established model-free PI algorithm. A critic NN and an action NN are used to approximate the value function and the control law of the subsystem, respectively. We assume that for the i th subsystem, $V_i(x_i)$ and $\mu_i(x_i)$ are represented on a compact set Ω_i by single-layer NNs as $V_i(x_i) = (W_c^i)^T \phi_c^i(x_i) + \varepsilon_c^i(x_i)$ and $\mu_i(x_i) = (W_a^i)^T \phi_a^i(x_i) + \varepsilon_a^i(x_i)$, where $W_c^i \in \mathbb{R}^{N_c^i}$ and $W_a^i \in \mathbb{R}^{N_a^i}$ are unknown bounded ideal NN weight parameters which will be determined by the established model-free PI algorithm, $\phi_c^i(x_i) \in \mathbb{R}^{N_c^i}$ and $\phi_a^i(x_i) \in \mathbb{R}^{N_a^i}$ are the continuously differentiable nonlinear activation functions, and $\varepsilon_c^i(x_i) \in \mathbb{R}$ and $\varepsilon_a^i(x_i) \in \mathbb{R}^{m_i}$ are the bounded NN approximation errors. Here, the subscripts “ c ” and “ a ” denote the critic and the action, respectively. Since the ideal weights are unknown, the outputs of the critic NN and the action NN are estimated as

$$\hat{V}_i(x_i) = (\hat{W}_c^i)^\top \phi_c^i(x_i), \quad (10.2.39)$$

$$\hat{\mu}_i(x_i) = (\hat{W}_a^i)^\top \phi_a^i(x_i), \quad (10.2.40)$$

where \hat{W}_c^i and \hat{W}_a^i are the current estimated weights.

Using (10.2.39) and (10.2.40), (10.2.38) can be rewritten in a general form as

$$[\psi_k^i]^\top \begin{bmatrix} \hat{W}_c^i \\ \hat{W}_a^i \end{bmatrix} = \theta_k^i \quad (10.2.41)$$

with

$$\begin{aligned} \theta_k^i &= \int_{t+(k-1)T}^{t+kT} \{Q_i^2(x_i) + \mu_i^\top(x_i)R_i\mu_i(x_i)\} d\tau, \\ \psi_k^i &= \left[(\phi_c^i(x_i(t + (k-1)T)) - \phi_c^i(x_i(t + kT)))^\top, \right. \\ &\quad \left. - 2 \int_{t+(k-1)T}^{t+kT} R_i e_i(\phi_a^i(x_i))^\top d\tau \right]^\top, \end{aligned}$$

where the measurement time is from $t + (k-1)T$ to $t + kT$. Since (10.2.41) is only a 1-dimensional equation, we cannot guarantee the uniqueness of the solution. Similar to [8], we use the least squares method to solve the parameter vector over a compact set Ω_i . For any positive integral K_i , we denote $\Phi_i = [\psi_1^i, \psi_2^i, \dots, \psi_{K_i}^i]$ and $\Theta_i = [\theta_1^i, \theta_2^i, \dots, \theta_{K_i}^i]^\top$. Then, we have the following K_i -dimensional equation

$$\Phi_i^\top \begin{bmatrix} \hat{W}_c^i \\ \hat{W}_a^i \end{bmatrix} = \Theta_i.$$

If Φ_i^\top has full column rank, the parameters can be obtained by solving the equation

$$\begin{bmatrix} \hat{W}_c^i \\ \hat{W}_a^i \end{bmatrix} = (\Phi_i \Phi_i^\top)^{-1} \Phi_i \Theta_i. \quad (10.2.42)$$

Therefore, we need to guarantee that the number of collected data points K_i satisfies $K_i \geq \text{rank}(\Phi_i) = N_c^i + N_a^i$, which will guarantee the existence of $(\Phi_i \Phi_i^\top)^{-1}$. The least squares problem in (10.2.42) can be solved in real time by collecting enough data points generated from the system (10.2.33).

Clearly, the problem of designing the decentralized control law becomes to derive the optimal controllers for the N isolated subsystems. Based on the online model-free integral PI algorithm and NN techniques, we obtain the approximate solutions of HJB equations. We can conclude that the approximate optimal control policies $\hat{\mu}_i(x_i)$ is obtained. Therefore, according to Theorem 10.2.1 and Remark 10.2.3, the stabilizing decentralized control law of the large-scale interconnected system can be derived.

10.2.4 Simulation Studies

Two simulation examples are provided to show the applicability of the decentralized control strategy established in this section.

Example 10.2.1 Consider the following continuous-time large-scale nonlinear system consisting of two interconnected subsystems given by

$$\begin{aligned}\dot{x}_1 &= \begin{bmatrix} -x_{11} + x_{12} \\ -0.5x_{11} - 0.5x_{12} - 0.5x_{12}(\cos(2x_{11}) + 2)^2 \end{bmatrix} \\ &\quad + \begin{bmatrix} 0 \\ \cos(2x_{11}) + 2 \end{bmatrix} (\bar{u}_1(x_1) + (x_{11} + x_{22}) \sin x_{12}^2 \cos(0.5x_{21})), \\ \dot{x}_2 &= \begin{bmatrix} x_{22} \\ -x_{21} - 0.5x_{22} + 0.5x_{21}^2 x_{22} \end{bmatrix} \\ &\quad + \begin{bmatrix} 0 \\ x_{21} \end{bmatrix} (\bar{u}_2(x_2) + 0.5(x_{12} + x_{22}) \cos(e^{x_{21}^2})),\end{aligned}\quad (10.2.43)$$

where $x_1 = [x_{11}, x_{12}]^\top \in \mathbb{R}^2$ and $\bar{u}_1(x_1) \in \mathbb{R}$ are the state and control variables of subsystem 1, and $x_2 = [x_{21}, x_{22}]^\top \in \mathbb{R}^2$ and $\bar{u}_2(x_2) \in \mathbb{R}$ are the state and control variables of subsystem 2. Let R_1 and $R_2 = I$ be identity matrices with suitable dimensions. Additionally, let $h_1(x_1) = \|x_1\|$ and $h_2(x_2) = |x_{22}|$. Then, we find that $Z_1(x)$ and $Z_2(x)$ with $x = [x_1^\top, x_2^\top]^\top$ are upper bounded as in (10.2.3). For example, we can select $\lambda_{11} = \lambda_{12} = 1$ and $\lambda_{21} = \lambda_{22} = 1/2$.

In order to design the decentralized controller of interconnected system (10.2.43), we first deal with the optimal control problem of two isolated subsystems. Here, we choose $Q_1(x_1) = \|x_1\|$ and $Q_2(x_2) = |x_{22}|$. Hence, the cost functions of the optimal control problem are, respectively,

$$J_1(x_{10}) = \int_0^\infty (x_{11}^2 + x_{12}^2 + u_1^\top u_1) d\tau$$

and

$$J_2(x_{20}) = \int_0^\infty (x_{22}^2 + u_2^\top u_2) d\tau.$$

We adopt the online PI algorithm to tackle the optimal control problem, where two critic networks are constructed to approximate the cost functions. We denote the weight vectors of the two critic networks as $\hat{W}_c^1 = [\hat{W}_{c1}^1, \hat{W}_{c2}^1, \hat{W}_{c3}^1]^\top$ and $\hat{W}_c^2 = [\hat{W}_{c1}^2, \hat{W}_{c2}^2, \hat{W}_{c3}^2]^\top$. During the simulation process, the initial weights of the critic networks are chosen randomly in $[0, 2]$. Moreover, the activation functions of the two critic networks are chosen as $\sigma_{c1}(x_1) = [x_{11}^2, x_{11}x_{12}, x_{12}^2]^\top$ and $\sigma_{c2}(x_2) = [x_{21}^2, x_{21}x_{22}, x_{22}^2]^\top$, respectively. Besides, choose the learning rates of the critic networks as $\alpha_{c1} = \alpha_{c2} = 0.1$ and the initial states of the two isolated subsystems as $x_{10} = x_{20} = [1, -1]^\top$.

During the implementation process of the online PI algorithm, for each isolated subsystem, we add the following small exploratory signals to satisfy the persistency of excitation condition:

$$\begin{aligned}\mathcal{N}_1(t) = & \sin^2(t) \cos(t) + \sin^2(2t) \cos(0.1t) + \sin^2(-1.2t) \cos(0.5t) \\ & + \sin^5(t) + \sin^2(1.12t) + \cos(2.4t) \sin^3(2.4t)\end{aligned}$$

and $\mathcal{N}_2(t) = 1.6\mathcal{N}_1(t)$. We can observe that the convergence of the weights occurred after 750 and 180s, respectively. Then, the exploratory signals are turned off. Actually, the weights of the critic networks converge to $\hat{W}_c^1 = [0.498969, 0.000381, 0.999843]^\top$ and $\hat{W}_c^2 = [1.000002, -0.000021, 0.999992]^\top$, respectively, which are depicted in Figs. 10.2 and 10.3.

Based on the converged weights \hat{W}_c^1 and \hat{W}_c^2 , we can obtain the approximate value function and control law for each isolated subsystem, namely, $\hat{V}_1(x_1)$, $\hat{\mu}_1(x_1)$, $\hat{V}_2(x_2)$, and $\hat{\mu}_2(x_2)$. In comparison, for the method proposed in [23], the optimal cost function and control law of isolated subsystem 1 are $J_1^*(x_1) = 0.5x_{11}^2 + x_{12}^2$ and $u_1^*(x_1) = -(\cos(2x_{11}) + 2)x_{12}$, respectively. Similarly, the optimal cost function and control law of isolated subsystem 2 are $J_2^*(x_2) = x_{21}^2 + x_{22}^2$ and $u_2^*(x_2) = -x_{21}x_{22}$.

As a result, for isolated subsystem 1, the error between the optimal cost function and the approximate one is presented in Fig. 10.4. Moreover, the error between the optimal control law and the approximate version is shown in Fig. 10.5. It is clear to see that both the approximation errors are close to zero, which verifies the good

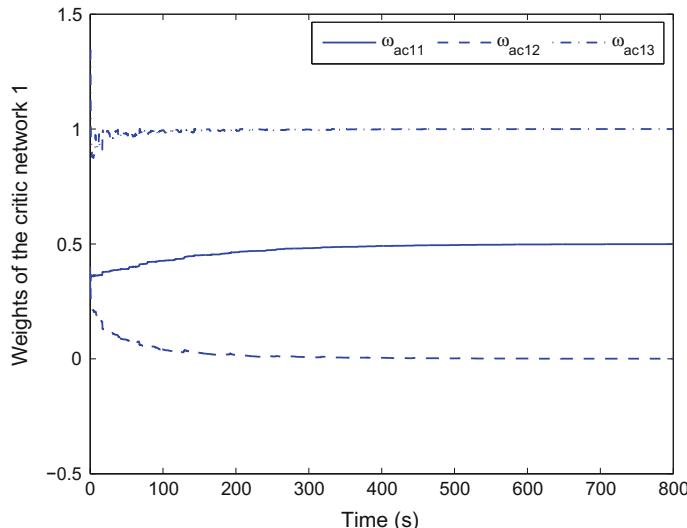


Fig. 10.2 Convergence of the weight vector of the critic network 1 (ω_{ac11} , ω_{ac12} , and ω_{ac13} represent \hat{W}_c^1 , \hat{W}_c^2 , and \hat{W}_c^3 , respectively)

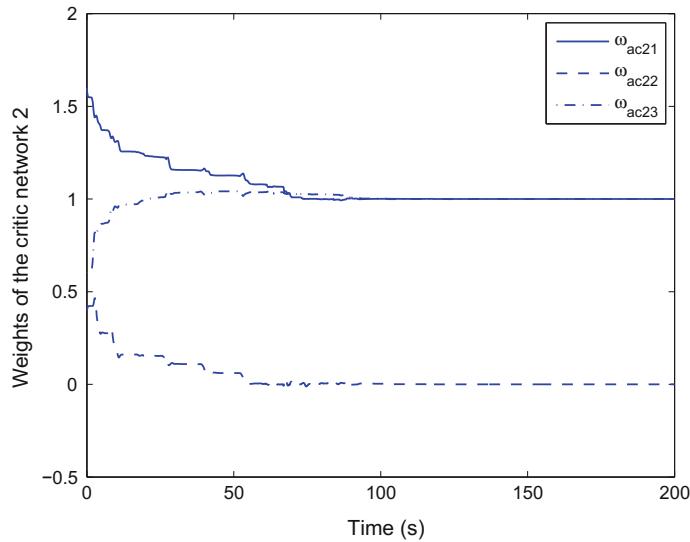


Fig. 10.3 Convergence of the weight vector of the critic network 2 (ω_{ac21} , ω_{ac22} , and ω_{ac23} represent \hat{W}_{c1}^2 , \hat{W}_{c2}^2 , and \hat{W}_{c3}^2 , respectively)

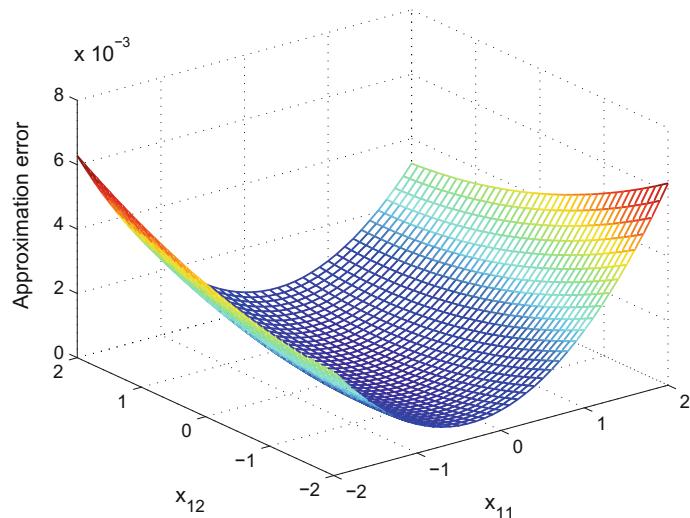


Fig. 10.4 3-D plot of the approximation error of the cost function of isolated subsystem 1, i.e., $J_1^*(x_1) - \hat{V}_1(x_1)$

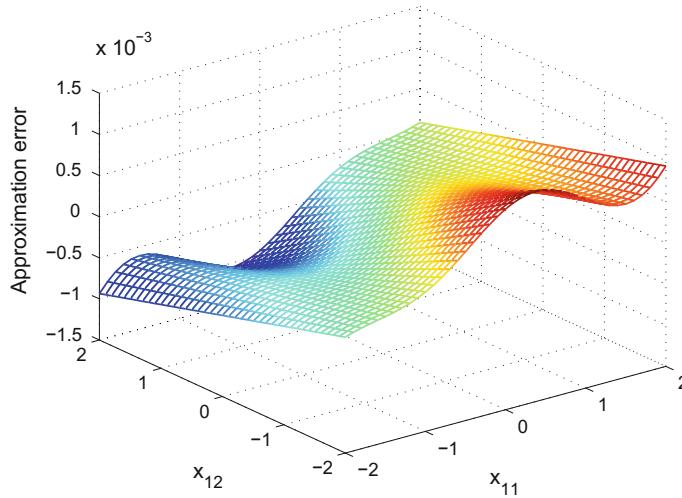


Fig. 10.5 3-D plot of the approximation error of the control law of isolated subsystem 1, i.e., $u_1^*(x_1) - \hat{\mu}_1(x_1)$

performance of the online learning algorithm. When regarding the isolated subsystem 2, we obtain the same simulation results shown in Figs. 10.6 and 10.7.

Next, by choosing $\theta_1 = \theta_2 = 1$ and $\pi_1 = \pi_2 = 2$, we can guarantee the positive definiteness of the matrix Ξ . Thus, $(\pi_1 \hat{\mu}_1(x_1), \pi_2 \hat{\mu}_2(x_2))$ is the decentralized control strategy of the original interconnected system (10.2.43). Here, we apply the decentralized control scheme to plant (10.2.43) for 40 s and obtain the evolution processes of the state trajectories illustrated in Figs. 10.8 and 10.9. By zooming in on the state trajectories near zero, it is demonstrated that the state trajectories of the closed-loop system are UUB. Obviously, these simulation results authenticate the validity of the decentralized control approach developed in this section.

Example 10.2.2 Consider the classical multi-machine power system with governor controllers [6]

$$\begin{aligned}\dot{\delta}_i(t) &= \omega_i(t), \\ \dot{\omega}_i(t) &= -\frac{D_i}{2H_i}\omega_i(t) + \frac{\omega_0}{2H_i}[P_{mi}(t) - P_{ei}(t)], \\ \dot{P}_{mi}(t) &= \frac{1}{T_i}[-P_{mi}(t) + u_{gi}(t)], \\ P_{ei}(t) &= E'_{qi} \sum_{j=1}^N E'_{qj} [B_{ij} \sin \delta_{ij}(t) + G_{ij} \cos \delta_{ij}(t)],\end{aligned}$$

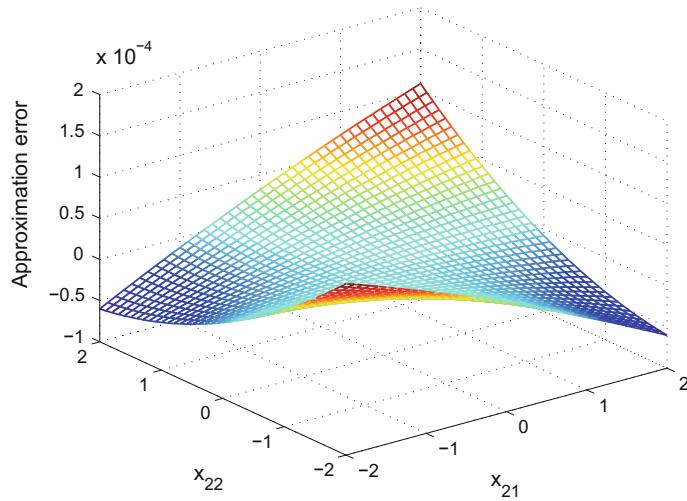


Fig. 10.6 3-D plot of the approximation error of the cost function of isolated subsystem 2, i.e., $J_2^*(x_2) - \hat{V}_2(x_2)$

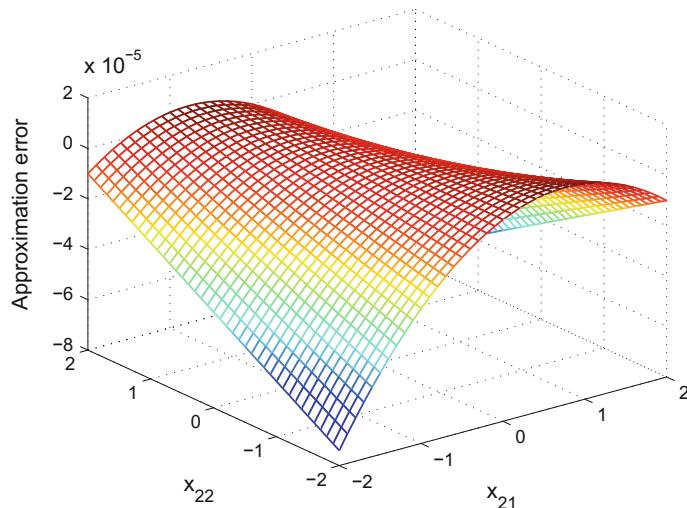


Fig. 10.7 3-D plot of the approximation error of the control law of isolated subsystem 2, i.e., $u_2^*(x_2) - \hat{\mu}_2(x_2)$

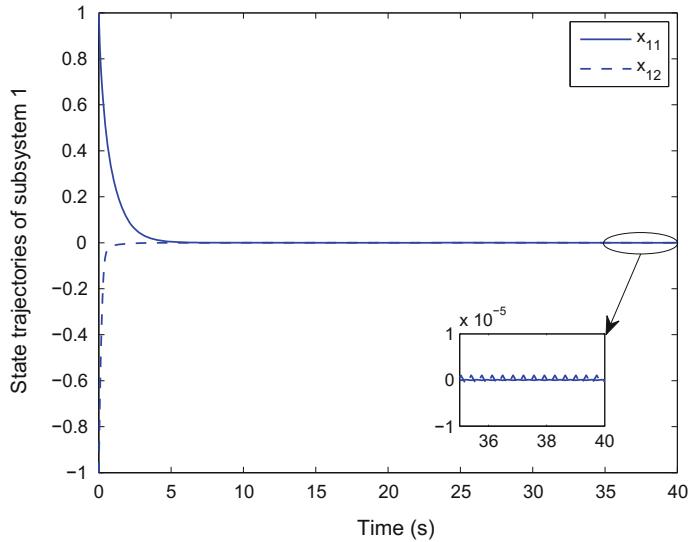


Fig. 10.8 The state trajectories of subsystem 1 under the action of the decentralized control strategy $(\pi_1 \hat{\mu}_1(x_1), \pi_2 \hat{\mu}_2(x_2))$

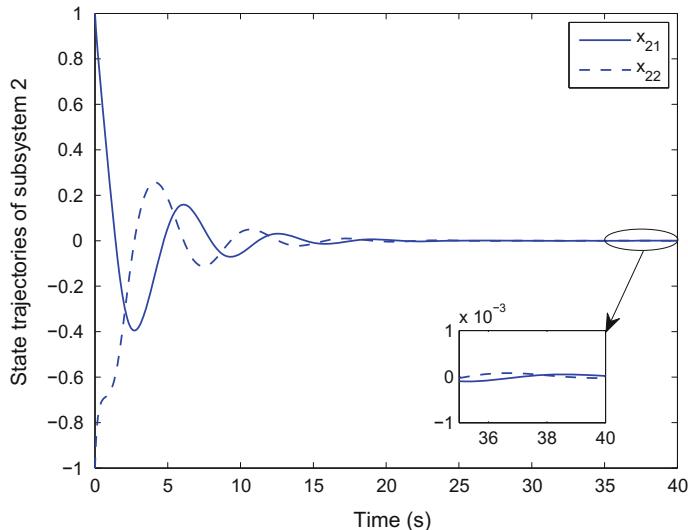


Fig. 10.9 The state trajectories of subsystem 2 under the action of the decentralized control strategy $(\pi_1 \hat{\mu}_1(x_1), \pi_2 \hat{\mu}_2(x_2))$

where, for $1 \leq i$ and $j \leq N$, $\delta_i(t)$ represents the angle of the i th generator; $\delta_{ij}(t) = \delta_i(t) - \delta_j(t)$ is the angular difference between the i th and j th generators; $\omega_i(t)$ is the relative rotor speed; $P_{mi}(t)$ and $P_{ei}(t)$ are the mechanical power and the electrical power, respectively; E'_{qi} is the transient electromotive force in quadrature axis and is assumed to be constant under high-gain SCR (silicon-controlled rectifier) controllers; D_i , H_i , and T_i are the damping constant, the inertia constant, and the governor time constant, respectively; B_{ij} and G_{ij} are the imaginary and real parts of the admittance matrix, respectively; $u_{gi}(t)$ is the speed governor control signal for the i th generator; and ω_0 is the steady-state frequency.

A three-machine power system is considered in our numerical simulation. The parameters of the system are the same as those in [6]. The weighting matrices are set to be $Q_i^2(x_i) = x_i^\top \times 1000I_3 \times x_i$ and $R_i = 1$, for $i = 1, 2, 3$. Similarly, as in [6], the multi-machine power system can be rewritten as the following form

$$\begin{aligned}\Delta \dot{\delta}_i(t) &= \Delta \omega_i(t), \\ \Delta \dot{\omega}_i(t) &= -\frac{D_i}{2H_i} \Delta \omega_i(t) + \frac{\omega_0}{2H_i} \Delta P_{mi}(t), \\ \Delta \dot{P}_{mi}(t) &= \frac{1}{T_i} [-\Delta P_{mi}(t) + u_i(t) - d_i(t)].\end{aligned}$$

We define the state

$$x_i = [\Delta \delta_i(t), \Delta \omega_i(t), \Delta P_{mi}(t)]^\top = [x_{i1}, x_{i2}, x_{i3}]^\top,$$

where

$$\Delta \delta_i(t) = \delta_i(t) - \delta_{i0},$$

$$\Delta \omega_i(t) = \omega_i(t) - \omega_{i0},$$

$$\Delta P_{mi}(t) = P_{mi}(t) - P_{ei}(t),$$

$$u_i(t) = u_{gi}(t) - P_{ei}(t),$$

and

$$d_i(t) = E'_{qi} \sum_{j=1, j \neq i}^N \{E'_{qj} [B_{ij} \cos \delta_{ij}(t) - G_{ij} \sin \delta_{ij}(t)] [\Delta \omega_i(t) - \Delta \omega_j(t)]\}.$$

For each isolated subsystem, we denote the weight vectors of the action and critic networks as

$$\hat{W}_a^i = \left[\hat{W}_{a1}^i, \hat{W}_{a2}^i, \hat{W}_{a3}^i \right]^\top,$$

$$\hat{W}_c^i = \left[\hat{W}_{c1}^i, \hat{W}_{c2}^i, \hat{W}_{c3}^i, \hat{W}_{c4}^i, \hat{W}_{c5}^i, \hat{W}_{c6}^i \right]^\top.$$

The activation functions are chosen as

$$\phi_a^i(x_i) = [x_{i1}, x_{i2}, x_{i3}]^\top$$

$$\phi_c^i(x_i) = [x_{i1}^2, x_{i1}x_{i2}, x_{i1}x_{i3}, x_{i2}^2, x_{i2}x_{i3}, x_{i3}^2]^\top.$$

From these parameters, $N_c^i = 6$ and $N_a^i = 3$. So, we conduct the simulation with $K_i = 10$. We set the initial state and the initial weights of the critic networks as $x_{i0} = [1, 1, 1]^\top$, $\hat{W}_c^i = 100 \times [1, 1, 1, 1, 1]^\top$, for $i = 1, 2, 3$. The initial weights of the action networks are chosen as $\hat{W}_a^1 = -[30, 30, 30]^\top$, $\hat{W}_a^2 = -[10, 20, 50]^\top$ and $\hat{W}_a^3 = -[10, 20, 30]^\top$, respectively. The period time $T = 0.1$ s and exploratory signals

$$\mathcal{N}_i(t) = 0.01(\sin(2\pi t) + \cos(2\pi t))$$

are used in the learning process. The least squares problem is solved after 10 samples are acquired, and thus, the weights of the NNs are updated every 1 s. Figs. 10.10, 10.11, and 10.12 illustrate the evolutions of the weights of the action network for the isolated subsystems 1, 2, and 3, respectively. It is clear that the weights converge after some update steps.

Then, we can choose $\pi_1 = \pi_2 = \pi_3 = 1$ to obtain a combined control vector $(\pi_1 \hat{\mu}_1(x_1), \pi_2 \hat{\mu}_2(x_2), \pi_3 \hat{\mu}_3(x_3))$, which can be regarded as the stabilizing decentralized control law of the interconnected system. By applying the decentralized control

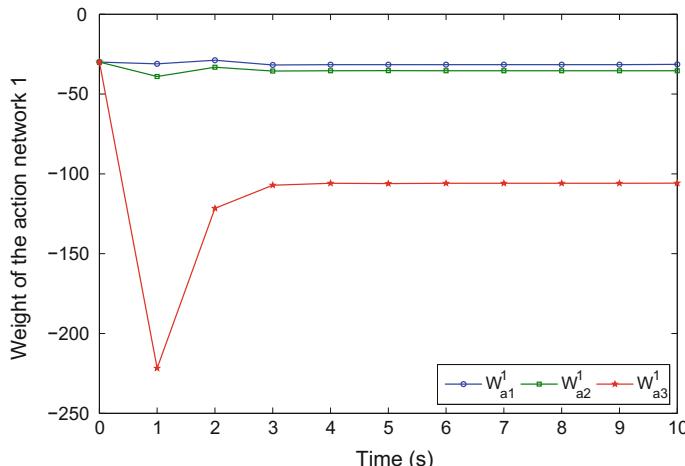


Fig. 10.10 Evolutions of the weights of the action network 1

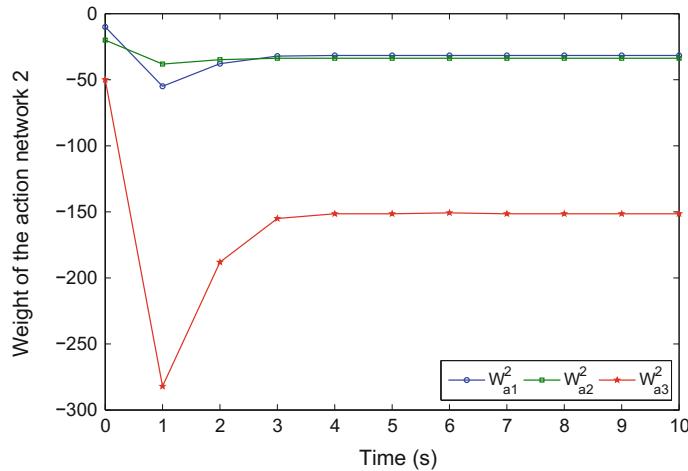


Fig. 10.11 Evolutions of the weights of the action network 2

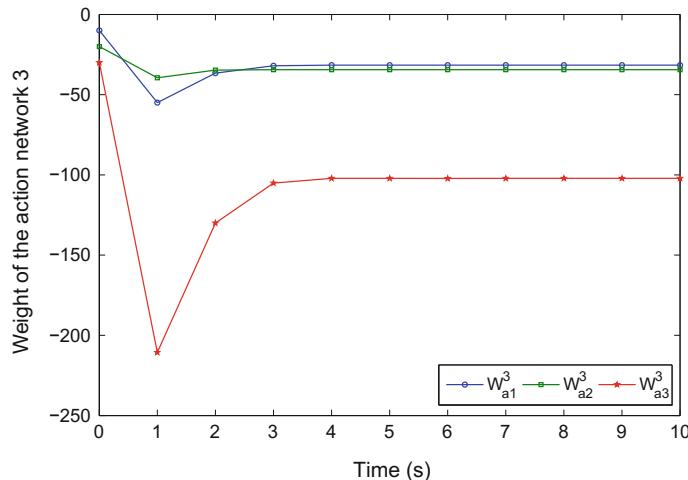


Fig. 10.12 Evolutions of the weights of the action network 3

law to the interconnected power system for 10 s, we obtain the evolution process of the power angle deviations and frequencies of the generators shown in Figs. 10.13 and 10.14, respectively. Obviously, the applicability of the decentralized control law developed in this section has been verified by these simulation results.

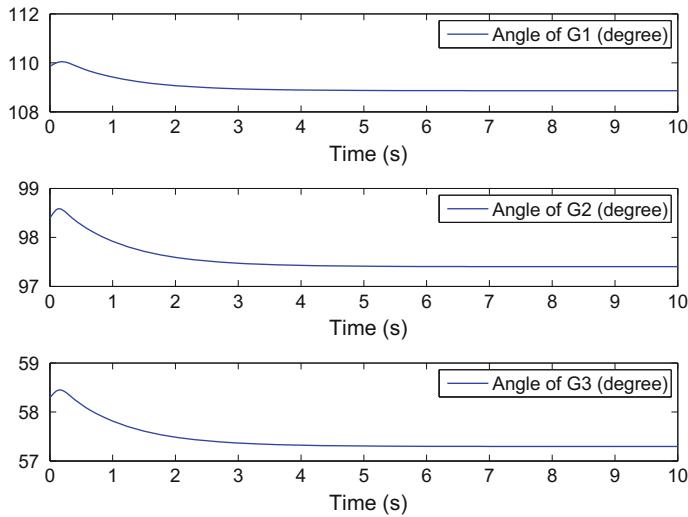


Fig. 10.13 Angles of the generators under the action of the decentralized control law

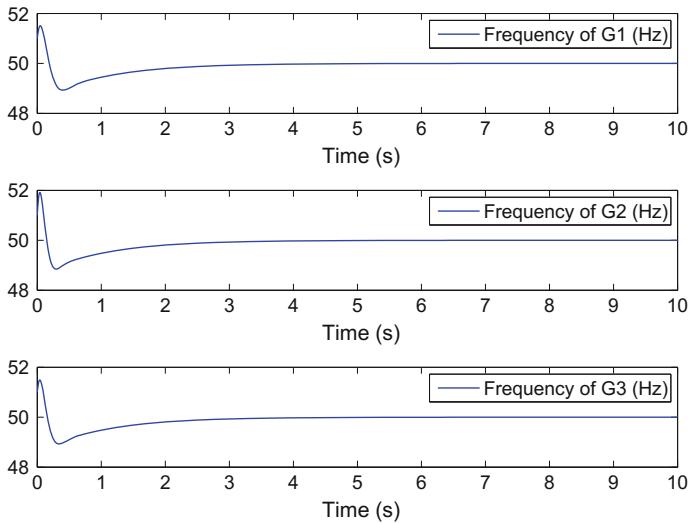


Fig. 10.14 Frequencies of the generators under the action of the decentralized control law

10.3 Conclusions

In this chapter, a decentralized control strategy is developed to deal with the stabilization problem of a class of continuous-time large-scale nonlinear systems using an online PI algorithm. It is shown that the decentralized control strategy of the overall system can be established by adding feedback gains to the obtained optimal control policies. Then, a stabilizing decentralized control law for a class of large-scale nonlinear systems with unknown dynamics is established by using an NN-based online model-free integral PI algorithm. We use an online model-free integral PI algorithm with an exploration to solve the HJB equations related to the optimal control problem of the isolated subsystems.

References

1. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791
2. Bakule L (2008) Decentralized control: an overview. *Ann Rev Control* 32(1):87–98
3. Beard RW, Sardis GN, Wen JT (1997) Galerkin approximations of the generalized Hamilton–Jacobi–Bellman equation. *Automatica* 33(12):2159–2177
4. Glad ST (1984) On the gain margin of nonlinear and optimal regulators. *IEEE Trans Autom Control* AC-29(7):615–620
5. Jiang Y, Jiang ZP (2012) Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica* 48(10):2699–2704
6. Jiang Y, Jiang ZP (2012) Robust adaptive dynamic programming for large-scale systems with an application to multimachine power systems. *IEEE Trans Circ Syst II Express Briefs* 59(10):693–697
7. Khan SG, Herrmann G, Lewis FL, Pipe T, Melhuish C (2012) Reinforcement learning and optimal adaptive control: an overview and implementation examples. *Ann Rev Control* 36(1):42–59
8. Lee JY, Park JB, Choi YH (2012) Integral Q-learning and explorized policy iteration for adaptive optimal control of continuous-time linear systems. *Automatica* 48(11):2850–2859
9. Lewis FL, Liu D (2012) Reinforcement learning and approximate dynamic programming for feedback control. Wiley, Hoboken
10. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circ Syst Mag* 9(3):32–50
11. Liang J, Venayagamoorthy GK, Harley RG (2012) Wide-area measurement based dynamic stochastic optimal power flow control for smart grids with high variability and uncertainty. *IEEE Trans Smart Grid* 3(1):59–69
12. Liu D, Wang D, Zhao D, Wei Q, Jin N (2012) Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming. *IEEE Trans Autom Sci Eng* 9(3):628–634
13. Liu D, Wang D, Li H (2014) Decentralized stabilization for a class of continuous-time nonlinear interconnected systems using online learning optimal control approach. *IEEE Trans Neural Network Learn Syst* 25(2):418–428
14. Liu D, Li C, Li H, Wang D, Ma H (2015) Neural-network-based decentralized control of continuous-time nonlinear interconnected systems with unknown dynamics. *Neurocomputing* 165:90–98
15. Mehraeen S, Jagannathan S (2011) Decentralized optimal control of a class of interconnected nonlinear discrete-time systems by using online Hamilton–Jacobi–Bellman formulation. *IEEE Trans Neural Network* 22(11):1757–1769

16. Park JW, Harley RG, Venayagamoorthy GK (2005) Decentralized optimal neuro-controllers for generation and transmission devices in an electric power network. *Eng Appl Artif Intell* 18(1):37–46
17. Rudin W (1976) Principles of mathematical analysis. McGraw-Hill, New York
18. Saberi A (1988) On optimality of decentralized control for a class of nonlinear interconnected systems. *Automatica* 24(1):101–104
19. Siljak DD (1991) Decentralized control of complex systems. Academic Press, Boston
20. Siljak DD, Zecevic AI (2005) Control of large-scale systems: beyond decentralized feedback. *Ann Rev Control* 29(2):169–179
21. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
22. Tsitsiklis JN, Athans M (1984) Guaranteed robustness properties of multivariable nonlinear stochastic optimal regulators. *IEEE Trans Autom Control* AC-29(8):690–696
23. Vamvoudakis KG, Lewis FL (2010) Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* 46(5):878–888
24. Vrabie D, Lewis FL (2009) Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Network* 22(3):237–246
25. Wang FY, Zhang H, Liu D (2009) Adaptive dynamic programming: an introduction. *IEEE Comput Intell Mag* 4(2):39–47
26. Wang D, Liu D, Li H, Ma H (2014) Neural-network-based robust optimal control design for a class of uncertain nonlinear systems via adaptive dynamic programming. *Inf Sci* 282:167–179
27. Werbos PJ (1977) Advanced forecasting methods for global crisis warning and models of intelligence. *General Syst Yearb* 22:25–38
28. Werbos PJ (1992) Approximate dynamic programming for real-time control and neural modeling. In: White DA, Sofge DA (eds) *Handbook of intelligent control: neural, fuzzy, and adaptive approaches* (Chapter 13). Van Nostrand Reinhold, New York
29. Zhang H, Liu D, Luo Y, Wang D (2013) *Adaptive dynamic programming for control: algorithms and stability*. Springer, London

Chapter 11

Learning Algorithms for Differential Games of Continuous-Time Systems

11.1 Introduction

Adaptive dynamic programming (ADP) [15, 16, 31, 33] has received significantly increasing attention owing to its self-learning capability. Value iteration-based ADP algorithms [6, 20, 32, 37, 40] can solve optimal control problems for discrete-time nonlinear systems without the requirement of complete knowledge of the dynamics by building a model neural network (NN). Policy iteration (PI)-based ADP algorithms can be used to solve optimal control problems of continuous-time nonlinear systems with the requirement of full knowledge [1, 24] or only partial knowledge [27, 30] of the dynamics. Some extended PI algorithms can solve optimal control problems of continuous-time nonlinear systems with completely unknown dynamics by building NN identifiers [10, 38] or with no NN identifiers [13, 14].

However, most of the previous works on ADP solving the optimal control problems assume that the system is only affected by a single control law. Game theory provides an ideal environment to study multi-player optimal decision and control problems. Two-player non-cooperative zero-sum game [9] has received much attention since it also provides the solution of H_∞ optimal control problems [8]. For a zero-sum game, it relies on solving the Hamilton–Jacobi–Isaacs (HJI) equation which reduces to solving the game algebraic Riccati equation (GARE) [9] when the system has linear dynamics and the cost function is quadratic. Different from the non-cooperative zero-sum game, nonzero-sum game offers a suitable theoretical method considering cooperative and non-cooperative objectives. For a multi-player nonzero-sum game, it will require to solve the coupled Hamilton–Jacobi (HJ) equations, which reduce to the coupled algebraic Riccati equations in the linear quadratic case. Generally speaking, both the HJI and coupled HJ equations cannot be solved analytically due to its nonlinear nature.

ADP algorithms have been used to solve the zero-sum game problems for discrete-time nonlinear systems [4, 5, 19]. For continuous-time linear systems, an online ADP algorithm was proposed for two-player zero-sum games without requiring the

knowledge of internal system dynamics [29]. For continuous-time nonlinear systems with constrained inputs, an offline PI scheme was proposed for two-player zero-sum games [2, 3]. In [39], four action networks and two critic networks were used to solve two-player zero-sum games without \mathcal{L}_2 gain condition. In [25], an online PI algorithm with two iterative loops was presented to solve two-player zero-sum games for continuous-time nonlinear systems with known dynamics. In [34, 35], an online simultaneous policy update algorithm with only one iterative loop was proposed for two-player zero-sum games by updating policies of both control and disturbance players simultaneously. Moreover, an online ADP approach was proposed to solve two-player nonzero-sum games of continuous-time linear systems without using complete information [28]. In [26], an online adaptive control algorithm based on PI was presented to solve multi-player nonzero-sum games of continuous-time nonlinear systems with known system dynamics. However, it is difficult for many practical problems to obtain the knowledge of the system dynamics.

In this chapter, we will develop ADP algorithms for differential games of continuous-time systems. First, an online integral PI algorithm is developed to learn the Nash equilibrium solution for two-player zero-sum linear differential games with completely unknown dynamics [17]. It results in a fully model-free method solving the GARE forward in time, where both internal and drift system dynamics are not required. The present algorithm updates value function, control policy, and disturbance policy simultaneously. Second, an iterative ADP algorithm is developed for multi-player zero-sum differential games of continuous-time uncertain nonlinear systems [18]. It is proved that the iterative value functions converge to the optimal solution if the optimal solution of the multi-player zero-sum differential games exists. By using NNs, the iterative control pairs can be obtained without knowing the system function where the stability and convergence properties are proved. Finally, an online synchronous approximate optimal learning algorithm based on PI is developed to solve the Nash equilibrium of multi-player nonzero-sum games with unknown nonlinear dynamics [21]. It is proved that the PI algorithm for nonzero-sum games with nonlinear dynamics is mathematically equivalent to the quasi-Newton's iteration. A model NN is established to identify the unknown continuous-time nonlinear systems using data measured along the system trajectories. For each player, a critic NN and an action NN are used to approximate its value function and control policy, respectively, and only critic NN weights need to be tuned. The uniform ultimate boundedness (UUB) of the closed-loop system is proved based on Lyapunov approach. Simulation examples are given to demonstrate the effectiveness of the present schemes.

11.2 Integral Policy Iteration for Two-Player Zero-Sum Games

Consider a class of continuous-time linear dynamical systems described by

$$\dot{x} = Ax + B_1u + B_2w, \quad (11.2.1)$$

where $x \in \mathbb{R}^n$ is the system state with initial state x_0 , $u \in \mathbb{R}^m$ is the control input, and $w \in \mathbb{R}^q$ is the external disturbance input with $w \in \mathcal{L}_2[0, \infty)$. $A \in \mathbb{R}^{n \times n}$, $B_1 \in \mathbb{R}^{n \times m}$, and $B_2 \in \mathbb{R}^{n \times q}$ are the unknown system matrices.

Define the infinite horizon performance index (or cost function)

$$\begin{aligned} J(x_0, u, w) &= \int_0^\infty (x^\top Qx + u^\top Ru - \gamma^2 w^\top w) d\tau \\ &\triangleq \int_0^\infty r(x, u, w) d\tau \end{aligned}$$

with $Q = Q^\top \geq 0$, $R = R^\top > 0$, and a predetermined constant $\gamma \geq \gamma^* \geq 0$, where γ^* denotes the smallest γ for which the system (11.2.1) is stable. For feedback control policy $u(x)$ and disturbance policy $w(x)$, we define the value function of the policies as

$$V(x) = \int_t^\infty (x^\top Qx + u^\top Ru - \gamma^2 w^\top w) d\tau. \quad (11.2.2)$$

Then, we define the two-player zero-sum differential game as

$$\begin{aligned} J^*(x_0) &= \min_u \max_w J(x_0, u, w) \\ &= \min_u \max_w \int_0^\infty (x^\top Qx + u^\top Ru - \gamma^2 w^\top w) d\tau, \end{aligned}$$

where the control policy player u seeks to minimize the value function while the disturbance policy player w desires to maximize it. The goal was to find the saddle point (u^*, w^*) which satisfies the following inequalities

$$J(x_0, u, w^*) \geq J(x_0, u^*, w^*) \geq J(x_0, u^*, w)$$

for any state feedback control policy u and disturbance policy w .

We use notations $u = -Kx$ and $w = Lx$ for the state feedback control policy and the disturbance policy, respectively. Then, the value function (11.2.2) can be represented as $V(x) = x^\top Px$, where the matrix P is determined by K and L . The saddle point can be obtained by solving the following continuous-time GARE [9]

$$A^\top P^* + P^* A + Q - P^* B_1 R^{-1} B_1^\top P^* + \gamma^{-2} P^* B_2 B_2^\top P^* = 0. \quad (11.2.3)$$

Defining P^* as the unique positive definite solution of (11.2.3), the saddle point of the zero-sum game is

$$\begin{aligned} u^* &= -K^*x = -R^{-1}B_1^T P^*x, \\ w^* &= L^*x = \gamma^{-2}B_2^T P^*x, \end{aligned} \quad (11.2.4)$$

and the optimal game value function is

$$V^*(x_0) = x_0^T P^* x_0.$$

The solution of the H_∞ control problem can be obtained by solving the saddle point of the equivalent two-player zero-sum game problem. The following H_∞ norm (the \mathcal{L}_2 -gain) is used to measure the performance of the control system.

Definition 11.2.1 Let $\gamma \geq 0$ be certain prescribed level of disturbance attenuation. The system (11.2.1) is said to have \mathcal{L}_2 -gain less than or equal to γ if

$$\int_0^\infty (x^T Q x + u^T R u) d\tau \leq \gamma^2 \int_0^\infty (w^T w) d\tau \quad (11.2.5)$$

for all $w \in \mathcal{L}_2[0, \infty)$.

The H_∞ control problem is to find a state feedback control policy u such that the closed-loop system is stable and satisfies the condition (11.2.5). For every $\gamma \geq \gamma^*$, the GARE has a unique positive definite solution [9].

A stabilizing control policy exists with the following standard assumption.

Assumption 11.2.1 The pair (A, B_1) is stabilizable and the pair $(A, Q^{1/2})$ is observable.

11.2.1 Derivation of Integral Policy Iteration

In this section, we will develop an online model-free integral PI algorithm for the linear continuous-time zero-sum differential game with completely unknown dynamics. First, we assume an initial stabilizing control matrix K_1 to be given. Define $V_i(x) = x^T P_i x$, $u_i(x) = -K_i x$, and $w_i(x) = L_i x$ as the value function, control policy, and disturbance policy, respectively, for each iterative step $i \geq 0$.

To relax the assumptions of exact knowledge of A , B_1 , and B_2 , we use \mathcal{N}_1 and \mathcal{N}_2 to denote the small exploratory signals added to the control policy u_i and disturbance policy w_i , respectively. The exploratory signals are assumed to be any nonzero measurable signal which is bounded by $e_M > 0$, i.e., $\|\mathcal{N}_1\| \leq e_M$, $\|\mathcal{N}_2\| \leq e_M$. Then, the original system (11.2.1) becomes

$$\dot{x} = Ax + B_1(u_i + \mathcal{N}_1) + B_2(w_i + \mathcal{N}_2). \quad (11.2.6)$$

The derivative of the value function with respect to time is calculated as

$$\dot{V}_i(x) = -x^T Q x - x^T K_i^T R K_i x + \gamma^2 x^T L_i^T L_i x + 2x^T K_{i+1}^T R \mathcal{N}_1 \\ + 2\gamma^2 x^T L_{i+1}^T \mathcal{N}_2, \quad (11.2.7)$$

where we have used $K_{i+1} = R^{-1} B_1^T P_i$ and $L_{i+1} = \gamma^{-2} B_2^T P_i$ according to (11.2.4). Integrating (11.2.7) from t and $t + T$ with any time interval $T > 0$, we have

$$x_{t+T}^T P_i x_{t+T} - x_t^T P_i x_t = - \int_t^{t+T} r(x, u_i, w_i) d\tau + 2 \int_t^{t+T} x^T K_{i+1}^T R \mathcal{N}_1 d\tau \\ + 2\gamma^2 \int_t^{t+T} x^T L_{i+1}^T \mathcal{N}_2 d\tau,$$

where the values of the state x at time t and $t + T$ are denoted by x_t and x_{t+T} . Therefore, we obtain the online model-free integral PI algorithm (Algorithm 11.2.1) for zero-sum differential games.

Algorithm 11.2.1 Online model-free integral PI for zero-sum games

- Step 1. Give an initial stabilizing policy $u_1 = -K_1 x$ and $w_1 = L_1 x$. Set $i = 1$ and $P_0 = 0$.
 Step 2. Policy Evaluation and Improvement: For the system (11.2.6) with policies $u_i = -K_i x$ and $w_i = L_i x$, and exploratory signals \mathcal{N}_1 and \mathcal{N}_2 , solve the following equation for P_i , K_{i+1} and L_{i+1}

$$x_t^T P_i x_t = x_{t+T}^T P_i x_{t+T} + \int_t^{t+T} r(x, u_i, w_i) d\tau - 2 \int_t^{t+T} x^T K_{i+1}^T R \mathcal{N}_1 d\tau \\ - 2\gamma^2 \int_t^{t+T} x^T L_{i+1}^T \mathcal{N}_2 d\tau. \quad (11.2.8)$$

- Step 3. If $\|P_i - P_{i-1}\| \leq \xi$ (ξ is a prescribed small positive real number), stop and return P_i ; else, set $i = i + 1$ and go to Step 2.
-

Remark 11.2.1 Equation (11.2.8) plays an important role in relaxing the assumption of the knowledge of system dynamics, since A , B_1 , and B_2 do not appear in (11.2.8). Only online data measured along the system trajectories are required to run this algorithm. Our method avoids the identification of A , B_1 , and B_2 whose information is embedded in the online measured data. In other words, the lack of knowledge about the system dynamics does not have any impact on our method to obtain the Nash equilibrium. Thus, our method will not be affected by the errors between the identification model and the real system, and it can respond fast to the changes of the system dynamics.

Remark 11.2.2 This algorithm is actually a PI method, but the policy evaluation and policy improvement are performed at the same time. Compared with the model-based method [25] and partially model-free method [35], our algorithm is a fully model-free method which does not require knowledge of the system dynamics. Different from the iterative method with inner loop on disturbance policy and outer loop on

control policy [25], and the method with only one iterative loop by updating control and disturbance policies simultaneously [35], the method developed here updates the value function, control, and disturbance policies at the same time.

Remark 11.2.3 Small exploratory signals can satisfy the persistence of excitation (PE) condition to efficiently update the value function and the policies. To guarantee the PE condition, the state may need to be reset during the iterative process, but it results in technical problems for stability analysis of the closed-loop system. An alternative way is to add exploratory signals. The solution obtained by our method is exactly the same as the one determined by the GARE by considering the effects of exploratory signals.

Next, we will show the relationship between the present algorithm and the Q-learning algorithm by extending the concept of Q-function to zero-sum games that are continuous in time, state, and action space. The optimal continuous-time Q-function for zero-sum games is defined as the following quadratic form:

$$\begin{aligned}\mathbf{Q}^*(x, u, w) &= [x^T u^T w^T] H^* [x^T u^T w^T]^T \\ &= [x^T u^T w^T] \begin{bmatrix} H_{11}^* & H_{12}^* & H_{13}^* \\ H_{21}^* & H_{22}^* & H_{23}^* \\ H_{31}^* & H_{32}^* & H_{33}^* \end{bmatrix} \begin{bmatrix} x \\ u \\ w \end{bmatrix} \\ &= [x^T u^T w^T] \begin{bmatrix} A^T P^* + P^* A + Q & P^* B_1 & P^* B_2 \\ B_1^T P^* & R & 0 \\ B_2^T P^* & 0 & -\gamma^2 \end{bmatrix} \begin{bmatrix} x \\ u \\ w \end{bmatrix}. \quad (11.2.9)\end{aligned}$$

It can be seen that the matrix H^* is associated with P^* in GARE. By solving

$$\nabla_u \mathbf{Q}^*(x, u, w) = 0 \text{ and } \nabla_w \mathbf{Q}^*(x, u, w) = 0,$$

we can obtain

$$\begin{aligned}u^* &= -(H_{22}^*)^{-1} (H_{12}^*)^T x, \\ w^* &= -(H_{33}^*)^{-1} (H_{13}^*)^T x,\end{aligned}$$

which are the same as the equations in (11.2.4). Since

$$\mathbf{Q}^*(x_0, u^*, w^*) = V^*(x_0),$$

the relationship between P^* and H^* can be represented as follows:

$$P^* = [I_n \ -K^T \ L^T] H^* \begin{bmatrix} I_n \\ -K \\ L \end{bmatrix}.$$

According to (11.2.9), we can obtain

$$H_{11}^* = H_{12}^*(H_{22}^*)^{-1}H_{21}^* + H_{13}^*(H_{33}^*)^{-1}H_{31}^*,$$

and thus H_{11}^* is a redundant term. Define

$$H^i = \begin{bmatrix} H_{11}^i & H_{12}^i & H_{13}^i \\ H_{21}^i & R & 0 \\ H_{31}^i & 0 & -\gamma^2 \end{bmatrix},$$

where

$$H_{21}^i = (H_{12}^i)^\top \text{ and } H_{31}^i = (H_{13}^i)^\top.$$

Now, we are in a position to develop the next online integral Q-learning algorithm (Algorithm 11.2.2).

Algorithm 11.2.2 Online integral Q-learning for zero-sum games

Step 1. Give an initial stabilizing policy $u_1 = -K_1 x$ and $w_1 = L_1 x$.

Step 2. Set $i = 0$, $H_{11}^0 = 0$, $H_{12}^0 = K_1^\top R$, and $H_{13}^0 = \gamma^2 L_1^\top$.

Step 3. Policy Evaluation: Let $i = i + 1$. For the system (11.2.6) with policies $u_i = -K_i x$ and $w_i = L_i x$, and exploratory signals \mathcal{N}_1 and \mathcal{N}_2 , solve the following equation for H_{11}^i , H_{12}^i and H_{13}^i

$$\begin{aligned} x_t^\top H_{11}^i x_t &= x_{t+T}^\top H_{11}^i x_{t+T} + \int_t^{t+T} r(x, u_i, w_i) d\tau - 2 \int_t^{t+T} x^\top H_{12}^i \mathcal{N}_1 d\tau \\ &\quad - 2 \int_t^{t+T} x^\top H_{13}^i \mathcal{N}_2 d\tau. \end{aligned} \quad (11.2.10)$$

Step 4. Policy Improvement: Update the following parameters

$$K_{i+1} = R^{-1}(H_{12}^i)^\top, \quad L_{i+1} = \gamma^{-2}(H_{13}^i)^\top.$$

Step 5. If $\|H_{11}^i - H_{11}^{i-1}\| + \|H_{12}^i - H_{12}^{i-1}\| + \|H_{13}^i - H_{13}^{i-1}\| \leq \zeta$ (ζ is a prescribed small positive real number), stop and return H_i ; else, go to Step 3.

Remark 11.2.4 Note that the model-free integral PI algorithm (Algorithm 11.2.1) is equivalent to the integral Q-learning algorithm (Algorithm 11.2.2) for zero-sum games. As PI methods, the algorithms developed above require an initial stabilizing control policy which is usually obtained by experience. We can also obtain $B_1 = (H_{11}^i)^{-1}H_{12}^i$ and $B_2 = (H_{11}^i)^{-1}H_{13}^i$.

11.2.2 Convergence Analysis

In this section, we will provide a convergence analysis of the present algorithms for two-player zero-sum differential games. It can be shown that the present model-free integral PI and Q-learning algorithms are equivalent to Newton's method.

Theorem 11.2.1 For an initial stabilizing control policy $u_1 = -K_1 x$, the sequences of $\{P_i\}_{i=1}^{\infty}$, $\{K_i\}_{i=1}^{\infty}$, and $\{L_i\}_{i=1}^{\infty}$ obtained by solving (11.2.8) in Algorithm 11.2.1 converge to the optimal solution P^* of GARE, the saddle point K^* , and L^* , respectively, as $i \rightarrow \infty$.

Proof First, for an initial stabilizing control policy $u_1 = -K_1 x$, we can prove that the present Algorithm 11.2.1 is equivalent to the following Lyapunov equation

$$A_i^T P_i + P_i A_i = -M_i, \quad (11.2.11)$$

where

$$A_i = A - B_1 K_i + B_2 L_i, \quad M_i = Q + K_i^T R K_i - \gamma^2 L_i^T L_i.$$

With the control policy $u_i = -K_i x$, the disturbance policy $w_i = L_i x$, and the exploratory signals \mathcal{N}_1 and \mathcal{N}_2 , the closed-loop system (11.2.1) becomes

$$\dot{x} = A_i x + B_1 \mathcal{N}_1 + B_2 \mathcal{N}_2,$$

where $A_i = A - B_1 K_i + B_2 L_i$. Considering the Lyapunov function $V_i(x) = x^T P_i x$, its derivative can be calculated as

$$\begin{aligned} \dot{V}_i(x) &= \dot{x}^T P_i x + x^T P_i \dot{x} \\ &= x^T A_i^T P_i x + x^T P_i A_i x + (B_1 \mathcal{N}_1 + B_2 \mathcal{N}_2)^T P_i x + x^T P_i (B_1 \mathcal{N}_1 + B_2 \mathcal{N}_2) \\ &= x^T (A_i^T P_i + P_i A_i) x + 2x^T P_i B_1 \mathcal{N}_1 + 2x^T P_i B_2 \mathcal{N}_2 \\ &= x^T (A_i^T P_i + P_i A_i) x + 2x^T K_{i+1}^T R \mathcal{N}_1 + 2\gamma^2 x^T L_{i+1}^T \mathcal{N}_2. \end{aligned} \quad (11.2.12)$$

Integrating (11.2.12) from t to $t+T$ yields

$$\begin{aligned} V_i(x_{t+T}) - V_i(x_t) &= \int_t^{t+T} x^T (A_i^T P_i + P_i A_i) x \, d\tau + 2 \int_t^{t+T} x^T K_{i+1}^T R \mathcal{N}_1 \, d\tau \\ &\quad + 2 \int_t^{t+T} \gamma^2 x^T L_{i+1}^T \mathcal{N}_2 \, d\tau. \end{aligned} \quad (11.2.13)$$

According to (11.2.8), we have

$$\begin{aligned} V_i(x_{t+T}) - V_i(x_t) &= - \int_t^{t+T} r(x, u_i, w_i) \, d\tau + 2 \int_t^{t+T} x^T K_{i+1}^T R \mathcal{N}_1 \, d\tau \\ &\quad + 2 \int_t^{t+T} \gamma^2 x^T L_{i+1}^T \mathcal{N}_2 \, d\tau. \end{aligned} \quad (11.2.14)$$

Therefore, combining (11.2.13) and (11.2.14), we have

$$\begin{aligned} x^\top (A_i^\top P_i + P_i A_i)x &= -r(x, u_i, w_i) \\ &= -x^\top (Q + K_i^\top R K_i - \gamma^2 L_i^\top L_i)x, \end{aligned}$$

i.e.,

$$A_i^\top P_i + P_i A_i = -M_i,$$

where

$$M_i = Q + K_i^\top R K_i - \gamma^2 L_i^\top L_i.$$

Then, according to the results in [35], the sequence $\{P_i\}_{i=1}^\infty$ generated by (11.2.11) is equivalent to Newton's method and converges to the optimal solution P^* of GARE, as $i \rightarrow \infty$. Furthermore, the sequences $\{K_i\}_{i=1}^\infty$ and $\{L_i\}_{i=1}^\infty$ converge to the saddle point K^* and L^* , respectively, as $i \rightarrow \infty$. This completes the proof of the theorem.

The next theorem will show the convergence of the model-free integral Q-learning algorithm for zero-sum games.

Theorem 11.2.2 *For an initial stabilizing control policy $u_1 = -K_1 x$, the sequence $\{H^i\}_{i=1}^\infty$ obtained by solving (11.2.10) in Algorithm 11.2.2 converges to the optimal solution H^* ; i.e., Q^* can be achieved, as $i \rightarrow \infty$.*

Proof Because the iteration process of H^i with solving (11.2.10) in Algorithm 11.2.2 is equivalent to that of P_i with solving (11.2.8) in Algorithm 11.2.1, then as $i \rightarrow \infty$

$$H^i \rightarrow \begin{bmatrix} A^\top P^* + P^* A + Q & P^* B_1 & P^* B_2 \\ B_1^\top P^* & R & 0 \\ B_2^\top P^* & 0 & -\gamma^2 \end{bmatrix} = H^*.$$

The proof is complete.

11.2.3 Neural Network Implementation

In this section, an online implementation of Algorithm 11.2.1 is developed based on ADP with the least squares method. Algorithm 11.2.2 can be implemented in the same way. Here parametric structures are used to approximate the game value function, control policy, and disturbance policy.

Given a stabilizing control policy $u_i = -K_i x$, a triplet (P_i, K_{i+1}, L_{i+1}) with $P_i = P_i^\top > 0$, can uniquely be determined by (11.2.8). We now define the following two operators: $P \in \mathbb{R}^{n \times n} \rightarrow \hat{P} \in \mathbb{R}^{\frac{1}{2}n(n+1)}$, $x \in \mathbb{R}^n \rightarrow \bar{x} \in \mathbb{R}^{\frac{1}{2}n(n+1)}$, where

$$\begin{aligned} \hat{P} &= [p_{11}, 2p_{12}, \dots, 2p_{1n}, p_{22}, 2p_{23}, \dots, 2p_{(n-1)n}, p_{nn}]^\top, \\ \bar{x} &= [x_1^2, x_1 x_2, \dots, x_1 x_n, x_2^2, x_2 x_3, \dots, x_{n-1} x_n, x_n^2]^\top. \end{aligned}$$

Hence,

$$x_{t+(k-1)T}^T P_i x_{t+(k-1)T} - x_{t+kT}^T P_i x_{t+kT} = (\bar{x}_{t+(k-1)T} - \bar{x}_{t+kT})^T \hat{P}_i,$$

where $k \in \mathbb{Z}^+$ and $k \geq 1$. Using Kronecker product \otimes , we obtain

$$\begin{aligned} x^T K_{i+1}^T R \mathcal{N}_1 &= (x \otimes \mathcal{N}_1)^T (I_n \otimes R) \text{vec}(K_{i+1}) \\ x^T L_{i+1}^T \mathcal{N}_2 &= (x \otimes \mathcal{N}_2)^T \text{vec}(L_{i+1}). \end{aligned}$$

Using the expressions established above, (11.2.8) can be rewritten in a general compact form as

$$\psi_k^T \begin{bmatrix} \hat{P}_i \\ \text{vec}(K_{i+1}) \\ \text{vec}(L_{i+1}) \end{bmatrix} = \theta_k, \quad \forall i \in \mathbb{Z}^+, \quad (11.2.15)$$

where

$$\begin{aligned} \theta_k &= \int_{t+(k-1)T}^{t+kT} r(x, u_i, w_i) d\tau, \\ \psi_k &= \left[(\bar{x}_{t+(k-1)T} - \bar{x}_{t+kT})^T, 2 \int_{t+(k-1)T}^{t+kT} (x \otimes \mathcal{N}_1)^T d\tau (I_n \otimes R), \right. \\ &\quad \left. 2\gamma^2 \int_{t+(k-1)T}^{t+kT} (x \otimes \mathcal{N}_2)^T d\tau \right]^T, \end{aligned}$$

and the measurement time is from $t + (k - 1)T$ to $t + kT$. Since (11.2.15) is only one-dimensional equation, we cannot guarantee the uniqueness of the solution. We will use the least squares method to solve this problem, where the parameter vector is determined in a least squares sense over a compact set Ω .

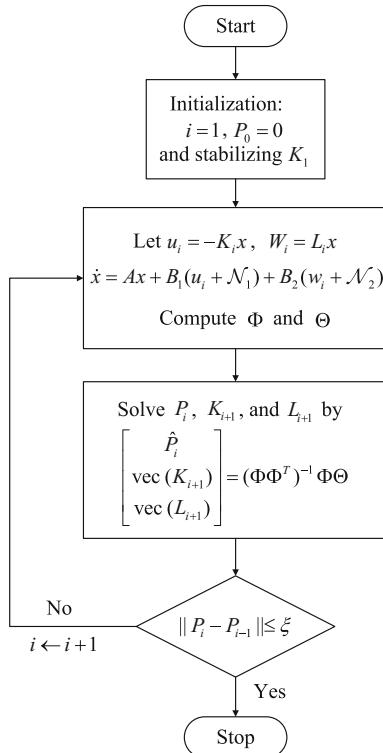
For any positive integer N , denote $\Phi = [\psi_1, \dots, \psi_N]$ and $\Theta = [\theta_1, \dots, \theta_N]^T$. Then, we have the following N -dimensional equation

$$\Phi^T \begin{bmatrix} \hat{P}_i \\ \text{vec}(K_{i+1}) \\ \text{vec}(L_{i+1}) \end{bmatrix} = \Theta, \quad \forall i \in \mathbb{Z}^+.$$

If Φ^T has full column rank, the parameters can be solved by

$$\begin{bmatrix} \hat{P}_i \\ \text{vec}(K_{i+1}) \\ \text{vec}(L_{i+1}) \end{bmatrix} = (\Phi \Phi^T)^{-1} \Phi \Theta. \quad (11.2.16)$$

Fig. 11.1 Flowchart of Algorithm 11.2.1



Therefore, we need to have the number of collected data points N to be at least

$$N_{\min} = \text{rank}(\Phi),$$

i.e.,

$$N_{\min} = \frac{n(n+1)}{2} + nm + nq,$$

which will guarantee the existence of $(\Phi\Phi^T)^{-1}$.

The least squares problem in (11.2.16) can be solved in real time after collecting enough data points generated from system (11.2.6). The flowchart of this algorithm is shown in Fig. 11.1. The solution can be obtained using the batch least squares, the recursive least squares algorithms, or the gradient descent algorithms.

Remark 11.2.5 The sequence $\{P_i\}_{i=1}^{\infty}$ calculated by the least squares method converges to the solution of GARE. The PE condition is required in adaptive control to perform system identification. Several types of exploratory signals have been used for that purpose, such as piecewise constant signals [14], sinusoidal signals with dif-

ferent frequencies [13], random noise [4, 39], and exponentially decreasing probing noise [25].

11.2.4 Simulation Studies

In this section, we demonstrate the effectiveness of the present algorithm by designing an H_∞ state feedback controller for a power system.

Example 11.2.1 Consider the following linear model of a power system that was studied in [29]

$$\begin{aligned} \dot{x} &= Ax + B_1 u + B_2 w \\ &= \begin{bmatrix} -0.0665 & 8 & 0 & 0 \\ 0 & -3.663 & 3.663 & 0 \\ -6.86 & 0 & -13.736 & -13.736 \\ 0.6 & 0 & 0 & 0 \end{bmatrix} x \\ &\quad + \begin{bmatrix} 0 \\ 0 \\ 13.736 \\ 0 \end{bmatrix} u + \begin{bmatrix} -8 \\ 0 \\ 0 \\ 0 \end{bmatrix} w, \end{aligned} \quad (11.2.17)$$

where the state vector is

$$x = [\Delta f, \Delta P_g, \Delta X_g, \Delta E]^\top,$$

Δf (Hz) is the incremental frequency deviation, ΔP_g (p.u. MW) is the incremental change in generator output, ΔX_g (p.u. MW) is the incremental change in governor value position, and ΔE is the incremental change in integral control. We assume that the dynamics of system (11.2.17) is unknown. The matrices Q and R in the cost function are identity matrices of appropriate dimensions, and $\gamma = 3.5$. Using the system model (11.2.17), the matrix in the optimal value function of the zero-sum game is

$$P^* = \begin{bmatrix} 0.8335 & 0.9649 & 0.1379 & 0.8005 \\ 0.9649 & 1.4751 & 0.2358 & 0.8046 \\ 0.1379 & 0.2358 & 0.0696 & 0.0955 \\ 0.8005 & 0.8046 & 0.0955 & 2.6716 \end{bmatrix}.$$

Now we will use the present online model-free integral PI algorithm to solve this problem. The initial state is selected as $x_0 = [0.1, 0.2, 0.2, 0.1]^\top$. The simulation is conducted using data obtained along the system trajectory at every 0.01 s. The least squares problem is solved after 50 data samples are acquired, and thus, the parameters of the control policy are updated every 0.5 s. The parameters of the critic network, the

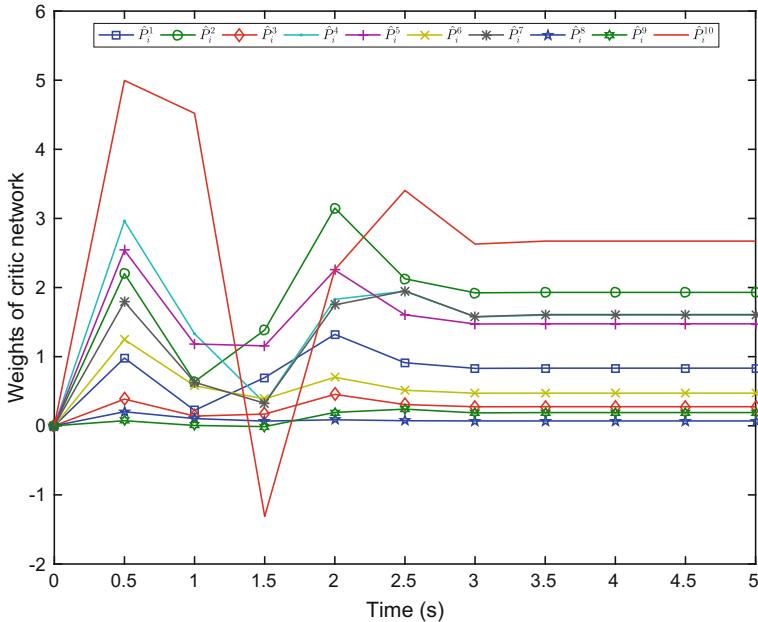


Fig. 11.2 Convergence of the game value function matrix \hat{P}_i

control action network, and the disturbance action network are all initialized to zero. Similar to [39], the PE condition is ensured by adding small zero-mean Gaussian noises with variances to the control and disturbance inputs.

Figure 11.2 presents the evolution of the parameters of the game value function during the learning process. It is clear that Algorithm 11.2.1 is convergent after 10 iterative steps. The obtained approximate game value function is given by the matrix

$$P_{10} = \begin{bmatrix} 0.8335 & 0.9649 & 0.1379 & 0.8005 \\ 0.9649 & 1.4752 & 0.2359 & 0.8047 \\ 0.1379 & 0.2359 & 0.0696 & 0.0956 \\ 0.8005 & 0.8047 & 0.0956 & 2.6718 \end{bmatrix},$$

and $\|P_{10} - P^*\| = 2.9375 \times 10^{-4}$. We can find that the solution obtained by the online model-free integral PI algorithm is quite close to the exact one obtained by solving GARE. Figures 11.3 and 11.4 show the convergence process of the control and disturbance action network parameters. The obtained H_∞ state feedback control policy is $u_{11} = -[1.8941, 3.2397, 0.9563, 1.3126]x$.

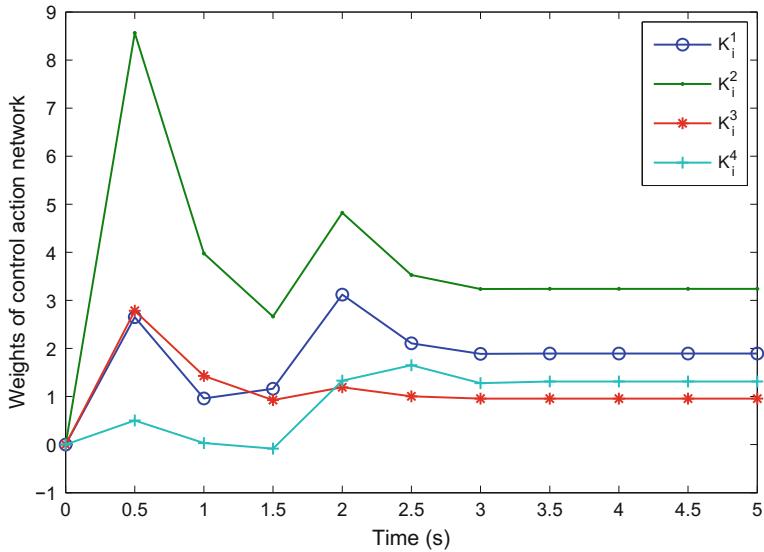


Fig. 11.3 Convergence of the control action network parameters K_i

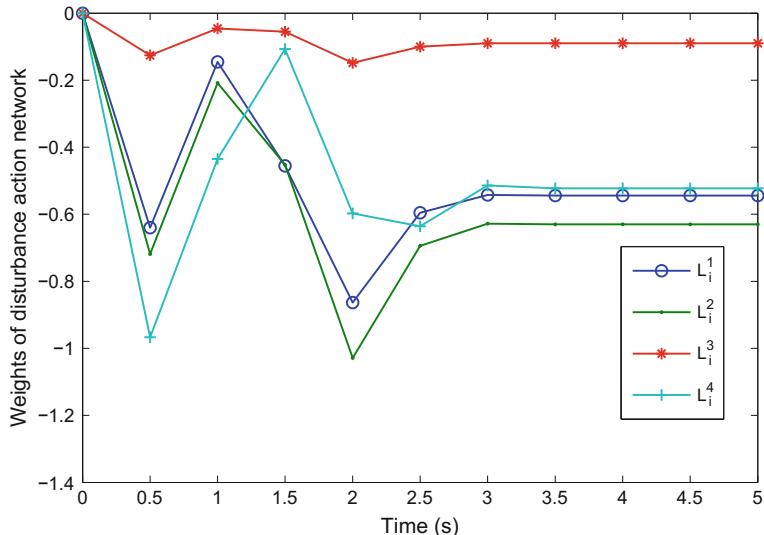


Fig. 11.4 Convergence of the disturbance action network parameters L_i

11.3 Iterative Adaptive Dynamic Programming for Multi-player Zero-Sum Games

Consider the following multi-player zero-sum differential game. The state trajectory at time t of the game denoted by $x = x(t)$ is described by the following continuous-time affine uncertain nonlinear function

$$\begin{aligned}\dot{x} &= f(x, u_1, u_2, \dots, u_p, w_1, w_2, \dots, w_q) \\ &= a(x) + b_1(x)u_1 + b_2(x)u_2 + \dots + b_p(x)u_p \\ &\quad + c_1(x)w_1 + c_2(x)w_2 + \dots + c_q(x)w_q,\end{aligned}\quad (11.3.1)$$

where $a(x)$, $b_k(x)$, $k = 1, \dots, p$, and $c_j(x)$, $j = 1, \dots, q$, are unknown system functions, $p, q > 0$ are positive integers. Let $x \in \mathbb{R}^n$ be the system state. Let $u_k \in \mathbb{R}^{n_k}$ and $w_j \in \mathbb{R}^{m_j}$ be the controls, where n_k , $k = 1, \dots, p$, and m_j , $j = 1, \dots, q$, are positive integers. The initial condition $x(0) = x_0$ is given. The value function is a generalized quadratic form given by

$$\begin{aligned}V(x(0), u_1, \dots, u_p, w_1, \dots, w_q) &= \int_0^\infty (x^\top A x + u_1^\top B_1 u_1 + u_2^\top B_2 u_2 + \dots + u_p^\top B_p u_p \\ &\quad + w_1^\top C_1 w_1 + w_2^\top C_2 w_2 + \dots + w_q^\top C_q w_q) dt,\end{aligned}\quad (11.3.2)$$

where A, B_k, C_j , $k = 1, 2, \dots, p$, $j = 1, 2, \dots, q$, are matrices with suitable dimensions, and $A > 0$, $B_k > 0$, $C_j < 0$. We assume that $\forall t \in [0, \infty)$, the value function $V(x(t), u_1, u_2, \dots, u_p, w_1, w_2, \dots, w_q)$ (denoted by $V(x)$ for brevity) is rigorously convex for all u_k , $k = 1, 2, \dots, p$ and concave for all w_j , $j = 1, 2, \dots, q$. Define the quadratic utility function as

$$\begin{aligned}l(x, u_1, u_2, \dots, u_p, w_1, w_2, \dots, w_q) &= x^\top A x + u_1^\top B_1 u_1 + \dots + u_p^\top B_p u_p \\ &\quad + w_1^\top C_1 w_1 + \dots + w_q^\top C_q w_q.\end{aligned}$$

For the above multi-player zero-sum differential game, there are two groups of controllers or players where group I (including u_1, u_2, \dots, u_p) tries to minimize the value function $V(x)$, while group II (including w_1, w_2, \dots, w_q) attempts to maximize it. According to the situation of the two groups, we have the following definitions.

Let

$$\bar{V}(x) \triangleq \inf_{u_1} \dots \inf_{u_p} \sup_{w_1} \dots \sup_{w_q} \{V(x(t), u_1, \dots, u_p, w_1, \dots, w_q)\} \quad (11.3.3)$$

be the upper value function and

$$\underline{V}(x) \triangleq \sup_{w_1} \dots \sup_{w_q} \inf_{u_1} \dots \inf_{u_p} \{V(x(t), u_1, \dots, u_p, w_1, \dots, w_q)\} \quad (11.3.4)$$

be the lower value function with the obvious inequality $\bar{V}(x) \geq \underline{V}(x)$ (see [9] for details). Define optimal control vectors as $(\bar{u}_1, \bar{u}_2, \dots, \bar{u}_p, \bar{w}_1, \bar{w}_2, \dots, \bar{w}_q)$ and $(\underline{u}_1, \underline{u}_2, \dots, \underline{u}_p, \underline{w}_1, \underline{w}_2, \dots, \underline{w}_q)$ for upper and lower value functions, respectively. Then,

$$\bar{V}(x) = V(x, \bar{u}_1, \dots, \bar{u}_p, \bar{w}_1, \dots, \bar{w}_q),$$

and

$$\underline{V}(x) = V(x, \underline{u}_1, \dots, \underline{u}_p, \underline{w}_1, \dots, \underline{w}_q).$$

If both $\bar{V}(x)$ and $\underline{V}(x)$ exist, and

$$\bar{V}(x) = \underline{V}(x) = V^*(x)$$

holds, then the optimal value function of the zero-sum differential game or the optimal solution exists and the corresponding optimal control vector is denoted by $(\bar{u}_1^*, \bar{u}_2^*, \dots, \bar{u}_p^*, \bar{w}_1^*, \bar{w}_2^*, \dots, \bar{w}_q^*)$.

The following assumptions and lemmas are needed.

Assumption 11.3.1 The nonlinear system (11.3.1) is controllable.

Assumption 11.3.2 The upper value function and the lower value function both exist.

Assumption 11.3.3 The controls $u_k, k = 1, 2, \dots, p$ in Group I choose their policy independent from each other. The controls $w_j, j = 1, 2, \dots, q$, in Group II choose their policy independent from each other.

Based on the above assumptions, the following two lemmas are important in applications of the ADP method.

Lemma 11.3.1 *If Assumptions 11.3.1–11.3.3 hold, then for $0 \leq t \leq \hat{t} < \infty$, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^k$, $w \in \mathbb{R}^m$, we have*

$$\begin{aligned} \bar{V}(x(t)) &= \inf_{u_1} \dots \inf_{u_p} \sup_{w_1} \dots \sup_{w_q} \left\{ \int_t^{\hat{t}} l(x, u_1, \dots, u_p, w_1, \dots, w_q) d\tau + \bar{V}(x(\hat{t})) \right\}, \\ \underline{V}(x(t)) &= \sup_{w_1} \dots \sup_{w_p} \inf_{u_1} \dots \inf_{u_q} \left\{ \int_t^{\hat{t}} l(x, u_1, \dots, u_p, w_1, \dots, w_q) d\tau + \underline{V}(x(\hat{t})) \right\}. \end{aligned}$$

Lemma 11.3.2 *If the upper and lower value functions are defined as (11.3.3) and (11.3.4), respectively, we can obtain the following HJI equations*

$$\begin{aligned} \text{HJI}(\bar{V}(x), \bar{u}_1, \dots, \bar{u}_p, \bar{w}_1, \dots, \bar{w}_q) \\ = \bar{V}_t(x) + \bar{H}(\bar{V}_x(x), \bar{u}_1, \dots, \bar{u}_p, \bar{w}_1, \dots, \bar{w}_q) = 0, \end{aligned} \quad (11.3.5)$$

where $\bar{V}_t(x) = d\bar{V}(x)/dt$, $\bar{V}_x(x) = d\bar{V}(x)/dx$, $\bar{H}(\bar{V}_x(x), \bar{u}_1, \dots, \bar{u}_p, \bar{w}_1, \dots, \bar{w}_q)$ is called the upper Hamiltonian, and

$$\begin{aligned} \text{HJI}(\underline{V}(x), \underline{u}_1, \dots, \underline{u}_p, \underline{w}_1, \dots, \underline{w}_q) \\ = \underline{V}_t(x) + \underline{H}(\underline{V}_x(x), \underline{u}_1, \dots, \underline{u}_p, \underline{w}_1, \dots, \underline{w}_q) = 0, \end{aligned} \quad (11.3.6)$$

where $\underline{V}_t(x) = d\underline{V}(x)/dt$, $\underline{V}_x(x) = d\underline{V}(x)/dx$, $\underline{H}(\underline{V}_x(x), \underline{u}_1, \dots, \underline{u}_p, \underline{w}_1, \dots, \underline{w}_q)$ is called the lower Hamiltonian.

Remark 11.3.1 Optimal control problems do not necessarily have smooth or even continuous value functions [1]. In [7], it is shown that if the Hamiltonians are strictly convex in u and concave in w , then $\bar{V}(x) \in \mathcal{C}^1$ and $\underline{V}(x) \in \mathcal{C}^1$ satisfy the HJI equations (11.3.5) and (11.3.6) everywhere. If the smoothness property is removed, using the theory of viscosity solutions [7], it shows that for infinite horizon optimal control problems with unbounded value functionals, $\bar{V}(x)$ and $\underline{V}(x)$ are the unique viscosity solutions that satisfy the HJI equations (11.3.5) and (11.3.6), respectively, under Assumptions 11.3.1–11.3.3.

11.3.1 Derivation of the Iterative ADP Algorithm

The optimal control vector can be obtained by solving the HJI equations (11.3.5) and (11.3.6), but these equations cannot be solved analytically in general. There is no current method for rigorously confronting this type of equations to find the optimal value functions of the system. We introduce the iterative ADP method to tackle this problem. In this section, the iterative ADP method for multi-player zero-sum differential games is developed.

Theorem 11.3.1 Suppose Assumptions 11.3.1–11.3.3 hold. If the upper and lower value functions $\bar{V}(x)$ and $\underline{V}(x)$ are defined as (11.3.3) and (11.3.4), respectively, then

$$\begin{aligned} \bar{V}(x) &= \inf_{u_1} \dots \inf_{u_m} \dots \inf_{u_n} \dots \inf_{u_p} \sup_{w_1} \dots \sup_{w_{\bar{m}}} \dots \sup_{w_{\bar{n}}} \dots \sup_{w_q} \{V(x, u_1, \dots, u_p, w_1, \dots, w_q)\} \\ &= \inf_{u_1} \dots \inf_{u_n} \dots \inf_{u_m} \dots \inf_{u_p} \sup_{w_1} \dots \sup_{w_{\bar{n}}} \dots \sup_{w_{\bar{m}}} \dots \sup_{w_q} \{V(x, u_1, \dots, u_p, w_1, \dots, w_q)\} \end{aligned} \quad (11.3.7)$$

and

$$\begin{aligned}
& \underline{V}(x) \\
&= \sup_{w_1} \cdots \sup_{w_{\bar{m}}} \cdots \sup_{w_{\bar{n}}} \cdots \sup_{w_q} \inf_{u_1} \cdots \inf_{u_m} \cdots \inf_{u_n} \cdots \inf_{u_p} \{V(x, u_1, \dots, u_p, w_1, \dots, w_q)\} \\
&= \sup_{w_1} \cdots \sup_{w_{\bar{n}}} \cdots \sup_{w_{\bar{m}}} \cdots \sup_{w_q} \inf_{u_1} \cdots \inf_{u_n} \cdots \inf_{u_m} \cdots \inf_{u_p} \{V(x, u_1, \dots, u_p, w_1, \dots, w_q)\}
\end{aligned} \tag{11.3.8}$$

hold, for any m, n, \bar{m} and \bar{n} .

Proof We consider the upper value function. For any $w_j, j = 1, 2, \dots, p$, differentiating the HJI equation (11.3.5) with respect to the control w_j through the upper value function, it yields

$$\frac{\partial \bar{H}}{\partial w_j} = \bar{V}_x^\top \frac{\partial f(x, u_1, \dots, u_p, w_1, \dots, w_q)}{\partial w_j} + \frac{\partial l(x, u_1, \dots, u_p, w_1, \dots, w_q)}{\partial w_j} = 0.$$

According to Assumption 11.3.3, we can get

$$\bar{w}_j = -\frac{1}{2} C_j^{-1} c_j^\top(x) \bar{V}_x. \tag{11.3.9}$$

Taking the derivative of u_k , we have

$$\frac{\partial \bar{H}}{\partial u_k} = \bar{V}_x^\top \frac{\partial f(x, u_1, \dots, u_p, w_1, \dots, w_q)}{\partial u_k} + \frac{\partial l(x, u_1, \dots, u_p, w_1, \dots, w_q)}{\partial u_k} = 0.$$

Substituting (11.3.9) into (11.3.5) and using Assumption 11.3.3, we can obtain

$$\bar{u}_k = -\frac{1}{2} B_k^{-1} b_k^\top(x) \bar{V}_x. \tag{11.3.10}$$

From (11.3.9), we can see that for any $j = 1, \dots, q$, \bar{w}_j is independent from $\bar{w}_{j'}$, where $j \neq j'$. For any $k = 1, \dots, p$, \bar{u}_k is independent from $\bar{u}_{k'}$, where $k \neq k'$, which proves the conclusion in (11.3.7).

On the other hand, according to $\partial \underline{H}/\partial u_k = 0$ and $\partial \underline{H}/\partial w_j = 0$, we can get

$$\underline{u}_k = -\frac{1}{2} B_k^{-1} b_k^\top(x) \underline{V}_x \tag{11.3.11}$$

and

$$\underline{w}_j = -\frac{1}{2} C_j^{-1} c_j^\top(x) \underline{V}_x. \tag{11.3.12}$$

From (11.3.12) we can also see that for any $j = 1, \dots, q$, \underline{w}_j is independent from $\underline{w}_{j'}$, where $j \neq j'$. For any $k = 1, \dots, p$, \underline{u}_k is independent from $\underline{u}_{k'}$ where $k \neq k'$, which proves the conclusion in (11.3.8). This completes the proof of the theorem.

From (11.3.9) to (11.3.12), we can see that if the system functions $b_k(x)$ and $c_j(x)$ are obtained, then the upper and lower control pairs \bar{u}_k , \bar{w}_j , \underline{u}_k , \underline{w}_j , can be well defined. Next, NNs are introduced to construct the uncertain nonlinear system (11.3.1). For convenience of training the NNs, discretization of the continuous-time system function is necessary. According to Euler and trapezoidal methods [11], we have $\dot{x}(t) = (x(t+1) - x(t))/\Delta t$, where Δt is the sampling time interval which satisfies the Shannon's sampling theorem [22]. Then, the uncertain nonlinear system can be constructed as

$$x(t+1) = x(t) + \left(a(x(t)) + \sum_{l=1}^p b_k(x(t))u_k(t) + \sum_{j=1}^q c_j(x(t))w_j(t) \right) \Delta t. \quad (11.3.13)$$

Using NNs, the nonlinear system can be constructed by

$$x(t+1) = W_m^\top \sigma(V_m^\top X(t)) + e_m(t), \quad (11.3.14)$$

where $X(t) = [x^\top(t), U^\top(t), W^\top(t)]^\top$, $U(t) = [u_1^\top, u_2^\top, \dots, u_p^\top]^\top$, $W(t) = [w_1^\top, w_2^\top, \dots, w_q^\top]^\top$, and $e_m(t)$ is the bounded approximation error. The parameters W_m , V_m are the NN weights and σ is the activation function. Then, for $k = 1, 2, \dots, p$ and $j = 1, 2, \dots, q$, we can obtain

$$b_k(x(t)) = \frac{\partial(W_m^\top \sigma(V_m^\top X(t)))}{\partial u_k(t)} = W_m^\top \frac{\partial(\sigma(V_m^\top X(t)))}{\partial(V_m^\top X(t))} V_m^\top \frac{\partial X(t)}{\partial u_k(t)}, \quad (11.3.15)$$

and

$$c_j(x(t)) = \frac{\partial(W_m^\top \sigma(V_m^\top X(t)))}{\partial w_j(t)} = W_m^\top \frac{\partial(\sigma(V_m^\top X(t)))}{\partial(V_m^\top X(t))} V_m^\top \frac{\partial X(t)}{\partial w_j(t)}. \quad (11.3.16)$$

We can see that the system functions $b_k(x(t))$ and $c_j(x(t))$ are constructed by a NN if the weights W_m and V_m are obtained. This guarantees that the upper and lower control pairs \bar{u}_k , \bar{w}_j , \underline{u}_k , \underline{w}_j in (11.3.9)–(11.3.12) to be well defined.

According to (11.3.13)–(11.3.16), the nonlinear system (11.3.1) can be written as

$$\begin{aligned} \dot{x} &= f(x, U, W) \\ &= a(x) + \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial U} \right) U + \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial W} \right) W. \end{aligned} \quad (11.3.17)$$

The upper and lower value functions can be, respectively, rewritten as

$$\bar{V}(x) = \inf_U \sup_W V(x, U, W),$$

and

$$\underline{V}(x) = \sup_W \inf_U V(x, U, W).$$

Hence, we can see that if Theorem 11.3.1 holds, the multi-player zero-sum differential games can be changed to the two-player differential games which simplifies the problem. But we see that the upper and lower control vectors still cannot be solved. For example, if we want to obtain the upper control vectors $(\bar{u}_1, \dots, \bar{u}_p, \bar{w}_1, \dots, \bar{w}_q)$, we must obtain the upper value function $\bar{V}(x)$. Generally speaking, $\bar{V}(x)$ is unknown before all the control vectors

$$(\bar{u}_1, \dots, \bar{u}_p, \bar{w}_1, \dots, \bar{w}_q) \in \mathbb{R}^{n_1 + \dots + n_p + m_1 + \dots + m_q}$$

are considered. It is not possible to adopt the traditional dynamic programming method to obtain the optimal cost function at every time step due to the “curse of dimensionality.” Furthermore, the optimal control is discussed in infinite horizon. This means the length of the control sequence is infinite, which implies that the upper optimal control vector is nearly impossible to obtain by the HJI equation (11.3.5). To overcome this difficulty, an iterative ADP algorithm is developed next.

In the iterative ADP algorithm, the value function and control policy are updated by recurrent iterations, with the iteration number i increasing from 0 to ∞ . The present algorithm initializes with a stabilizing control pair $(U^{(0)}, W^{(0)})$, where Assumptions 11.3.1–11.3.3 hold. Then, for $i = 0, 1, \dots$, let the upper iterative value function be expressed as

$$\bar{V}^{(i)}(x(0)) = \int_0^\infty l(x, \bar{U}^{(i)}, \bar{W}^{(i)}) dt, \quad (11.3.18)$$

where

$$l(x, \bar{U}^{(i)}, \bar{W}^{(i)}) = x^\top A x + (\bar{U}^{(i)})^\top B \bar{U}^{(i)} + (\bar{W}^{(i)})^\top C \bar{W}^{(i)}$$

with B and C defined as

$$B = \begin{bmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_p \end{bmatrix}, \quad C = \begin{bmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_q \end{bmatrix}.$$

The upper iterative control pair is formulated as

$$\bar{U}^{(i)} = -\frac{1}{2}B^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{U}^{(i)}} \right)^\top \bar{V}_x^{(i)}, \quad (11.3.19)$$

and

$$\bar{W}^{(i)} = -\frac{1}{2}C^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right)^\top \bar{V}_x^{(i)}, \quad (11.3.20)$$

where $(\bar{U}^{(i)}, \bar{W}^{(i)})$ satisfies the HJI equation $\text{HJI}(\bar{V}^{(i)}(x), \bar{U}^{(i)}, \bar{W}^{(i)}) = 0$.

For $i = 0, 1, \dots$, let the lower iterative value function be

$$\underline{V}^{(i)}(x(0)) = \int_0^\infty l(x, \underline{U}^{(i)}, \underline{W}^{(i)}) dt. \quad (11.3.21)$$

The lower iterative control pair is formulated as

$$\underline{U}^{(i)} = -\frac{1}{2}B^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \underline{U}^{(i)}} \right)^\top \underline{V}_x^{(i)}, \quad (11.3.22)$$

and

$$\underline{W}^{(i)} = -\frac{1}{2}C^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \underline{W}^{(i)}} \right)^\top \underline{V}_x^{(i)}, \quad (11.3.23)$$

where $(\underline{U}^{(i)}, \underline{W}^{(i)})$ satisfies the HJI equation $\text{HJI}(\underline{V}^{(i)}(x), \underline{U}^{(i)}, \underline{W}^{(i)}) = 0$.

Remark 11.3.2 We should point out the following important fact. In [39], for two-player zero-sum differential games, it is proved that the iterative value functions converge to the optimal solution when the system function is given. As the nonlinear system (11.3.1) is replaced by (11.3.17), the upper control pair (\bar{u}_k, \bar{w}_j) in (11.3.9) and (11.3.10) is replaced by (11.3.19) and (11.3.20), respectively. The lower control pair (u_k, w_j) in (11.3.11) and (11.3.12) is replaced by (11.3.22) and (11.3.23), respectively. From (11.3.9) to (11.3.12), we can see that the system functions $b_k(x)$, $k = 1, 2, \dots, p$ and $c_j(x)$, $j = 1, 2, \dots, q$ are only the functions of x . While for (11.3.19) and (11.3.20), the NN-based functions are functions of x , $\bar{U}^{(i)}$ and $\bar{W}^{(i)}$. For (11.3.22) and (11.3.23), the NN-based functions are functions of x , $\underline{U}^{(i)}$ and $\underline{W}^{(i)}$. On the other hand, in [39], the system function is invariable for all i . While in this chapter, for different i , the system functions are also different for $i = 0, 1, \dots$. These are the two obvious differences from the results in [39].

In the next section, we will show that using NNs to construct the uncertain nonlinear system (11.3.1), the iterative control pairs can also guarantee the upper and lower iterative value functions to converge to the optimal solution of the game.

11.3.2 Properties

In this section, we show that the present iterative ADP algorithm for multi-player zero-sum differential games can be used to improve the properties of the nonlinear system.

Theorem 11.3.2 *Let Assumptions 11.3.1–11.3.3 hold, and $\bar{U}^{(i)} \in \mathbb{R}^k$, $\bar{W}^{(i)} \in \mathbb{R}^m$, $\bar{V}^{(i)}(x) \in \mathcal{C}^1$ satisfy the HJI equation*

$$\text{HJI}(\bar{V}^{(i)}(x), \bar{U}^{(i)}, \bar{W}^{(i)}) = 0, \quad i = 0, 1, \dots$$

If for any t , $l(x, \bar{U}^{(i)}, \bar{W}^{(i)}) \geq 0$, then the new control pairs $(\bar{U}^{(i+1)}, \bar{W}^{(i+1)})$ given by (11.3.19) and (11.3.20) which satisfy (11.3.18) guarantee the asymptotic stability of the nonlinear system (11.3.1).

Proof Since $\bar{V}^{(i)}(x) \in \mathcal{C}^1$, according to (11.3.20), we can get

$$\begin{aligned} \frac{d\bar{V}^{(i)}(x)}{dt} &= (\bar{V}_x^{(i)})^\top a(x) + (\bar{V}_x^{(i)})^\top \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{U}^{(i)}} \right) \bar{U}^{(i+1)} \\ &\quad - \frac{1}{2} (\bar{V}_x^{(i)})^\top \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right) \\ &\quad \times C^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right)^\top \bar{V}_x^{(i)}. \end{aligned} \quad (11.3.24)$$

From the HJI equation (11.3.5), we have

$$\begin{aligned} 0 &= (\bar{V}_x^{(i)})^\top a(x) + (\bar{V}_x^{(i)})^\top \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{U}^{(i)}} \right) \bar{U}^{(i)} \\ &\quad + x^\top A x + (\bar{U}^{(i)})^\top B \bar{U}^{(i)} - \frac{1}{4} (\bar{V}_x^{(i)})^\top \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right) \\ &\quad \times C^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right)^\top \bar{V}_x^{(i)}. \end{aligned} \quad (11.3.25)$$

Combining (11.3.24) and (11.3.25), we obtain

$$\begin{aligned} \frac{d\bar{V}^{(i)}(x)}{dt} = & -(\bar{U}^{(i+1)} - \bar{U}^{(i)})^\top B(\bar{U}^{(i+1)} - \bar{U}^{(i)}) - (\bar{U}^{(i+1)})^\top B\bar{U}^{(i+1)} \\ & - x^\top Ax - \frac{1}{4}(\bar{V}_x^{(i)})^\top \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial\bar{W}^{(i)}} \right) \\ & \times C^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial\bar{W}^{(i)}} \right)^\top \bar{V}_x^{(i)}. \end{aligned} \quad (11.3.26)$$

On the other hand, substituting (11.3.20) into the utility function, it follows

$$\begin{aligned} l\left(x, \bar{U}^{(i+1)}, \bar{W}^{(i+1)}\right) = & (\bar{U}^{(i+1)})^\top B\bar{U}^{(i+1)} + x^\top Ax \\ & + \frac{1}{4}(\bar{V}_x^{(i)})^\top \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial\bar{W}^{(i)}} \right) \\ & \times C^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial\bar{W}^{(i)}} \right)^\top \bar{V}_x^{(i)} \geq 0. \end{aligned} \quad (11.3.27)$$

Combining (11.3.26) and (11.3.27), we can derive $d\bar{V}^{(i)}(x)/dt \leq 0$.

As $\bar{V}^{(i)}(x)$ is positive definite (since $l(x, u, w)$ is positive definite) and $d\bar{V}^{(i)}(x)/dt \leq 0$, $\bar{V}^{(i)}(x)$ is a Lyapunov function. We can conclude that the system (11.3.1) is asymptotically stable. The proof is complete.

Theorem 11.3.3 *Let Assumptions 11.3.1–11.3.3 hold, and $U^{(i)} \in \mathbb{R}^k$, $W^{(i)} \in \mathbb{R}^m$, $V^{(i)}(x) \in \mathcal{C}^1$ satisfy the HJI equation $\text{HJI}(V^{(i)}(x), U^{(i)}, W^{(i)}) = 0$, $i = 0, 1, \dots$. If for all t , $l(x, U^{(i)}, W^{(i)}) < 0$, then the control pairs $(U^{(i)}, W^{(i)})$ formulated by (11.3.22) and (11.3.23) which satisfy the value function (11.3.21) guarantee system (11.3.1) to be asymptotically stable.*

Since the proof is similar to Theorem 11.3.2, we omit it here.

Corollary 11.3.1 *Let Assumptions 11.3.1–11.3.3 hold, $U^{(i)} \in \mathbb{R}^k$, $W^{(i)} \in \mathbb{R}^m$, $V^{(i)}(x) \in \mathcal{C}^1$ satisfy the HJI equation $\text{HJI}(V^{(i)}(x), U^{(i)}, W^{(i)}) = 0$, $i = 0, 1, \dots$. If for all t , $l(x, U^{(i)}, W^{(i)}) \geq 0$, then the control pairs $(U^{(i)}, W^{(i)})$ which satisfy the value function (11.3.21) guarantee system (11.3.1) to be asymptotically stable.*

Corollary 11.3.2 *Let Assumptions 11.3.1–11.3.3 hold, and $\bar{U}^{(i)} \in \mathbb{R}^k$, $\bar{W}^{(i)} \in \mathbb{R}^m$, $\bar{V}^{(i)}(x) \in \mathcal{C}^1$ satisfy the HJI equation $\text{HJI}(\bar{V}^{(i)}(x), \bar{U}^{(i)}, \bar{W}^{(i)}) = 0$, $i = 0, 1, \dots$. If for any t , $l(x, \bar{U}^{(i)}, \bar{W}^{(i)}) < 0$, then the control pairs $(\bar{U}^{(i)}, \bar{W}^{(i)})$ which satisfy the value function (11.3.18) guarantee system (11.3.1) to be asymptotically stable.*

Theorem 11.3.4 *Let Assumptions 11.3.1–11.3.3 hold. Also, let $\bar{U}^{(i)} \in \mathbb{R}^k$, $\bar{W}^{(i)} \in \mathbb{R}^m$, and $\bar{V}^{(i)}(x) \in \mathcal{C}^1$ satisfy the HJI equation $\text{HJI}(\bar{V}^{(i)}(x), \bar{U}^{(i)}, \bar{W}^{(i)}) = 0$, for*

$i = 0, 1, \dots$. Let $l(x, \bar{U}^{(i)}, \bar{W}^{(i)})$ be the utility function. Then, the control pairs $(\bar{U}^{(i)}, \bar{W}^{(i)})$ which satisfy the upper value function (11.3.18) guarantee system (11.3.1) to be asymptotically stable.

The proof is similar to Theorem 4 in [39], and thus it is omitted here.

Theorem 11.3.5 *Let Assumptions 11.3.1–11.3.3 hold, and $U^{(i)} \in \mathbb{R}^k$, $W^{(i)} \in \mathbb{R}^m$, $V^{(i)}(x) \in \mathcal{C}^1$ satisfies the HJI equation $\text{HJI}(V^{(i)}(x), U^{(i)}, W^{(i)}) = 0$, $i = 0, 1, \dots$. Let $l(x, U^{(i)}, W^{(i)})$ be the utility function. Then, the control pair $(U^{(i)}, W^{(i)})$ which satisfies the lower value function (11.3.21) is a pair of asymptotically stable controls for system (11.3.1).*

Next, the analysis of convergence property for the zero-sum differential games is presented to guarantee that the iterative control pairs reach the optimal solution.

Proposition 11.3.1 *Let Assumptions 11.3.1–11.3.3 hold, and $\bar{U}^{(i)} \in \mathbb{R}^k$, $\bar{W}^{(i)} \in \mathbb{R}^m$, $\bar{V}^{(i)}(x) \in \mathcal{C}^1$ satisfy the HJI equation $\text{HJI}(\bar{V}^{(i)}(x), \bar{U}^{(i)}, \bar{W}^{(i)}) = 0$. Then, the iterative control pairs $(\bar{U}^{(i)}, \bar{W}^{(i)})$ formulated by (11.3.19) and (11.3.20) guarantee the upper value function $\bar{V}^{(i)}(x) \rightarrow \bar{V}(x)$ as $i \rightarrow \infty$.*

Proof The proof consists of the following two steps.

(1) Show the convergence of the upper value function. From the HJI equation $\text{HJI}(\bar{V}^{(i)}(x), \bar{U}^{(i)}, \bar{W}^{(i)}) = 0$, we can obtain $d\bar{V}^{(i+1)}(x)/dt$ by replacing the index “ i ” by the index “ $i + 1$ ”

$$\begin{aligned} \frac{d\bar{V}^{(i+1)}(x)}{dt} = & - \left[x^\top A x + (\bar{U}^{(i+1)})^\top B \bar{U}^{(i+1)} \right. \\ & + \frac{1}{4} (\bar{V}_x^{(i)})^\top \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right) \\ & \left. \times C^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right)^\top \bar{V}_x^{(i)} \right]. \end{aligned}$$

According to (11.3.26), we can obtain

$$\begin{aligned} \frac{d(\bar{V}^{(i+1)}(x) - \bar{V}^{(i)}(x))}{dt} &= \frac{d\bar{V}^{(i+1)}(x)}{dt} - \frac{d\bar{V}^{(i)}(x)}{dt} \\ &= - \left[x^\top A x + (\bar{U}^{(i+1)})^\top B \bar{U}^{(i+1)} \right. \\ & + \frac{1}{4} (\bar{V}_x^{(i)})^\top \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right) \\ & \left. \times C^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right)^\top \bar{V}_x^{(i)} \right] \end{aligned}$$

$$\begin{aligned}
& - \left[- (\bar{U}^{(i+1)} - \bar{U}^{(i)})^\top B (\bar{U}^{(i+1)} - \bar{U}^{(i)}) \right. \\
& - x^\top A x - \frac{1}{4} (\bar{V}_x^{(i)})^\top \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right) \\
& \times C^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X))}{\partial(V_m^\top X)} V_m^\top \frac{\partial X}{\partial \bar{W}^{(i)}} \right)^\top \bar{V}_x^{(i)} \left. \right] \\
& = (\bar{U}^{(i+1)})^\top B \bar{U}^{(i+1)} \geq 0.
\end{aligned}$$

Since the system (11.3.1) is asymptotically stable, its state trajectory x converges to zero, and so does $\bar{V}^{(i+1)}(x) - \bar{V}^{(i)}(x)$. Since $d(\bar{V}^{(i+1)}(x) - \bar{V}^{(i)}(x))/dt \geq 0$ on these trajectories, it implies $\bar{V}^{(i+1)}(x) - \bar{V}^{(i)}(x) \leq 0$, i.e., $\bar{V}^{(i+1)}(x) \leq \bar{V}^{(i)}(x)$. Thus, $\bar{V}^{(i)}(x)$ is convergent as $i \rightarrow \infty$ since $\bar{V}^{(i)}(x) \geq 0$.

(2) Show that $\bar{V}^{(i)}(x) \rightarrow \bar{V}(x)$ as $i \rightarrow \infty$. We define

$$\lim_{i \rightarrow \infty} \bar{V}^{(i)}(x) = \bar{V}^{(\infty)}(x).$$

For any i , let

$$\bar{W}^* = \arg \max_W \left\{ \int_t^{\hat{t}} l(x, U, W) d\tau + \bar{V}^{(i)}(x(\hat{t})) \right\}.$$

Then, according to the principle of optimality expressed in Lemma 11.3.1, we have

$$\begin{aligned}
\bar{V}^{(i)}(x) & \leq \sup_W \left\{ \int_t^{\hat{t}} l(x, U, W) d\tau + \bar{V}^{(i)}(x(\hat{t})) \right\} \\
& = \int_t^{\hat{t}} l(x, U, \bar{W}^*) d\tau + \bar{V}^{(i)}(x(\hat{t})).
\end{aligned}$$

Since $\bar{V}^{(i+1)}(x) \leq \bar{V}^{(i)}(x)$, we have

$$\bar{V}^{(\infty)}(x) \leq \int_t^{\hat{t}} l(x, U, \bar{W}^*) d\tau + \bar{V}^{(i)}(x(\hat{t})).$$

Let $i \rightarrow \infty$. We can then obtain

$$\bar{V}^{(\infty)}(x) \leq \int_t^{\hat{t}} l(x, U, \bar{W}^*) d\tau + \bar{V}^{(\infty)}(x(\hat{t})).$$

So, it follows

$$\bar{V}^{(\infty)}(x) \leq \inf_U \sup_W \left\{ \int_t^{\hat{t}} l(x, u, w) d\tau + \bar{V}^{(\infty)}(x(\hat{t})) \right\}. \quad (11.3.28)$$

Let $\varepsilon > 0$ be an arbitrary positive number. Since the upper value function is nonincreasing and convergent, there exists a positive integer i such that

$$\bar{V}^{(i)}(x) - \varepsilon \leq \bar{V}^{(\infty)}(x) \leq \bar{V}^{(i)}(x).$$

Let $\bar{U}^* = \arg \min_U \left\{ \int_t^{\hat{t}} l(x, \bar{U}, \bar{W}^*) d\tau + \bar{V}^{(i)}(x(\hat{t})) \right\}$. Then, we can get

$$\bar{V}^{(i)}(x) = \int_t^{\hat{t}} l(x, \bar{U}^*, \bar{W}^*) d\tau + \bar{V}^{(i)}(x(\hat{t})).$$

Thus,

$$\begin{aligned} \bar{V}^{(\infty)}(x) &\geq \int_t^{\hat{t}} l(x, \bar{U}^*, \bar{W}^*) d\tau + \bar{V}^{(i)}(x(\hat{t})) - \varepsilon \\ &\geq \int_t^{\hat{t}} l(x, \bar{U}^*, \bar{W}^*) d\tau + \bar{V}^{(\infty)}(x(\hat{t})) - \varepsilon \\ &= \inf_U \sup_W \left\{ \int_t^{\hat{t}} l(x, U, W) d\tau + \bar{V}^{(\infty)}(x(\hat{t})) \right\} - \varepsilon. \end{aligned}$$

Since ε is arbitrary, we obtain

$$\bar{V}^{(\infty)}(x) \geq \inf_U \sup_W \left\{ \int_t^{\hat{t}} l(x, U, W) d\tau + \bar{V}^{(\infty)}(x(\hat{t})) \right\}. \quad (11.3.29)$$

Combining (11.3.28) and (11.3.29), it follows

$$\bar{V}^{(\infty)}(x) = \inf_U \sup_W \left\{ \int_t^{\hat{t}} l(x, U, W) d\tau + \bar{V}^{(\infty)}(x(\hat{t})) \right\}.$$

Letting $\hat{t} \rightarrow \infty$, we have

$$\bar{V}^{(\infty)}(x) = \inf_U \sup_W V(x, U, W),$$

which is the same as (11.3.3). So, $\bar{V}^{(i)}(x) \rightarrow \bar{V}(x)$ as $i \rightarrow \infty$. The proof is complete.

Proposition 11.3.2 *Let Assumptions 11.3.1–11.3.3 hold, $\underline{U}^{(i)} \in \mathbb{R}^k$, $\underline{W}^{(i)} \in \mathbb{R}^m$ and $\underline{V}^{(i)}(x) \in \mathcal{C}^1$ satisfy the HJI function $\text{HJI}(\underline{V}^{(i)}(x), \underline{U}^{(i)}, \underline{W}^{(i)}) = 0$. Then the iterative control pair $(\underline{U}^{(i)}, \underline{W}^{(i)})$ formulated by (11.3.22) and (11.3.23) guarantees the lower value function $\underline{V}^{(i)}(x) \rightarrow \underline{V}(x)$ as $i \rightarrow \infty$.*

Remark 11.3.3 In [39], for the known nonlinear systems, the convergence property was proved for two-player zero-sum differential games. We should point out that in [39], the system function is invariant for any i . For multi-player zero-sum differential games, the NN-based system functions are also updated for any $i = 0, 1, \dots$. While from Propositions 11.3.1 and 11.3.2, we can see that by the approximation of NNs for nonlinear systems, the convergence property can also be guaranteed which shows the effectiveness of the present method.

Theorem 11.3.6 *If the optimal value function of the zero-sum differential game or the optimal solution exists, then the control pairs $(\overline{U}^{(i+1)}, \overline{W}^{(i+1)})$ and $(\underline{U}^{(i+1)}, \underline{W}^{(i+1)})$ guarantee $\overline{V}^{(i)}(x) \rightarrow V^*(x)$ and $\underline{V}^{(i)}(x) \rightarrow V^*(x)$, respectively, as $i \rightarrow \infty$.*

Proof For the upper value function, according to Proposition 11.3.1, we have $\overline{V}^{(i)}(x) \rightarrow \overline{V}(x)$ under the control pair $(\overline{U}^{(i+1)}, \overline{W}^{(i+1)})$ as $i \rightarrow \infty$. So, the optimal control pair for upper value function satisfies

$$\overline{V}(x) = V(x, \overline{U}, \overline{W}) = \inf_U \sup_W V(x, U, W). \quad (11.3.30)$$

On the other hand, there exists an optimal control pair (U^*, W^*) under which the value function reaches the optimal solution. According to the property of the optimal solution, the optimal control pair (U^*, W^*) satisfies

$$V^*(x) = V(x, U^*, W^*) = \inf_U \sup_W V(x, U, W),$$

which is the same as (11.3.30). So, $\overline{V}(x) \rightarrow V^*(x)$ under the control pair $(\overline{U}^{(i+1)}, \overline{W}^{(i+1)})$ as $i \rightarrow \infty$.

Similarly, we can derive $\underline{V}(x) \rightarrow V^*(x)$ under the control pair $(\underline{U}^{(i+1)}, \underline{W}^{(i+1)})$ as $i \rightarrow \infty$. This completes the proof of the theorem.

Remark 11.3.4 From Theorem 11.3.6, we can see that if the optimal solution exists, the upper and lower iterative value functions converge to the optimum under the iterative control pairs $(\overline{U}^{(i)}, \overline{W}^{(i)})$ and $(\underline{U}^{(i)}, \underline{W}^{(i)})$, respectively. Since the existence criterions of the optimal solution of the games in [2, 3, 8] are unnecessary, we can see that the present iterative ADP algorithm is more effective for solving zero-sum differential games.

11.3.3 Neural Network Implementation

As the computer can only deal with the digital and discrete signals, it is necessary to transform the continuous-time value function to a corresponding discrete-time form. Discretization of the value function using Euler and trapezoidal methods [11] leads to

$$V(x(0)) = \sum_{t=0}^{\infty} (x^T(t) Ax(t) + U^T(t) BU(t) + W^T(t) CW(t)) \Delta t,$$

where Δt is the sample time interval.

In [1, 24, 25], polynomial approximation is used to construct single-layer NNs that approximate the value function and the control policy, respectively, to implement the ADP algorithm. While using the polynomial approximation method, the order of the polynomial sequence and the length of the polynomial sequence are both difficult to determine. As is known a three-layer BP NN can approximate any MIMO continuous functions [12]. Three-layer BP NNs will be used to implement the present iterative ADP algorithm.

Assume the number of hidden layer neurons is denoted by λ , the weight matrix between the input layer and hidden layer is denoted by Y_f , and the weight matrix between the hidden layer and output layer is denoted by W_f . Then, the output of three-layer NN is represented by:

$$\hat{F}(X, Y_f, W_f) = W_f^T \sigma(Y_f^T X),$$

where $\sigma(Y_f^T X) \in \mathbb{R}^\lambda$, $[\sigma(z)]_i = \frac{e^{z_i} - e^{-z_i}}{e^{z_i} + e^{-z_i}}$, $i = 1, \dots, \lambda$, are the activation functions. Using NNs, the estimation error can be expressed by

$$F(X) = F(X, V_f^*, W_f^*) + \xi(X),$$

where V_f^* , W_f^* are the ideal weight parameters, and $\xi(X)$ is the estimation error.

A. Neural Network Identifier for Uncertain Nonlinear Systems

In this case, the uncertain nonlinear system will be constructed by NN using the input–output data. For convenience of analysis, only the output weights are updated during the training, while the hidden weights are kept fixed. Let $\bar{X}(t) = Y_m^T X(t)$. The activation function $\sigma(Y_m^T X(t))$ can be rewritten as $\sigma(\bar{X}(t))$. Thus, the NN model for the system is constructed as

$$\hat{x}(t+1) = \hat{W}_m^T(t) \sigma(\bar{X}(t)), \quad (11.3.31)$$

where $\hat{x}(t+1)$ is the estimated system state vector, $\hat{W}_m(t)$ is the estimation of the ideal weight matrix W_m . According to (11.3.14), we can define the system identification error as

$$\tilde{x}(t+1) = \hat{x}(t+1) - x(t+1) = \tilde{W}_m(t)\sigma(\bar{Z}(t)) - e_m(t),$$

where $\tilde{W}_m(t) = \hat{W}_m(t) - W_m$. Let $\phi(t) = \tilde{W}_m^\top(t)\sigma(\bar{Z}(t))$. Then, we can get

$$\tilde{x}(t+1) = \phi(t) - e_m(t). \quad (11.3.32)$$

Define the performance error index as:

$$E_m(t) = \frac{1}{2}\tilde{x}^\top(t+1)\tilde{x}(t+1).$$

Then, the gradient-based weight update rule for the critic network can be described as

$$\tilde{W}_m(t+1) = \tilde{W}_m(t) + \Delta\tilde{W}_m(t), \quad (11.3.33)$$

$$\Delta\tilde{W}_m(t) = \eta_m \left[-\frac{\partial E_m(t)}{\partial \tilde{W}_m(t)} \right], \quad (11.3.34)$$

where $\eta_m > 0$ is the learning rate of the model network. After the model network is trained, its weights are kept unchanged.

Next, we will give the convergence theorem of NNs.

Theorem 11.3.7 *Let the NN be expressed by (11.3.31) and let the NN weights be updated by (11.3.33) and (11.3.34). The approximation error e_m is expressed as in (11.3.14). If there exists a constant $0 < \lambda_M < 1$ such that*

$$e_m^\top(t)e_m(t) \leq \lambda_M \tilde{x}^\top(t)\tilde{x}(t), \quad (11.3.35)$$

then the system identification error $\tilde{x}(t)$ is asymptotically stable while the parameter estimation error $\tilde{W}_m(t)$ is bounded.

Proof First, to obtain the conclusion, we built the Lyapunov function as follows:

$$L(x) = \tilde{x}^\top(t)\tilde{x}(t) + \frac{1}{\eta_m} \text{tr}\{\tilde{W}_m^\top(t)\tilde{W}_m(t)\}.$$

The first difference of the Lyapunov function candidate is given by

$$\begin{aligned} \Delta L(x) &= \tilde{x}^\top(t+1)\tilde{x}(t+1) - \tilde{x}^\top(t)\tilde{x}(t) \\ &\quad + \frac{1}{\eta_m} \text{tr}\{\tilde{W}_m^\top(t+1)\tilde{W}_m(t+1) - \tilde{W}_m^\top(t)\tilde{W}_m(t)\}. \end{aligned}$$

From (11.3.33)–(11.3.34), the weights of the NN are updated as

$$\hat{W}_m(t+1) = \hat{W}_m(t) - \eta_m \sigma(\bar{Z}(t))\tilde{x}(t+1).$$

According to (11.3.32), we can obtain

$$\begin{aligned}\Delta L(x) = & \phi^T(t) \phi(t) - 2\phi^T(t) e_m(t) + \eta_m \sigma^T(\bar{Z}(t)) \sigma(\bar{Z}(t)) \tilde{x}^T(t+1) \tilde{x}(t+1) \\ & + e_m^T(t) e_m(t) - \tilde{x}^T(t) \tilde{x}(t) - 2\phi^T(t) \tilde{x}(t+1).\end{aligned}$$

Using the Cauchy–Schwarz inequality, we can get

$$\begin{aligned}\Delta L(x) \leq & -\phi^T(t) \phi(t) + e_m^T(t) e_m(t) - \tilde{x}^T(t) \tilde{x}(t) \\ & + 2\eta_m \sigma^T(\bar{Z}(t)) \sigma(\bar{Z}(t)) (\phi^T(t) \phi(t) + e_m^T(t) e_m(t)).\end{aligned}$$

As $||\sigma(\bar{Z}(t))||$ is finite, we can define $||\sigma(\bar{Z}(t))|| \leq \sigma_M$, where $\sigma_M > 0$ is the upper bound. According to (11.3.35), we can get

$$\Delta L(x) \leq -(1 - 2\eta_m \sigma_M^2) ||\phi(t)||^2 - (1 - \lambda_M - 2\eta_m \lambda_M \sigma_M^2) ||\tilde{x}(t)||^2. \quad (11.3.36)$$

Let η_m be selected as $\eta_m \leq \beta^2 / 2\sigma_M^2$, then (11.3.36) becomes

$$\Delta L(x) \leq -(1 - \beta^2) ||\phi(t)||^2 - (1 - \lambda_M - \lambda_M \beta^2) ||\tilde{x}(t)||^2. \quad (11.3.37)$$

Therefore, $\Delta L(x) \leq 0$ provided that $0 < \lambda_M \leq 1$ and

$$\max \left\{ -\sqrt{\frac{1 - \lambda_M}{\lambda_M}}, -1 \right\} \leq \beta \leq \min \left\{ \sqrt{\frac{1 - \lambda_M}{\lambda_M}}, 1 \right\},$$

where $\beta \neq 0$. As long as the parameters are selected as discussed above, we can have $\Delta L(x) \leq 0$. Therefore, the identification error $\tilde{x}(t)$ and the weight estimation error $\tilde{W}_m(t)$ are bounded if $\tilde{x}(0)$ and $\tilde{W}_m(0)$ are bounded. Furthermore, by summing both sides of (11.3.37) to infinity and taking limits when $t \rightarrow \infty$, it can be concluded that the estimation error $\tilde{x}(t)$ approaches zero when $t \rightarrow \infty$. This completes the proof of the theorem.

Remark 11.3.5 In [26], ADP algorithm was used to solve a nonlinear multi-player nonzero-sum game which can be converted to zero-sum one. In [26], we can see that the system function is necessary in order to obtain the optimal control pair. In this chapter, we see that the control pair is obtained without the system model. If Theorem 11.3.7 is satisfied, the system dynamics can be well constructed by an NN which guarantees the effectiveness of the present algorithm.

B. Neural Networks for the Iterative ADP

We use eight NNs to implement the iterative ADP method, where two are model networks (the weights of the two model NNs are the same), two are critic networks, and four are action networks, respectively. All the NNs are chosen as three-layer feedforward NNs. The structural diagram is shown in Fig. 11.5.

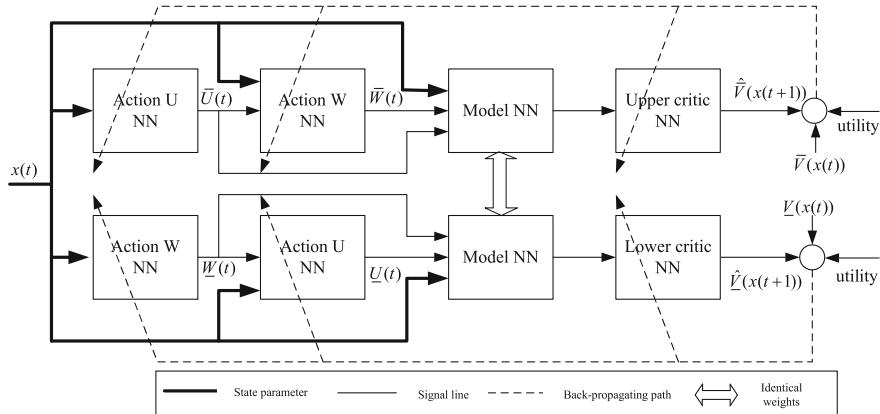


Fig. 11.5 Structural diagram of the iterative ADP method for multi-person zero-sum games

The critic network is used to approximate upper and lower iterative value functions, i.e., $\bar{V}^{(i)}(x)$ and $\underline{V}^{(i)}(x)$. The output of the critic network is denoted as

$$\hat{V}^{(i)}(x(t)) = (W_c^{(i)})^\top \sigma((V_c^{(i)})^\top X(t)),$$

where $X(t) = [x^\top(t), u^\top(t), w^\top(t)]^\top$ is the input of critic networks. The two critic networks have the following target functions. For the upper value function, the target function can be written as

$$\begin{aligned} \bar{V}^{(i)}(x(t)) = & \left(x^\top(t) A x(t) + (\bar{U}^{(i+1)}(t))^\top B \bar{U}^{(i+1)}(t) \right. \\ & \left. + (\bar{W}^{(i+1)}(t))^\top C \bar{W}^{(i+1)}(t) \right) \Delta t + \hat{\bar{V}}^{(i)}(x(t+1)), \end{aligned}$$

where $\hat{\bar{V}}^{(i)}(x(t+1))$ is the output of the upper critic network. For the lower value function, the target function can be written as

$$\begin{aligned} \underline{V}^{(i)}(x(t)) = & \left(x^\top(t) Q x(t) + (\underline{U}^{(i+1)}(t))^\top B \underline{U}^{(i+1)}(t) \right. \\ & \left. + (\underline{W}^{(i+1)}(t))^\top C \underline{W}^{(i+1)}(t) \right) \Delta t + \hat{\underline{V}}^{(i)}(x(t+1)), \end{aligned} \quad (11.3.38)$$

where $\hat{\underline{V}}^{(i)}(x(t+1))$ is the output of the lower critic network.

Then, for the upper value function, we define the error function of the critic network by

$$\bar{e}_c^{(i)}(t) = \bar{V}^{(i)}(x(t)) - \hat{\bar{V}}^{(i)}(x(t)),$$

where $\hat{V}^{(i)}(x(t))$ is the output of the upper critic network. The objective function to be minimized in the critic network is

$$E_c^{(i)}(t) = \frac{1}{2}(\bar{e}_c^{(i)}(t))^2.$$

So, the gradient-based weight update rule for the critic network is given by

$$W_c^{(i)}(t+1) = W_c^{(i)}(t) + \Delta W_c^{(i)}(t), \quad \Delta W_c^{(i)}(t) = \eta_c \left[-\frac{\partial E_c^{(i)}(t)}{\partial W_c^{(i)}(t)} \right],$$

$$\frac{\partial E_c^{(i)}(t)}{\partial W_c^{(i)}(t)} = \frac{\partial E_c^{(i)}(t)}{\partial \hat{V}^{(i)}(x(t))} \frac{\partial \hat{V}^{(i)}(x(t))}{\partial W_c^{(i)}(t)},$$

where $\eta_c > 0$ is the learning rate of critic network and $W_c(t)$ is the weight vector in the critic network, which can be replaced by $W_c^{(i)}$ and $V_c^{(i)}$.

For the lower value function, the error function of the critic network is defined by

$$\underline{e}_c^{(i)}(t) = \underline{V}^{(i)}(x(t)) - \hat{V}^{(i)}(x(t)).$$

For the lower iterative value function, the weight updating rule of the critic network is the same as the one for upper value function. The details are omitted here. To implement the present iterative ADP algorithm, four action networks are used to approximate the laws of the upper and lower iterative control pairs, where two are used to approximate the laws of upper iterative control pairs and the other two are used to approximate the laws of lower iterative control pairs.

The target functions of the upper U and W action networks are the discretization formulation of equations (11.3.19) and (11.3.20), respectively, which can be written as

$$\bar{U}^{(i+1)}(t) = -\frac{1}{2}B^{-1} \left(W_m^T \frac{\partial(\sigma(V_m^T X(t)))}{\partial(V_m^T X(t))} V_m^T \frac{\partial X(t)}{\partial \bar{U}^{(i)}(t)} \right)^T \frac{d\bar{V}^{(i)}(x(t+1))}{dx(t+1)}, \quad (11.3.39)$$

and

$$\bar{W}^{(i+1)}(t) = -\frac{1}{2}C^{-1} \left(W_m^T \frac{\partial(\sigma(V_m^T X(t)))}{\partial(V_m^T X(t))} V_m^T \frac{\partial X(t)}{\partial \bar{W}^{(i)}(t)} \right)^T \frac{d\bar{V}^{(i)}(x(t+1))}{dx(t+1)}.$$

The target functions of the lower U and W action networks are the discretization formulation of (11.3.22) and (11.3.23), respectively, which can be written as

$$\underline{U}^{(i+1)}(t) = -\frac{1}{2} B^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X(t)))}{\partial(V_m^\top X(t))} V_m^\top \frac{\partial X(t)}{\partial \underline{U}^{(i)}(t)} \right)^\top \frac{dV^{(i)}(x(t+1))}{dx(t+1)},$$

and

$$\underline{W}^{(i+1)}(t) = -\frac{1}{2} C^{-1} \left(W_m^\top \frac{\partial(\sigma(V_m^\top X(t)))}{\partial(V_m^\top X(t))} V_m^\top \frac{\partial X(t)}{\partial \underline{W}^{(i)}(t)} \right)^\top \frac{dV^{(i)}(x(t+1))}{dx(t+1)}.$$

The output of the action network (the upper U action network for example) can be formulated as

$$\hat{\underline{U}}^{(i)}(t) = (\bar{W}_{ua}^{(i)})^\top \sigma((\bar{Y}_{ua}^{(i)})^\top x(t)).$$

The target of the output of the action network is given by (11.3.39). So, we can define the output error of the action network as

$$\bar{e}_{ua}^{(i)}(t) = \bar{U}^{(i)}(t) - \hat{\underline{U}}^{(i)}(t).$$

The weights of the action network are updated to minimize the following performance error measure

$$\bar{E}_{ua}^{(i)}(t) = \frac{1}{2} (\bar{e}_{ua}^{(i)}(t))^2.$$

The weight update algorithm is similar to the one for the critic network. Using the gradient descent rule, we can obtain

$$\begin{aligned} W_a^{(i)}(t+1) &= W_a^{(i)}(t) + \Delta W_a^{(i)}(t), \\ \Delta W_a^{(i)}(t) &= \eta_a \left[-\frac{\partial E_a^{(i)}(t)}{\partial W_a^{(i)}(t)} \right], \\ \frac{\partial E_a^{(i)}(t)}{\partial W_a^{(i)}(t)} &= \frac{\partial E_a^{(i)}(t)}{\partial e_a^{(i)}(t)} \frac{\partial e_a^{(i)}(t)}{\partial \hat{\underline{U}}^{(i)}(t)} \frac{\partial \hat{\underline{U}}^{(i)}(t)}{\partial W_a^{(i)}(t)}, \end{aligned}$$

where $\eta_a > 0$ is the learning rate of action network, and $W_a^{(i)}(t)$ is the weight vector of the action network, which can be replaced by $\bar{W}_{ua}^{(i)}$ and $\bar{V}_{ua}^{(i)}$, respectively. The weight update rule for the other action networks is similar and is omitted.

Given the above preparation, we now formulate the iterative ADP algorithm for the nonlinear multi-player zero-sum differential games as follows.

- Step 1: Initialize the algorithm with a stabilizing control pair $(U^{(0)}, W^{(0)})$, where Assumptions 11.3.1–11.3.3 hold. Choose the computation precision $\zeta > 0$.
- Step 2: Discretize the nonlinear system (11.3.1) as (11.3.13) and construct a model NN. Train the model NN according to (11.3.31)–(11.3.34), and obtain the system function $b_k(x(t))$, $k = 1, 2, \dots, p$ and $c_j(x(t))$, $j = 1, 2, \dots, q$, according to (11.3.15) and (11.3.16), respectively.

Step 3: For $i = 0, 1, \dots$, for upper value function, let

$$\bar{V}^{(i)}(x(0)) = \sum_{t=0}^{\infty} l \left(x(t), \bar{U}^{(i+1)}(t), \bar{W}^{(i+1)}(t) \right) \Delta t,$$

where

$$\begin{aligned} l \left(x(t), \bar{U}^{(i+1)}(t), \bar{W}^{(i+1)}(t) \right) &= x^T(t) A x(t) + (\bar{U}^{(i+1)}(t))^T B \bar{U}^{(i+1)}(t) \\ &\quad + (\bar{W}^{(i+1)}(t))^T C \bar{W}^{(i+1)}(t). \end{aligned}$$

Update the upper iterative control pairs by

$$\begin{cases} \bar{U}^{(i+1)} = -\frac{1}{2} B^{-1} \left(W_m^T \frac{\partial(\sigma(V_m^T X))}{\partial(V_m^T X)} V_m^T \frac{\partial X}{\partial \bar{U}^{(i)}} \right)^T \bar{V}_x^{(i)}, \\ \bar{W}^{(i+1)} = -\frac{1}{2} C^{-1} \left(W_m^T \frac{\partial(\sigma(V_m^T X))}{\partial(V_m^T X)} V_m^T \frac{\partial X}{\partial \bar{W}^{(i)}} \right)^T \bar{V}_x^{(i)}, \end{cases}$$

where the HJI equation $\text{HJI}(\bar{V}^{(i)}(x(t)), \bar{U}^{(i)}(t), \bar{W}^{(i)}(t)) = 0$ is satisfied for any $i = 0, 1, \dots$

Step 4: If

$$|\bar{V}^{(i+1)}(x(0)) - \bar{V}^{(i)}(x(0))| < \zeta,$$

let

$$\bar{U}(t) = \bar{U}^{(i)}(t), \quad \bar{W}(t) = \bar{W}^{(i)}(t), \quad \text{and} \quad \bar{V}(x(t)) = \bar{V}^{(i+1)}(x(t)),$$

and go to next step; else, set $i = i + 1$ and go to Step 3.

Step 5: For $i = 0, 1, \dots$, for lower value function, let

$$\underline{V}^{(i)}(x(0)) = \sum_{t=0}^{\infty} l \left(x(t), \underline{U}^{(i+1)}(t), \underline{W}^{(i+1)}(t) \right) \Delta t,$$

and the lower iterative control pairs be updated by

$$\begin{cases} \underline{U}^{(i+1)} = -\frac{1}{2} B^{-1} \left(W_m^T \frac{\partial(\sigma(V_m^T X))}{\partial(V_m^T X)} V_m^T \frac{\partial X}{\partial \underline{V}^{(i)}} \right)^T \underline{V}_x^{(i)}, \\ \underline{W}^{(i+1)} = -\frac{1}{2} C^{-1} \left(W_m^T \frac{\partial(\sigma(V_m^T X))}{\partial(V_m^T X)} V_m^T \frac{\partial X}{\partial \underline{W}^{(i)}} \right)^T \underline{V}_x^{(i)}, \end{cases}$$

where the HJI equation

$$\text{HJI}\left(\underline{V}^{(i)}(x(t)), \underline{U}^{(i)}(t), \underline{W}^{(i)}(t)\right) = 0$$

is satisfied for any $i = 0, 1, \dots$

Step 6: If

$$|\underline{V}^{(i+1)}(x(0)) - \underline{V}^{(i)}(x(0))| < \zeta,$$

let

$$\underline{U}(t) = \underline{U}^{(i)}(t), \quad \underline{W}(t) = \underline{W}^{(i)}(t), \quad \text{and} \quad \underline{V}(x(t)) = \underline{V}^{(i+1)}(x(t)),$$

and go to the next step; else, set $i = i + 1$ and go to Step 5.

Step 7: If

$$|\bar{V}(x(0)) - \underline{V}(x(0))| < \zeta,$$

stop, and the optimal solution is achieved; else, stop, and the optimal solution dose not exist.

11.3.4 Simulation Studies

In this section, two examples will be provided to demonstrate the effectiveness of the present optimal control scheme.

Example 11.3.1 Our first example is chosen from [9] with modifications. Consider the following linear system

$$\dot{x} = x + u + w,$$

with the initial state $x(0) = 1$. The value function is defined by (11.3.2) with

$$A = 1, \quad B = \frac{1}{4}, \quad \text{and} \quad C = -1.$$

Discretization of the value function using Euler and trapezoidal methods with the sampling time interval $\Delta t = 10^{-2}$. NNs are used to implement the iterative ADP algorithm.

To obtain a good approximation, it is important to consider the NN structure carefully. As we choose three-layer NNs to implement the iterative ADP algorithm, the number of the hidden layer neurons λ can be chosen by the following equation from experience

$$\lambda = \sqrt{N_I + N_O} + a, \quad (11.3.40)$$

where N_I and N_O are the dimensions of the input and output vectors, respectively. The constant a is an integer between 1 and 10. Equation (11.3.40) shows that for low-

dimensional system, the number of the hidden layer neurons can be small. While for high-dimensional complex nonlinear systems, the number of the hidden layer neurons should increase correspondingly.

According to (11.3.40), the number of the hidden layer neurons is chosen as 8. The structure of the model NN is chosen as 2–8–1. The structures of the critic networks for upper and lower value functions are chosen as 1–8–1. For the upper value functions, the structures of the U action network and the W action network are chosen as 1–8–1 and 2–8–1, respectively. For the lower value functions, the structures of the U action network and the W action network are chosen as 2–8–1 and 1–8–1, respectively. The initial weights of action networks, critic networks, and model network are all set to be random in $[-0.5, 0.5]$. It should be mentioned that the model network should be trained first. For the given initial state $x(0) = 1$, we train the model network for 1000 steps under the learning rate $\eta_m = 0.002$ to reach the given accuracy of $\varepsilon = 10^{-6}$. After the training of the model network is complete, the weights are kept unchanged. During each iteration step, the critic networks and the action networks are trained for 200 steps to reach the given accuracy of $\varepsilon = 10^{-6}$. In the training process, the learning rate $\eta_a = \eta_c = 0.01$. The convergence curves of the upper and lower value functions are shown in Fig. 11.6.

The convergence trajectories of the first row of the weights of the critic network are shown in Fig. 11.7. The convergence trajectories of the hidden layer weights for the U action network are shown in Fig. 11.8. The corresponding state and control trajectories are displayed in Fig. 11.9a and b, respectively.

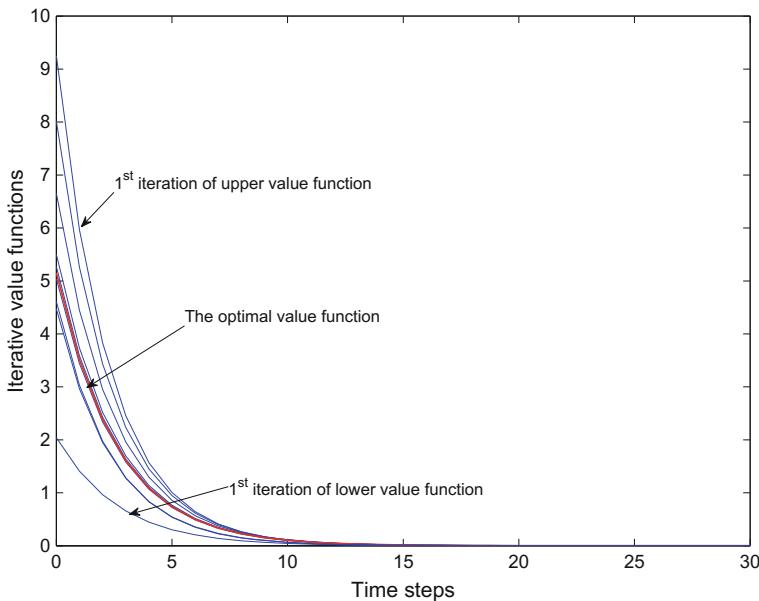


Fig. 11.6 Convergence of the upper and lower value functions

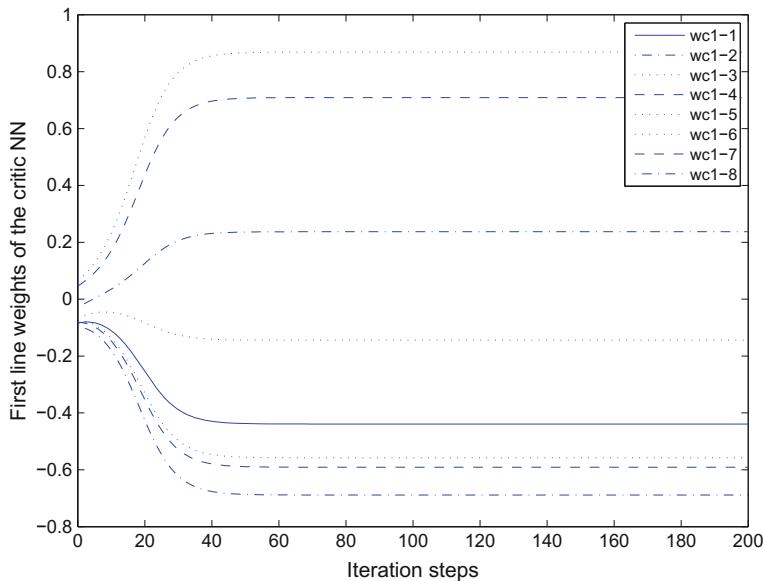


Fig. 11.7 Convergence trajectories of the first row of weights of the critic network

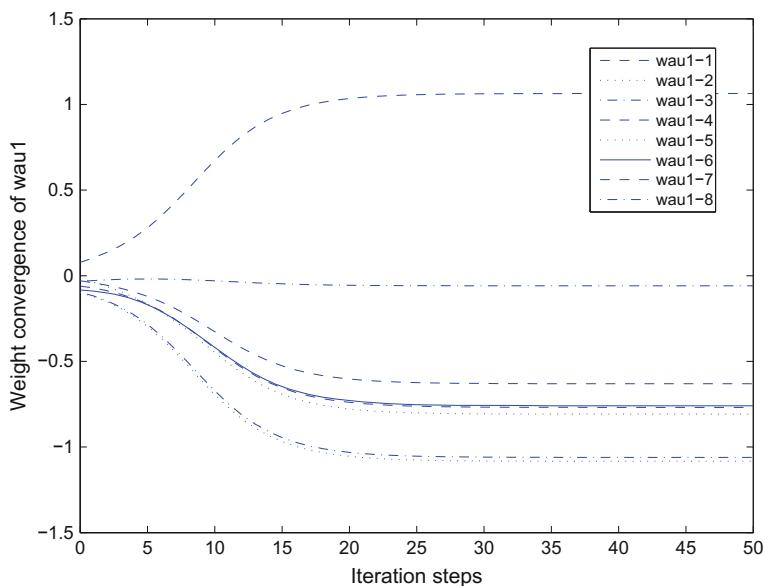


Fig. 11.8 Convergence trajectories of the first row of weights of the U action network

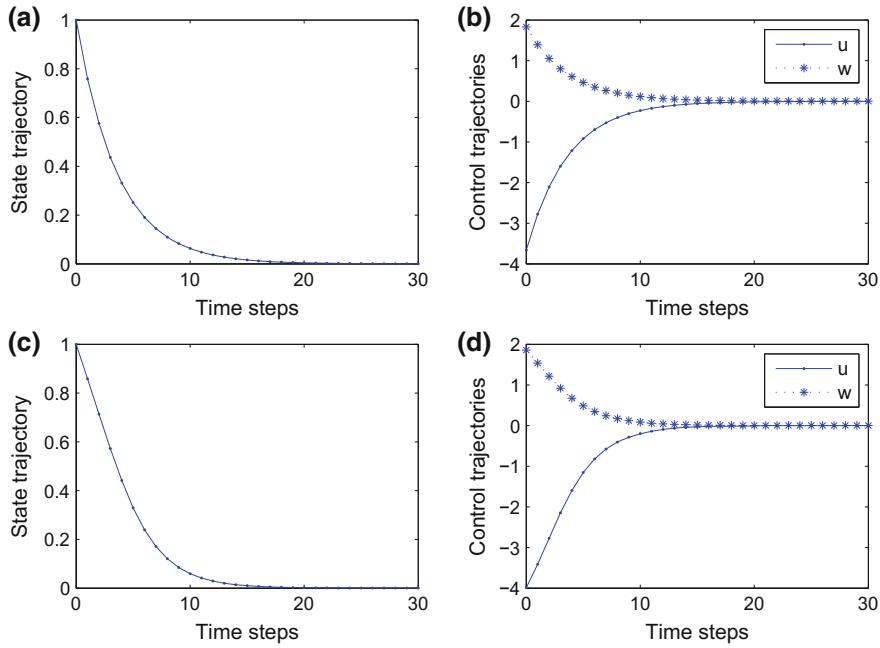


Fig. 11.9 State and control trajectories

On the other hand, it is known that the solution of the optimal control problem for the linear system is quadratic in the state [8]. We can easily obtain that $P^* = 2.4142$ and the optimal control laws are $u^* = -2P^*x$ and $w^* = P^*x$, respectively. The state and control trajectories by GARE approach are displayed in Fig. 11.9c and Fig. 11.9d, respectively.

From the comparison results, we can see that the present ADP algorithm is effective for the zero-sum differential games of linear systems.

Example 11.3.2 The system function is expressed as

$$\begin{aligned}\dot{x} = & \begin{bmatrix} 0.1x_1^2 + 0.05x_2 \\ 0.2x_1^2 - 0.15x_2 \end{bmatrix} \\ & + \begin{bmatrix} 0.1 + x_1 + x_2^2 & 0.1 + x_2 + x_1^2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \\ & + \begin{bmatrix} 0.1 + x_1 + x_1x_2 & 0 \\ 0 & 0.2 + x_1 + x_1x_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}\end{aligned}$$

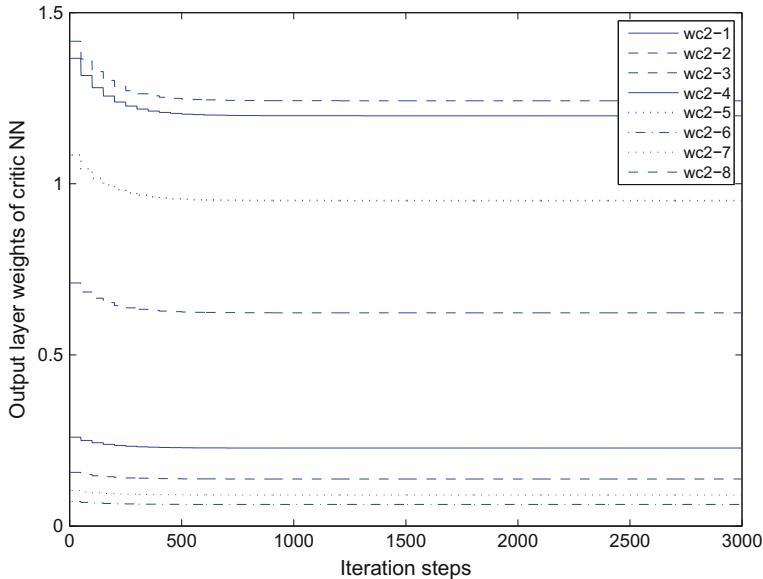


Fig. 11.10 Convergence trajectories of the first row of weights of the critic network

with

$$x(0) = [-1, 0.5]^\top.$$

The value function is defined as (11.3.2), where $C = -2I$ and A , B , and I are identity matrices of appropriate dimensions. Discretization of the value function using Euler and trapezoidal methods with the sampling time $\Delta t = 10^{-3}$. According to (11.3.40), in order to obtain good approximations, the number of hidden neurons is chosen as 10. The structure of the model NN is chosen as 2–8–1. The structures of the critic networks for upper and lower value functions both are chosen as 2–8–1. For the upper value functions, the structures of the U action network and the W action network are chosen as 3–8–1 and 5–8–1, respectively. For the lower value functions, the structures of the U action network and the W action network are chosen as 5–8–1 and 3–8–1, respectively. The initial weights of action networks, critic networks, and model network are all set to be random in $[-1, 1]$. It should be mentioned that the model network should be trained first. For the given initial state $x(0) = [-1, 0.5]^\top$, we train the model network for 10,000 steps under the learning rate $\eta_m = 0.005$ to reach the given accuracy of $\varepsilon = 10^{-6}$. After the training of the model network is complete, the weights are kept unchanged. During each iteration step, the critic networks and the action networks are trained for 3000 steps

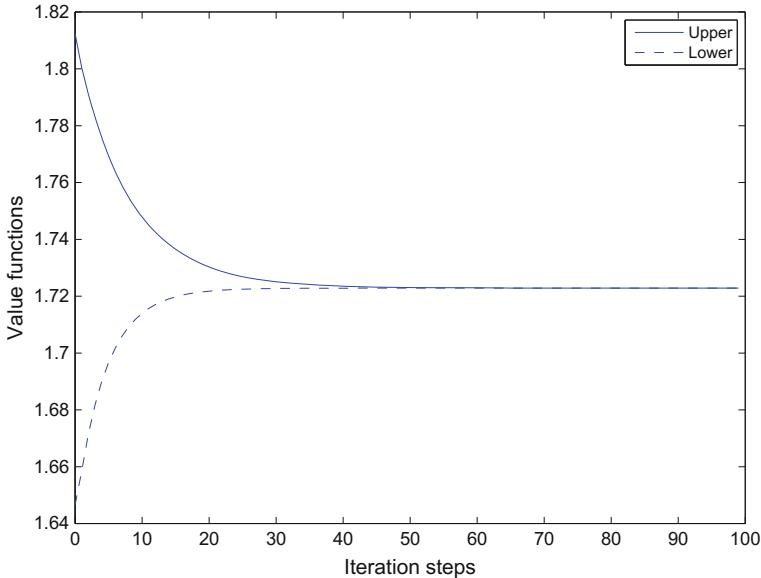


Fig. 11.11 Convergence of value functions

to reach the given accuracy of $\varepsilon = 10^{-6}$. In the training process, the learning rate $\eta_a = \eta_c = 0.005$. The convergence trajectories of the first row of weights of the critic network are shown in Fig. 11.10. The convergence curves of the upper and lower value functions are shown in Fig. 11.11.

We can see that upper and lower value functions converge to the optimal solution of the zero-sum games and the optimal value function exists. The iterative control pairs are also convergent to the optimal. The convergence trajectories of the first row of weights of the U action network are shown in Fig. 11.12. Next, we give the convergent results of the upper iterative control pairs for upper value function.

The convergence curves of the iterative controls w_1 and w_2 for upper value functions are shown in Fig. 11.13a and b, respectively. The convergence curves of the iterative controls u_1 , u_2 and u_3 for upper value functions are shown in Fig. 11.14a–c, respectively. The curves of the optimal controls are shown in Fig. 11.14d. Then, we apply the optimal control pair to the system for $T_f = 500$ time steps and the corresponding state trajectories are given in Fig. 11.15.

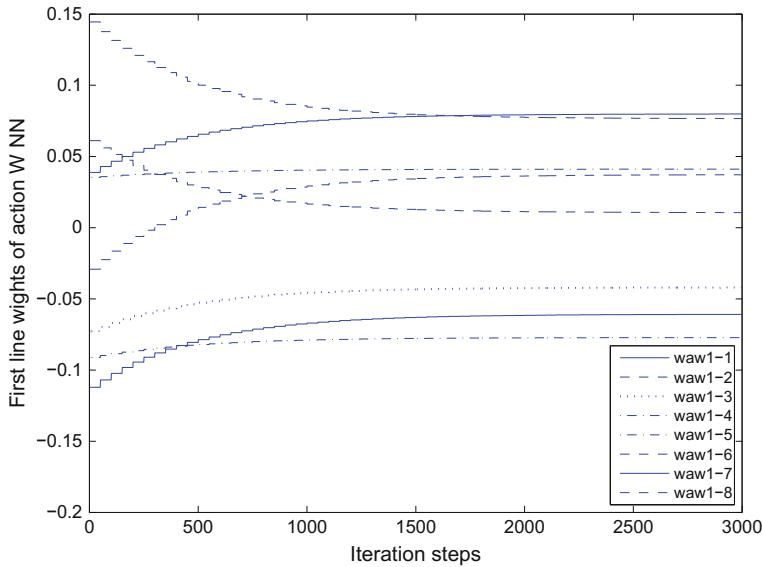


Fig. 11.12 Convergence trajectories of the first row of weights of the U action network

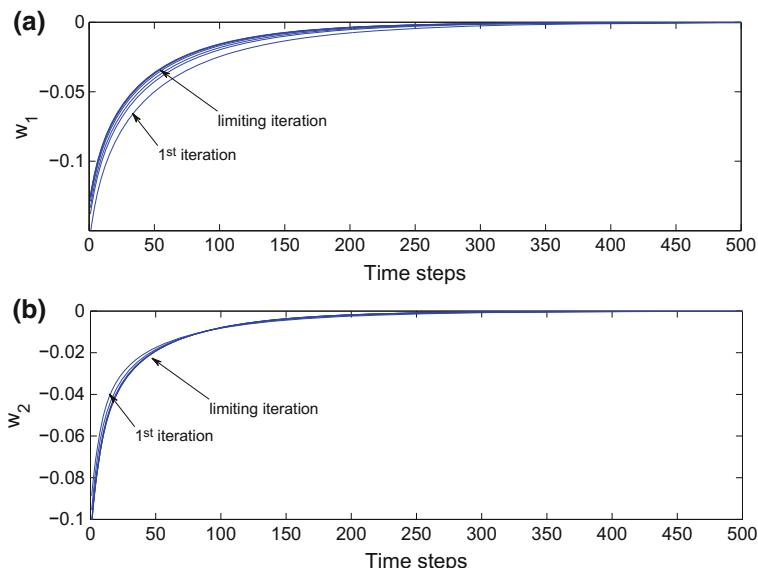


Fig. 11.13 Convergence of upper iterative controls w_1 and w_2

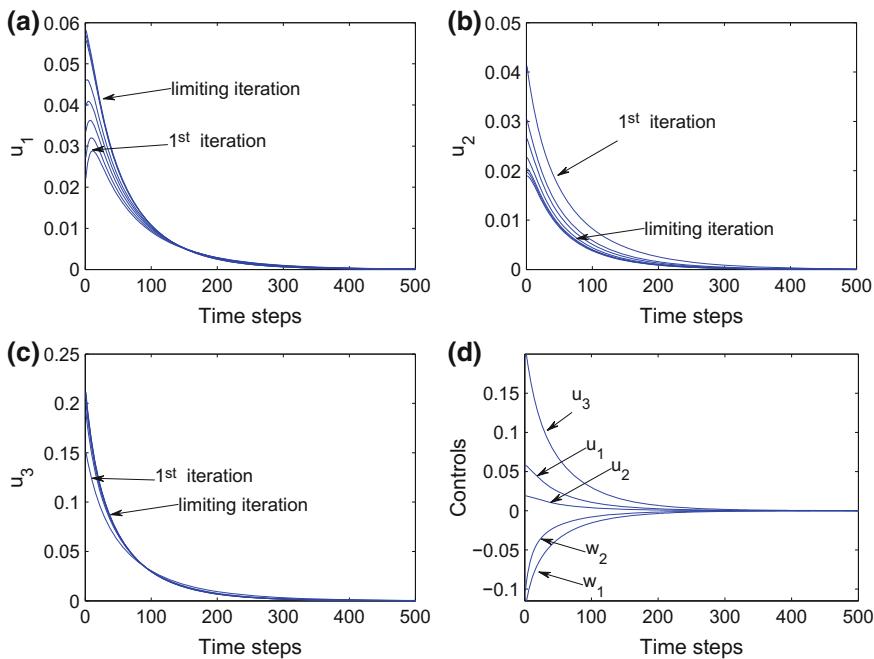


Fig. 11.14 Convergence of upper iterative controls u_1, u_2, u_3 and optimal control trajectories

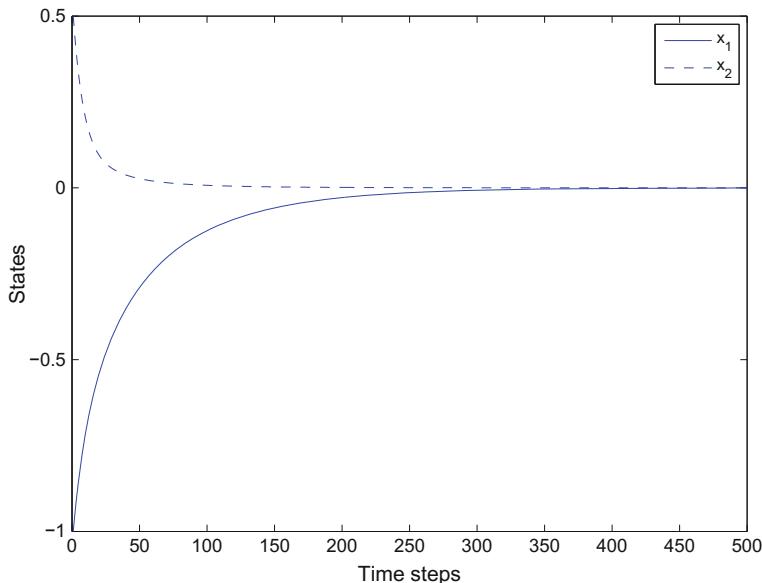


Fig. 11.15 State trajectories

11.4 Synchronous Approximate Optimal Learning for Multi-player Nonzero-Sum Games

Consider the following N -player nonlinear differential game described by

$$\dot{x} = f(x) + \sum_{j=1}^N g_j(x)u_j, \quad (11.4.1)$$

where $x(t) \in \mathbb{R}^n$ is the system state with initial state x_0 , $f(x) \in \mathbb{R}^n$, $g_j(x) \in \mathbb{R}^{n \times m_j}$, and $u_j \in \mathbb{R}^{m_j}$ is the controller or player. We assume that $f(0) = 0$, $f(x)$ and $g_j(x)$ are Lipschitz continuous on a compact set $\Omega \subseteq \mathbb{R}^n$ containing the origin, and the system is stabilizable on Ω . The system dynamics, i.e., $f(x)$ and $g_j(x)$, $j = 1, 2, \dots, N$, are assumed to be unknown.

Define the infinite horizon cost functions associated with each player as

$$\begin{aligned} J_i(x_0, u_1, \dots, u_N) &= \int_0^\infty \left(x^\top Q_i x + \sum_{j=1}^N u_j^\top R_{ij} u_j \right) dt \\ &\triangleq \int_0^\infty r_i(x, u_1, \dots, u_N) dt, \quad i \in \mathbb{N}, \end{aligned} \quad (11.4.2)$$

where $Q_i \in \mathbb{R}^{n \times n}$ and $R_{ij} \in \mathbb{R}^{m_j \times m_j}$ are symmetric positive definite matrices.

The value functions associated with admissible control policies $\mu_i(x) \in \mathcal{A}(\Omega)$ are defined as

$$\begin{aligned} V_i(x(t), \mu_1, \dots, \mu_N) &= \int_t^\infty \left(x^\top Q_i x + \sum_{j=1}^N \mu_j^\top R_{ij} \mu_j \right) dt \\ &\triangleq \int_t^\infty r_i(x(\tau), \mu_1, \dots, \mu_N) d\tau, \quad i \in \mathbb{N}. \end{aligned} \quad (11.4.3)$$

It is desirable to find the optimal admissible control vector $\{\mu_1^*, \dots, \mu_N^*\}$ such that the cost functions (11.4.2) are minimized. The control vector $\{\mu_1^*, \dots, \mu_N^*\}$ corresponds to the Nash equilibrium of the differential game.

Definition 11.4.1 ([9]) An N -tuple of policies $\{\mu_1^*, \dots, \mu_N^*\}$ with $\mu_i^* \in \mathcal{A}(\Omega)$ is said to constitute a Nash equilibrium for an N -player game, if the following N inequalities are satisfied

$$J_i(\mu_1^*, \dots, \mu_i^*, \dots, \mu_N^*) \leq J_i(\mu_1^*, \dots, \mu_i, \dots, \mu_N^*), \quad i \in \mathbb{N}.$$

Assuming that the value functions (11.4.3) are continuously differentiable, the infinitesimal version of (11.4.3) are

$$0 = r_i(x, \mu_1, \dots, \mu_N) + (\nabla V_i)^\top \left(f(x) + \sum_{j=1}^N g_j(x) \mu_j \right), \quad i \in \mathbb{N}, \quad (11.4.4)$$

with $V_i(0) = 0$. Define the Hamiltonians as

$$\begin{aligned} H_i(x, \nabla V_i, \mu_1, \dots, \mu_N) &= r_i(x, \mu_1, \dots, \mu_N) \\ &+ (\nabla V_i)^\top \left(f(x) + \sum_{j=1}^N g_j(x) \mu_j \right), \quad i \in \mathbb{N}. \end{aligned} \quad (11.4.5)$$

The optimal value functions $V_i^*(x_0)$ are defined as

$$V_i^*(x_0) = \min_{\mu_i \in \mathcal{A}(\Omega)} \int_0^\infty \left(x^\top Q_i x + \sum_{j=1}^N \mu_j^\top R_{ij} \mu_j \right) d\tau, \quad i \in \mathbb{N}.$$

Then, the associated state feedback control policies can be obtained by

$$\frac{\partial H_i}{\partial \mu_i} = 0 \Rightarrow \mu_i(x) = -\frac{1}{2} R_{ii}^{-1} g_i^\top(x) \nabla V_i^*, \quad i \in \mathbb{N}. \quad (11.4.6)$$

Therefore, the coupled HJ equations can be written as

$$\begin{aligned} 0 &= x^\top Q_i x + (\nabla V_i^*)^\top f(x) - \frac{1}{2} (\nabla V_i^*)^\top \sum_{j=1}^N g_j(x) R_{jj}^{-1} g_j^\top(x) \nabla V_j^* \\ &+ \frac{1}{4} \sum_{j=1}^N (\nabla V_j^*)^\top g_j(x) R_{jj}^{-1} R_{ij} R_{jj}^{-1} g_j^\top(x) \nabla V_j^*, \quad i \in \mathbb{N} \end{aligned} \quad (11.4.7)$$

with $V_i^*(0) = 0$.

The coupled HJ equations will reduce to the well-known coupled algebraic Riccati equations for the linear quadratic regulator problem. However, the coupled HJ equations cannot generally be solved due to its nonlinear nature. In the next section, we will present an online PI algorithm with convergence analysis for the N -player nonzero-sum game.

11.4.1 Derivation and Convergence Analysis

In this section, an PI algorithm (Algorithm 11.4.1) [26] is presented to solve the coupled HJ equations.

For linear systems, it is shown that the Nash equilibrium of nonzero-sum games can be determined by a quasi-Newton method [28]. However, this property has not

Algorithm 11.4.1 PI for N -player non-zero-sum games

Step 1. Start with an initial admissible control vector $\mu^0 = \{\mu_1^0(x), \dots, \mu_N^0(x)\}$, set $V_i^0(\cdot) = 0$, and let $k = 1$.

Step 2. Policy Evaluation: With the policies $\{\mu_1^{k-1}(x), \dots, \mu_N^{k-1}(x)\}$, solve the following N nonlinear Lyapunov equations

$$0 = r_i(x, \mu_1^{k-1}, \dots, \mu_N^{k-1}) + (\nabla V_i^k)^\top \left(f(x) + \sum_{j=1}^N g_j(x) \mu_j^{k-1} \right), \quad V_i^k(0) = 0, \quad i \in \mathbb{N}. \quad (11.4.8)$$

Step 3. Policy Improvement: Update the N -tuple of control policies simultaneously by

$$\mu_i^k(x) = \arg \min_{\mu_i \in \mathcal{A}(\Omega)} H_i(x, \nabla V_i^k, \mu_1, \dots, \mu_N) = -\frac{1}{2} R_{ii}^{-1} g_i^\top(x) \nabla V_i^k, \quad i \in \mathbb{N}. \quad (11.4.9)$$

Step 4. If $\max \{\|V_1^k - V_1^{k-1}\|, \dots, \|V_N^k - V_N^{k-1}\|\} \leq \varepsilon$ (ε is a small positive number), stop and obtain the approximate optimal control vector $\mu^k = \{\mu_1^k(x), \dots, \mu_N^k(x)\}$; else, set $k = k + 1$, and go back to Step 2.

been demonstrated for nonlinear systems [26, 28]. In this chapter, we will prove the convergence of the online PI algorithm for N -player nonzero-sum games with nonlinear dynamics. It can also be shown that the online PI algorithm for N -player nonzero-sum game is the quasi-Newton method. The analysis is based on the work of [28, 34].

Consider a Banach space $\mathbb{V} \subset \{V(x): \Omega \rightarrow \mathbb{R}, V(0) = 0\}$ with a norm $\|\cdot\|_\Omega$. Define a mapping $\mathcal{G}_i: \underbrace{\mathbb{V} \times \mathbb{V} \times \dots \times \mathbb{V}}_N \rightarrow \mathbb{V}$ as follows:

$$\begin{aligned} \mathcal{G}_i &= x^\top Q_i x + (\nabla V_i)^\top f(x) - \frac{1}{2} (\nabla V_i)^\top \sum_{j=1}^N g_j(x) R_{jj}^{-1} g_j^\top(x) \nabla V_j \\ &\quad + \frac{1}{4} \sum_{j=1}^N (\nabla V_j)^\top g_j(x) R_{jj}^{-1} R_{ij} R_{jj}^{-1} g_j^\top(x) \nabla V_j, \quad i \in \mathbb{N}. \end{aligned} \quad (11.4.10)$$

A mapping $\mathcal{T}_i: \mathbb{V} \rightarrow \mathbb{V}$ is defined as

$$\mathcal{T}_i V_i = V_i - (\mathcal{G}'_{iV_i})^{-1} \mathcal{G}_i, \quad i \in \mathbb{N}, \quad (11.4.11)$$

where \mathcal{G}'_{iV_i} denotes the Fréchet derivative of \mathcal{G}_i taken with respect to V_i . Since the Fréchet derivative is difficult to compute directly, we introduce the following Gâteaux derivative [34, 36].

Definition 11.4.2 Let $\mathcal{G}: \mathbb{U}(V) \subseteq \mathbb{X} \rightarrow \mathbb{Y}$ be a given map, where \mathbb{X} and \mathbb{Y} are Banach spaces, and $\mathbb{U}(V)$ denotes a neighborhood of V . The map \mathcal{G} is Gâteaux differentiable at V if and only if there exists a bounded linear operator $\mathcal{L}: \mathbb{X} \rightarrow \mathbb{Y}$ such that $\mathcal{G}(V + sM) - \mathcal{G}(V) = s\mathcal{L}(M) + o(s)$, $s \rightarrow 0$, for all M with $\|M\|_\Omega = 1$.

and all real numbers s in some neighborhood of zero, where $\lim_{s \rightarrow 0} (\mathcal{G}(V + sM) - \mathcal{G}(V))/s = 0$. \mathcal{L} is called the Gâteaux derivative of \mathcal{G} at V . Then, the Gâteaux derivative at V is defined as

$$\mathcal{L}(M) = \lim_{s \rightarrow 0} \frac{\mathcal{G}(V + sM) - \mathcal{G}(V)}{s}. \quad (11.4.12)$$

To compute the Fréchet derivative, we introduce the following Lemma 11.4.1 (Chap. 4.2, [36]).

Lemma 11.4.1 *If the Gâteaux derivative \mathcal{L} exists in some neighborhood of V , and if \mathcal{L} is continuous at V , then $\mathcal{G}' = \mathcal{L}$ is also a Fréchet derivative at V .*

Therefore, we can compute the Fréchet derivative \mathcal{G}'_{iV_i} which is given in the following lemma by (11.4.12).

Lemma 11.4.2 *Let \mathcal{G}_i be a mapping defined in (11.4.10). Then, $\forall V_i \in \mathbb{V}$, the Fréchet differential of \mathcal{G}_i at V_i is*

$$\mathcal{G}'_{iV_i} M = \mathcal{L}_{iV_i}(M) = (\nabla M)^\top f - \frac{1}{2} (\nabla M)^\top \sum_{j=1}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j.$$

Proof First, according to the definition of \mathcal{G}_i in (11.4.10), $\forall V_i \in \mathbb{V}$, we have

$$\begin{aligned} & \mathcal{G}_i(V_i + sM) - \mathcal{G}_i(V_i) \\ &= x^\top Q_i x - \frac{1}{4} (\nabla(V_i + sM))^\top g_i R_{ii}^{-1} g_i^\top \nabla(V_i + sM) + (\nabla(V_i + sM))^\top f \\ & \quad - \frac{1}{2} (\nabla(V_i + sM))^\top \sum_{j=1, j \neq i}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j \\ & \quad + \frac{1}{4} \sum_{j=1, j \neq i}^N (\nabla V_j)^\top g_j R_{jj}^{-1} R_{ij} R_{jj}^{-1} g_j^\top \nabla V_j \\ & \quad - \left(x^\top Q_i x - \frac{1}{4} (\nabla V_i)^\top g_i R_{ii}^{-1} g_i^\top \nabla V_i + (\nabla V_i)^\top f \right. \\ & \quad \left. - \frac{1}{2} (\nabla V_i)^\top \sum_{j=1, j \neq i}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j + \frac{1}{4} \sum_{j=1, j \neq i}^N (\nabla V_j)^\top g_j R_{jj}^{-1} R_{ij} R_{jj}^{-1} g_j^\top \nabla V_j \right) \\ &= s(\nabla M)^\top f - \frac{s}{4} (\nabla M)^\top g_i R_{ii}^{-1} g_i^\top \nabla V_i - \frac{s}{4} (\nabla V_i)^\top g_i R_{ii}^{-1} g_i^\top \nabla M \\ & \quad - \frac{s^2}{4} (\nabla M)^\top g_i R_{ii}^{-1} g_i^\top \nabla M - \frac{s}{2} (\nabla M)^\top \sum_{j=1, j \neq i}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j \\ &= s(\nabla M)^\top f - \frac{s}{2} (\nabla M)^\top \sum_{j=1}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j - \frac{s^2}{4} (\nabla M)^\top g_i R_{ii}^{-1} g_i^\top \nabla M. \end{aligned}$$

Therefore, the Gâteaux differential at V_i is

$$\begin{aligned}
\mathcal{L}_{iV_i}(M) &= \lim_{s \rightarrow 0} \frac{\mathcal{G}_i(V_i + sM) - \mathcal{G}_i(V_i)}{s} \\
&= (\nabla M)^\top f - \frac{1}{2} (\nabla M)^\top \sum_{j=1}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j. \tag{11.4.13}
\end{aligned}$$

Next, we will prove that \mathcal{L}_{iV_i} is continuous at V_i . For any $M_0 \in \mathbb{V}$, we have

$$\begin{aligned}
&\|\mathcal{L}_{iV_i}(M) - \mathcal{L}_{iV_i}(M_0)\|_\Omega \\
&= \left\| (\nabla(M - M_0))^\top f - \frac{1}{2} (\nabla(M - M_0))^\top \sum_{j=1}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j \right\|_\Omega \\
&\leq \|(\nabla(M - M_0))^\top f\|_\Omega + \left\| \frac{1}{2} (\nabla(M - M_0))^\top \sum_{j=1}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j \right\|_\Omega \\
&\leq \left(\|f\|_\Omega + \left\| \frac{1}{2} \sum_{j=1}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j \right\|_\Omega \right) \|\nabla(M - M_0)\|_\Omega \\
&\leq \left(\|f\|_\Omega + \left\| \frac{1}{2} \sum_{j=1}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j \right\|_\Omega \right) \alpha \|M - M_0\|_\Omega \\
&\triangleq \Phi \|M - M_0\|_\Omega
\end{aligned}$$

where $\alpha > 0$. Therefore, $\forall \varepsilon > 0$, there exists a $\delta = \varepsilon/\Phi$ such that

$$\|\mathcal{L}_{iV_i}(M) - \mathcal{L}_{iV_i}(M_0)\|_\Omega \leq \Phi \|M - M_0\|_\Omega < \varepsilon,$$

when $\|M - M_0\|_\Omega < \delta$; i.e., \mathcal{L}_{iV_i} is continuous at V_i . Then, according to Lemma 11.4.1, $\mathcal{G}'_{iV_i} M = \mathcal{L}_{iV_i}(M)$ is the Fréchet differential as in (11.4.13), and $\mathcal{G}'_{iV_i} = \mathcal{L}_{iV_i}$ is the Fréchet derivative. This completes the proof of the lemma.

With the results in Lemma 11.4.2, we can prove that the online PI algorithm is mathematically equivalent to the quasi-Newton's iteration in a Banach space \mathbb{V} .

Theorem 11.4.1 *Let \mathcal{T}_i be a mapping defined in (11.4.11). Then, the iteration between (11.4.8) and (11.4.9) is equivalent to the following quasi-Newton's iteration*

$$V_i^{k+1} = \mathcal{T}_i V_i^k = V_i^k - \left(\mathcal{G}'_{iV_i^k} \right)^{-1} \mathcal{G}_i, \quad k = 0, 1, \dots \tag{11.4.14}$$

Proof According to (11.4.9) and Lemma 11.4.2, we have

$$\begin{aligned}
\mathcal{G}'_{iV_i^k} V_i^{k+1} &= (\nabla V_i^{k+1})^\top f - \frac{1}{2} (\nabla V_i^{k+1})^\top \sum_{j=1}^N g_j R_{jj}^{-1} g_j^\top \nabla V_j^k \\
&= (\nabla V_i^{k+1})^\top \left(f - \sum_{j=1}^N g_j \frac{1}{2} R_{jj}^{-1} g_j^\top \nabla V_j^k \right) \\
&= (\nabla V_i^{k+1})^\top \left(f + \sum_{j=1}^N g_j \mu_j^k \right),
\end{aligned}$$

and

$$\mathcal{G}'_{iV_i^k} V_i^k = (\nabla V_i^k)^\top \left(f + \sum_{j=1}^N g_j \mu_j^k \right).$$

From (11.4.9) and (11.4.10), we have

$$\mathcal{G}_i = r_i(x, \mu_1^k, \dots, \mu_N^k) + (\nabla V_i^k)^\top \left(f(x) + \sum_{j=1}^N g_j(x) \mu_j^k \right).$$

Thus,

$$\mathcal{G}'_{iV_i^k} V_i^k - \mathcal{G}_i = -r_i(x, \mu_1^k, \dots, \mu_N^k).$$

Considering (11.4.8), we have $\mathcal{G}'_{iV_i^k} V_i^{k+1} = \mathcal{G}'_{iV_i^k} V_i^k - \mathcal{G}_i$, which is the same as (11.4.14). The proof is complete.

Since the PI algorithm for N -player nonzero-sum games is equivalent to the quasi-Newton's iteration, the value function V_i^{k+1} will converge to the optimal value function V_i^* as $k \rightarrow \infty, \forall i \in \mathbb{N}$.

11.4.2 Neural Network Implementation

Based on the above results, we will develop an online synchronous approximate optimal learning algorithm using NN approximation for the multi-player nonzero-sum game with unknown dynamics (see Fig. 11.16). By using a model NN for nonzero-sum game problems, both the internal and drift dynamics are not required. Compared with the algorithm in reference [26], there are fewer parameters to tune in the present algorithm.

A. Model NN Design

In this section, a model NN is used to reconstruct the unknown system dynamics by using input–output data [38]. The unknown nonlinear system dynamics (11.4.1) can be represented as

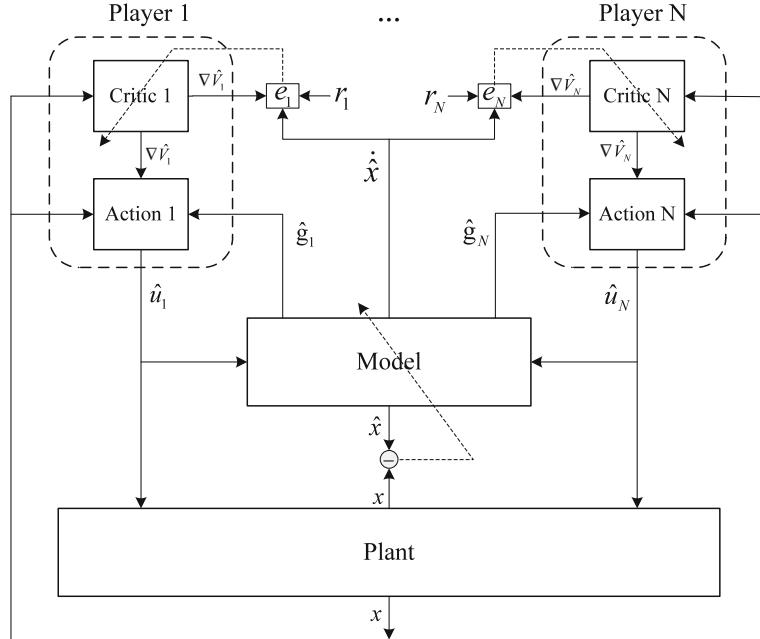


Fig. 11.16 Structural diagram of the online synchronous approximate optimal learning algorithm

$$\dot{x} = Ax + W_f^\top \sigma_f(x) + \varepsilon_f(x) + \sum_{j=1}^N (W_{gj}^\top \sigma_{gj}(x) + \varepsilon_{gj}(x)) u_j, \quad (11.4.15)$$

where A is a stable matrix, $W_f \in \mathbb{R}^{n \times n}$ and $W_{gj} \in \mathbb{R}^{n \times n}$ are the unknown bounded ideal weight matrices, $\sigma_f(\cdot) \in \mathbb{R}^n$ and $\sigma_{gj}(\cdot) \in \mathbb{R}^n$ are the activation functions, and $\varepsilon_f(\cdot) \in \mathbb{R}^n$ and $\varepsilon_{gj}(\cdot) \in \mathbb{R}^n$ are the bounded reconstruction errors, respectively. The model (11.4.15) is obtained by letting $m_j = 1$ in (11.4.1), and it can easily be extended to the general case. Moreover, $f(x)$ is approximated by

$$f(x) = Ax + W_f^\top \sigma_f(x) + \varepsilon_f(x),$$

and $g_j(x)$ is approximated by

$$g_j(x) = W_{gj}^\top \sigma_{gj}(x) + \varepsilon_{gj}(x). \quad (11.4.16)$$

The activation function $\sigma(\cdot)$ is selected as a monotonically increasing function satisfying

$$0 \leq \sigma(x) - \sigma(y) \leq k(x - y), \quad (11.4.17)$$

$\forall x, y \in \mathbb{R}$ and $x \geq y, k > 0$, such as $\sigma(x) = \tanh(x)$.

Assumption 11.4.1 The ideal NN weights are bounded by positive constraints, i.e., $W_f^T W_f \leq \bar{W}_f^T \bar{W}_f$ and $W_{gj}^T W_{gj} \leq \bar{W}_{gj}^T \bar{W}_{gj}$, where \bar{W}_f and \bar{W}_{gj} are known positive definite matrices.

Assumption 11.4.2 The reconstruction errors $\varepsilon_f(x)$ and $\varepsilon_{gj}(x)$ are assumed to be upper bounded by a function of modeling error such that

$$\varepsilon_f^T(x) \varepsilon_f(x) \leq \lambda \tilde{x}^T \tilde{x}, \quad \varepsilon_{gj}^T(x) \varepsilon_{gj}(x) \leq \lambda \tilde{x}^T \tilde{x}$$

where λ is a constant value, and $\tilde{x} = x - \hat{x}$ is the system modeling error (\hat{x} is the estimated system state).

Assumption 11.4.3 The control inputs are bounded, i.e., $\|u_j\| \leq \bar{u}_j$.

Based on (11.4.15), the model NN used to identify the system (11.4.1) is given by

$$\dot{\hat{x}} = A\hat{x} + \hat{W}_f^T \sigma_f(\hat{x}) + \sum_{j=1}^N \hat{W}_{gj}^T \sigma_{gj}(\hat{x}) u_j, \quad (11.4.18)$$

where \hat{W}_f and \hat{W}_{gj} are the estimates of the ideal weight matrices W_f and W_{gj} , respectively. Then, the modeling error dynamics is written as

$$\begin{aligned} \dot{\tilde{x}} &= A\tilde{x} + \tilde{W}_f^T \sigma_f(\hat{x}) + W_f^T [\sigma_f(x) - \sigma_f(\hat{x})] + \varepsilon_f(x) \\ &\quad + \sum_{j=1}^N [\tilde{W}_{gj}^T \sigma_{gj}(\hat{x}) + W_{gj}^T (\sigma_{gj}(x) - \sigma_{gj}(\hat{x})) + \varepsilon_{gj}(x)] u_j, \end{aligned} \quad (11.4.19)$$

where $\tilde{W}_f = W_f - \hat{W}_f$ and $\tilde{W}_g = W_g - \hat{W}_g$.

Theorem 11.4.2 *The modeling error \tilde{x} will asymptotically converge to zero as $t \rightarrow \infty$, if the weight matrices are updated through the following equations*

$$\begin{aligned} \dot{\hat{W}}_f &= \Gamma_f \sigma_f(\hat{x}) \tilde{x}^T, \\ \dot{\hat{W}}_{gj} &= \Gamma_{gj} \sigma_{gj}(\hat{x}) u_j \tilde{x}^T, \quad j = 1, \dots, N \end{aligned} \quad (11.4.20)$$

where Γ_f and Γ_{gj} are symmetric positive definite matrices.

Proof Consider the following Lyapunov function candidate

$$L(x) = \frac{1}{2} \tilde{x}^T \tilde{x} + \frac{1}{2} \text{tr}(\tilde{W}_f^T \Gamma_f^{-1} \tilde{W}_f) + \sum_{j=1}^N \frac{1}{2} \text{tr}(\tilde{W}_{gj}^T \Gamma_{gj}^{-1} \tilde{W}_{gj}). \quad (11.4.21)$$

The time derivative of the Lyapunov function (11.4.21) along the trajectories of the modeling error system (11.4.19) is computed by

$$\dot{L}(t) = \tilde{x}^T \dot{\tilde{x}} + \text{tr}(\tilde{W}_f^T \Gamma_f^{-1} \dot{\tilde{W}}_f) + \sum_{j=1}^N \text{tr}(\tilde{W}_{gj}^T \Gamma_{gj}^{-1} \dot{\tilde{W}}_{gj}). \quad (11.4.22)$$

Substituting (11.4.19) and (11.4.20) into (11.4.22), we have

$$\begin{aligned} \dot{L}(t) &= \tilde{x}^T A \tilde{x} + \tilde{x}^T W_f^T [\sigma_f(x) - \sigma_f(\hat{x})] + \tilde{x}^T \varepsilon_f(x) \\ &\quad + \tilde{x}^T \sum_{j=1}^N [W_{gj}^T (\sigma_{gj}(x) - \sigma_{gj}(\hat{x}))] u_j + \tilde{x}^T \sum_{j=1}^N \varepsilon_{gj}(x) u_j. \end{aligned} \quad (11.4.23)$$

According to (11.4.17), we have

$$\tilde{x}^T W_f^T [\sigma_f(x) - \sigma_f(\hat{x})] \leq \frac{1}{2} \tilde{x}^T W_f^T W_f \tilde{x} + \frac{1}{2} k^2 \tilde{x}^T \tilde{x},$$

and

$$\tilde{x}^T W_{gj}^T [\sigma_{gj}(x) - \sigma_{gj}(\hat{x})] \leq \frac{1}{2} \tilde{x}^T W_{gj}^T W_{gj} \tilde{x} + \frac{1}{2} k^2 \tilde{x}^T \tilde{x}.$$

According to Assumption 11.4.2, we obtain

$$\tilde{x}^T \varepsilon_f(x) \leq \frac{1}{2} \tilde{x}^T \tilde{x} + \frac{1}{2} \lambda \tilde{x}^T \tilde{x} \quad \text{and} \quad \tilde{x}^T \varepsilon_{gj}(x) \leq \frac{1}{2} \tilde{x}^T \tilde{x} + \frac{1}{2} \lambda \tilde{x}^T \tilde{x}.$$

Therefore, (11.4.23) can be upper bounded as

$$\begin{aligned} \dot{L}(t) &\leq \tilde{x}^T A \tilde{x} + \frac{1}{2} \tilde{x}^T W_f^T W_f \tilde{x} + \frac{1}{2} k^2 \tilde{x}^T \tilde{x} + \frac{1}{2} \tilde{x}^T \tilde{x} + \frac{1}{2} \lambda \tilde{x}^T \tilde{x} \\ &\quad + \frac{1}{2} \tilde{x}^T \sum_{j=1}^N u_j W_{gj}^T W_{gj} \tilde{x} + \frac{1}{2} k^2 \sum_{j=1}^N u_j \tilde{x}^T \tilde{x} + \frac{1+\lambda}{2} \sum_{j=1}^N u_j \tilde{x}^T \tilde{x} \\ &= \tilde{x}^T \Xi \tilde{x}, \end{aligned}$$

where

$$\Xi = A + \frac{1}{2} W_f^T W_f + \frac{1}{2} \sum_{j=1}^N u_j W_{gj}^T W_{gj} + \left(\frac{1}{2} + \frac{1}{2} \lambda + \frac{1}{2} k^2 + \frac{1+\lambda}{2} \sum_{j=1}^N u_j + \frac{1}{2} k^2 \sum_{j=1}^N u_j \right) I_n.$$

If A is selected such that $\mathcal{E} \leq 0$, it can be concluded that $\dot{L}(t) \leq 0$, and then $\tilde{x}(t) \rightarrow 0$ as $t \rightarrow \infty$. The proof is complete.

The model NN is a stable and asymptotic identifier, and thus, the exact knowledge of the system dynamics can be removed. Consequently, we can obtain the following model NN

$$\dot{x} = Ax + W_f^T \sigma_f(x) + \sum_{j=1}^N (W_{gj}^T \sigma_{gj}(x)) u_j. \quad (11.4.24)$$

B. Online Synchronous Approximate Optimal Learning Algorithm

Assume that the value functions $V_i(x)$ are continuously differentiable. Then, the value functions $V_i(x)$ are approximated on a compact set Ω by feedforward NNs as

$$V_i(x) = W_c^{i\top} \phi_i(x) + \varepsilon_i(x), \quad i \in \mathbb{N}, \quad (11.4.25)$$

where $W_c^i \in \mathbb{R}^K$ are the unknown bounded ideal weights ($\|W_c^i\| \leq \bar{W}_c^i$), $\phi_i(x) \in \mathbb{R}^K$ are the activation functions, K is the number of neurons in the hidden layer, and $\varepsilon_i(x) \in \mathbb{R}$ are the bounded NN approximation errors. The activation functions can be selected as polynomial, sigmoid, tanh, etc.

The derivatives of value functions with respect to x are represented as

$$\frac{\partial V_i(x)}{\partial x} = \left(\frac{\partial \phi_i(x)}{\partial x} \right)^T W_c^i + \frac{\partial \varepsilon_i(x)}{\partial x} = \nabla \phi_i^T W_c^i + \nabla \varepsilon_i, \quad i \in \mathbb{N}, \quad (11.4.26)$$

where $\nabla \phi_i \in \mathbb{R}^{K \times n}$ and $\nabla \varepsilon_i \in \mathbb{R}^n$ are bounded gradients of the activation functions and approximation errors, respectively. As the number of neurons in the hidden layer $K \rightarrow \infty$, the approximation errors $\varepsilon_i \rightarrow 0$, and the derivatives $\nabla \varepsilon_i \rightarrow 0$ uniformly. The approximation errors ε_i and the derivatives $\nabla \varepsilon_i$ are bounded by constants on a compact set Ω . Thus, (11.4.4) can be rewritten as

$$0 = r_i(x, \mu_1, \dots, \mu_N) + (W_c^{i\top} \nabla \phi_i + (\nabla \varepsilon_i)^T) \dot{x}, \quad i \in \mathbb{N}. \quad (11.4.27)$$

Since the ideal weights are unknown, the critic NNs can be written in terms of the weight estimates as

$$\hat{V}_i(x) = \hat{W}_c^{i\top} \phi_i(x), \quad i \in \mathbb{N}.$$

To avoid using the knowledge of the system dynamics, the model NN is used to approximate the system dynamics. Then, (11.4.27) can be rewritten as

$$0 = r_i(x, \mu_1, \dots, \mu_N) + (W_c^{i\top} \nabla \phi_i + (\nabla \varepsilon_i)^\top) \left(Ax + W_f^\top \sigma_f(x) + \sum_{j=1}^N (W_{gj}^\top \sigma_{gj}(x)) u_j \right).$$

The approximate Hamiltonians can be derived as follows:

$$\begin{aligned} H_i(x, \hat{W}_c^i, u_1, \dots, u_N) &= r_i(x, \mu_1, \dots, \mu_N) + \hat{W}_c^{i\top} \nabla \phi_i(x) \\ &\quad \times \left(Ax + W_f^\top \sigma_f(x) + \sum_{j=1}^N (W_{gj}^\top \sigma_{gj}(x)) u_j \right) \\ &\triangleq e_i, \quad i \in \mathbb{N}. \end{aligned}$$

It is desired to design \hat{W}_c^i to minimize the following objective functions

$$E_i(\hat{W}_c^i) = \frac{1}{2} e_i^2, \quad i \in \mathbb{N}.$$

The tuning law for the critic NN weights is a standard steepest descent algorithm, which is given by

$$\dot{\hat{W}}_i = -\eta_i \left(\frac{\partial E_i}{\partial \hat{W}_c^i} \right)^\top = -\eta_i \theta_i (r_i + \hat{W}_c^{i\top} \theta_i), \quad i \in \mathbb{N}, \quad (11.4.28)$$

where $\eta_i > 0$ is the learning rate of the critic NN, and

$$\theta_i = \nabla \phi_i \left(Ax + W_f^\top \sigma_f(x) + \sum_{j=1}^N (W_{gj}^\top \sigma_{gj}(x)) u_j \right).$$

There exists a positive constant θ_{iM} such that

$$\|\theta_i\| \leq \theta_{iM}.$$

According to (11.4.5), the Hamiltonians become

$$\begin{aligned} H_i(x, W_c^i, u_1, \dots, u_N) &= r_i(x, \mu_1, \dots, \mu_N) + W_c^{i\top} \nabla \phi_i(x) \\ &\quad \times \left(Ax + W_f^\top \sigma_f(x) + \sum_{j=1}^N (W_{gj}^\top \sigma_{gj}(x)) u_j \right) \\ &\triangleq e_{Hi}, \quad i \in \mathbb{N}, \end{aligned}$$

where the residual errors due to the NN approximation are

$$e_{Hi} = -(\nabla \varepsilon_i)^\top \left(Ax + W_f^\top \sigma_f(x) + \sum_{j=1}^N (W_{gj}^\top \sigma_{gj}(x)) u_j \right), \quad i \in \mathbb{N}.$$

Define the weight estimation errors of the critic NN as $\tilde{W}_c^i = W_c^i - \hat{W}_c^i$. Thus, we can have the following error dynamics

$$\dot{\tilde{W}}_i = \eta_i \theta_i (e_{Hi} - \tilde{W}_c^{i\top} \theta_i). \quad (11.4.29)$$

The PE condition is needed to tune the critic NNs ensuring $\|\theta_i\| \geq \theta_{im}$, $i \in \mathbb{N}$, where θ_{im} are positive constants. In order to satisfy the PE condition, a small exploratory signal can be injected into the system or reset system states.

The objective of the action NN is to select a policy which minimizes the current estimate of the value functions in (11.4.25). Since a closed-form expression for the optimal control is available, there is no need for training action NNs. Substituting $g_i(x)$ and $\nabla V_i(x)$, the expressions in (11.4.6) can be rewritten as

$$u_i = -\frac{1}{2} R_{ii}^{-1} (W_{gi}^\top \sigma_{gi} + \varepsilon_{gj})^\top (\nabla \phi_i^\top W_c^i + \nabla \varepsilon_i), \quad i \in \mathbb{N}.$$

The approximate control policies \hat{u}_i are given by

$$\hat{u}_i = -\frac{1}{2} R_{ii}^{-1} (W_{gi}^\top \sigma_{gi})^\top \nabla \phi_i^\top \hat{W}_c^i, \quad i \in \mathbb{N}. \quad (11.4.30)$$

The UUB stability of the closed-loop system can be proved based on Lyapunov approach.

Theorem 11.4.3 *Consider the system described by (11.4.24). The weight-updating laws of the critic NNs are given by (11.4.28), and the control policies are updated by (11.4.30). The initial weights of the critic NNs are chosen to generate an initial admissible control pair. Then, the weight estimation errors of the critic NNs are UUB.*

Proof Choose the following Lyapunov function

$$L(x) = \sum_{i=1}^N \frac{1}{2\eta_i} \text{tr}(\tilde{W}_c^{i\top} \tilde{W}_c^i) \triangleq \sum_{i=1}^N S_i(t).$$

The time derivatives of the Lyapunov functions along the trajectories of the error dynamics (11.4.29) are computed as

$$\dot{L}_i(x) = \frac{1}{\eta_i} \text{tr}\{\tilde{W}_c^{i\top} \dot{\tilde{W}}_i\} = \frac{1}{\eta_i} \text{tr}\{\tilde{W}_c^{i\top} [\eta_i \theta_i (e_{Hi} - \tilde{W}_c^{i\top} \theta_i)]\}, \quad i \in \mathbb{N}.$$

According to $\theta_{im} \leq \|\theta_i\| \leq \theta_{iM}$, we have

$$\dot{L}_i(x) \leq -\left(\theta_{im}^2 - \frac{\eta_i}{2}\theta_{iM}^2\right)\|\tilde{W}_c^i\|^2 + \frac{e_{Hi}^2}{2\eta_i}, \quad i \in \mathbb{N}.$$

If the learning rates η_i are selected to satisfy

$$\eta_i \leq \frac{2\theta_{im}^2}{\theta_{iM}^2}, \quad i \in \mathbb{N},$$

and given that the following inequalities hold

$$\|\tilde{W}_c^i\| > \sqrt{\frac{e_{Hi}^2}{\eta_i(2\theta_{im}^2 - \eta_i\theta_{iM}^2)}}, \quad i \in \mathbb{N},$$

then

$$\dot{L}_i(x) < 0.$$

Using Lyapunov theory, it can be concluded that the weight estimation errors of the critic NNs $\|\tilde{W}_c^i\|$ are UUB. This completes the proof of the theorem.

Theorem 11.4.4 *Consider the system described by (11.4.24). The weight-updating laws of the critic NNs are given by (11.4.28), and the control policies are updated by (11.4.30). The initial weights of the critic NNs are chosen to generate an initial admissible control pair. For some initial condition x_0 , there exists a time $T(x_0, B)$ such that $x(t)$ is UUB, where the bound B is given by*

$$\begin{aligned} \|x(t)\| &\leq \max \left\{ \sqrt{\frac{\xi_1}{\lambda_{\min}(Q_1)}}, \dots, \sqrt{\frac{\xi_N}{\lambda_{\min}(Q_N)}} \right\} \\ &\triangleq B, \quad t \geq t_0 + T, \end{aligned}$$

where $\xi_i \in \mathbb{R}^+, i \in \mathbb{N}$.

Proof To show the stability of the approximate control policies in (11.4.30), we take the derivatives of $V_i(x)$ along the trajectories generated by the approximate control policies \hat{u}_i as

$$\dot{V}_i(t) = (\nabla V_i(x))^\top \left(f(x) + \sum_{j=1}^N g_j(x) \hat{u}_j \right), \quad i \in \mathbb{N}. \quad (11.4.31)$$

Considering (11.4.7), we have

$$\begin{aligned} (\nabla V_i(x))^\top f &= -x^\top Q_i x + \frac{1}{2} (\nabla V_i)^\top \sum_{j=1}^N g_j(x) R_{jj}^{-1} g_j^\top(x) \nabla V_j \\ &\quad - \frac{1}{4} \sum_{j=1}^N (\nabla V_j)^\top g_j(x) R_{jj}^{-1} R_{ij} R_{jj}^{-1} g_j^\top(x) \nabla V_j. \end{aligned}$$

Substituting $(\nabla V_i(x))^\top f$ into (11.4.31) yields

$$\begin{aligned} \dot{V}_i(t) &= -x^\top Q_i x + (\nabla V_i)^\top \left(\sum_{j=1}^N g_j(x) \hat{u}_j \right) + \frac{1}{2} (\nabla V_i)^\top \sum_{j=1}^N g_j(x) R_{jj}^{-1} g_j^\top(x) \nabla V_j \\ &\quad - \frac{1}{4} \sum_{j=1}^N (\nabla V_j)^\top g_j(x) R_{jj}^{-1} R_{ij} R_{jj}^{-1} g_j^\top(x) \nabla V_j. \end{aligned}$$

Considering (11.4.6), we have

$$\begin{aligned} \dot{V}_i(t) &= -x^\top Q_i x - \frac{1}{4} \sum_{j=1}^N (\nabla V_j)^\top g_j(x) R_{jj}^{-1} R_{ij} R_{jj}^{-1} g_j^\top(x) \nabla V_j \\ &\quad - (\nabla V_i)^\top \left(\sum_{j=1}^N g_j(x) (u_j - \hat{u}_j) \right). \end{aligned} \quad (11.4.32)$$

Substituting (11.4.6), (11.4.16), (11.4.26), and (11.4.30) into (11.4.32) yields

$$\begin{aligned} \dot{V}_i(t) &= -x^\top Q_i x - \frac{1}{4} \sum_{j=1}^N (\nabla V_j)^\top g_j(x) R_{jj}^{-1} R_{ij} R_{jj}^{-1} g_j^\top(x) \nabla V_j \\ &\quad + \frac{1}{2} (\nabla \phi_i^\top W_c^i + \nabla \varepsilon_i)^\top \left[\sum_{j=1}^N (W_{gj}^\top \sigma_{gj}(x)) R_{jj}^{-1} \right. \\ &\quad \times \left. (\sigma_{gj}^\top W_{gj} \nabla \phi_j^\top \tilde{W}_j + \sigma_{gj}^\top W_{gj} \nabla \varepsilon_j) \right] \\ &\triangleq -x^\top Q_i x - \frac{1}{4} \sum_{j=1}^N (\nabla V_j)^\top g_j(x) R_{jj}^{-1} R_{ij} R_{jj}^{-1} g_j^\top(x) \nabla V_j + \Lambda_i, \end{aligned}$$

where $\tilde{W}_j = W_j - \hat{W}_j$. According to the boundedness assumption on W_{gj} , ε_{gj} , W_c^i , ε_i , and \tilde{W}_c^i is UUB, the term Λ_i has an upper bound, i.e.,

$$\begin{aligned}
A_i &\leq \left\| \frac{1}{2} (\nabla \phi_i^\top W_c^i + \nabla \varepsilon_i)^\top \left[\sum_{j=1}^N (W_{gj}^\top \sigma_{gj}(x)) R_{jj}^{-1} \right. \right. \\
&\quad \times \left. \left. (\sigma_{gj}^\top W_{gj} \nabla \phi_j^\top \tilde{W}_j + \sigma_{gj}^\top W_{gj} \nabla \varepsilon_j) \right] \right\| \\
&\leq \xi_i,
\end{aligned} \tag{11.4.33}$$

where $\xi_i \in \mathbb{R}^+$ is a computable constant. Since R_{ij} is a symmetric positive definite matrix, the following term is positive definite

$$\begin{aligned}
&\frac{1}{4} \sum_{j=1}^N (\nabla V_j)^\top g_j(x) R_{jj}^{-1} R_{ij} R_{jj}^{-1} g_j^\top(x) \nabla V_j \\
&= \frac{1}{4} \sum_{j=1}^N (R_{jj}^{-1} g_j^\top(x) \nabla V_j)^\top R_{ij} R_{jj}^{-1} g_j^\top(x) \nabla V_j > 0.
\end{aligned} \tag{11.4.34}$$

Combining (11.4.33) and (11.4.34), we can obtain that $\dot{V}_i(t)$ is upper bounded by

$$\dot{V}_i(t) \leq -x^\top Q_i x + \xi_i \leq -\lambda_{\min}(Q_i) \|x\|^2 + \xi_i.$$

For each value function $V_i(x(t))$, it can be shown that its derivative $\dot{V}_i(t)$ is negative whenever $x(t)$ lies outside the compact set $\Omega_i \triangleq \left\{ x : \|x\| \leq \sqrt{\frac{\xi_i}{\lambda_{\min}(Q_i)}} \right\}$. Denote the compact set Ω_x as

$$\Omega_x \triangleq \left\{ x : \|x\| \leq \max \left\{ \sqrt{\frac{\xi_1}{\lambda_{\min}(Q_1)}}, \dots, \sqrt{\frac{\xi_N}{\lambda_{\min}(Q_N)}} \right\} \right\}.$$

Then, all the derivatives $\dot{V}_i(t)$ are negative whenever $x(t)$ lies outside the compact set Ω_x ; i.e., $\|x(t)\|$ is UUB. If we increase $\lambda_{\min}(Q_i)$, the size of Ω_x can be made smaller. This completes the proof of the theorem.

11.4.3 Simulation Study

In this section, we give a simulation example to demonstrate the effectiveness of the present scheme. This example is constructed by the converse HJB method [23] which can give the optimal value functions and control policies.

Example 11.4.1 Consider a 3-player continuous-time affine nonlinear differential game given by

$$\dot{x} = f(x) + g_1(x)u_1 + g_2(x)u_2 + g_3(x)u_3,$$

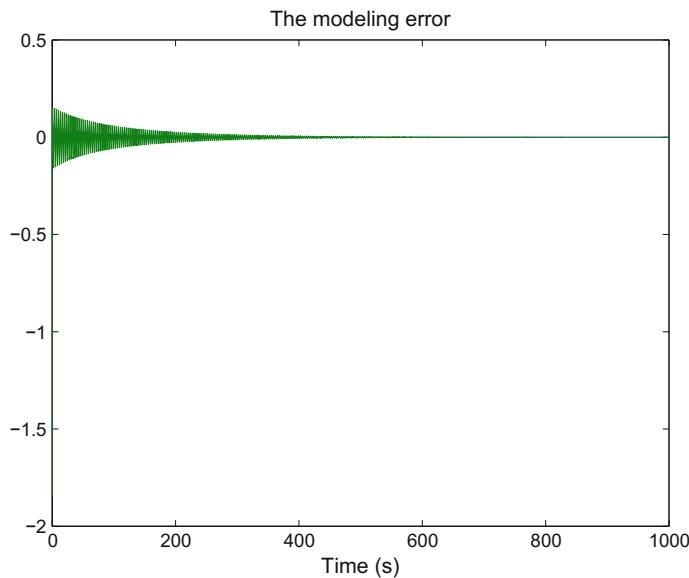


Fig. 11.17 Curve of the modeling error

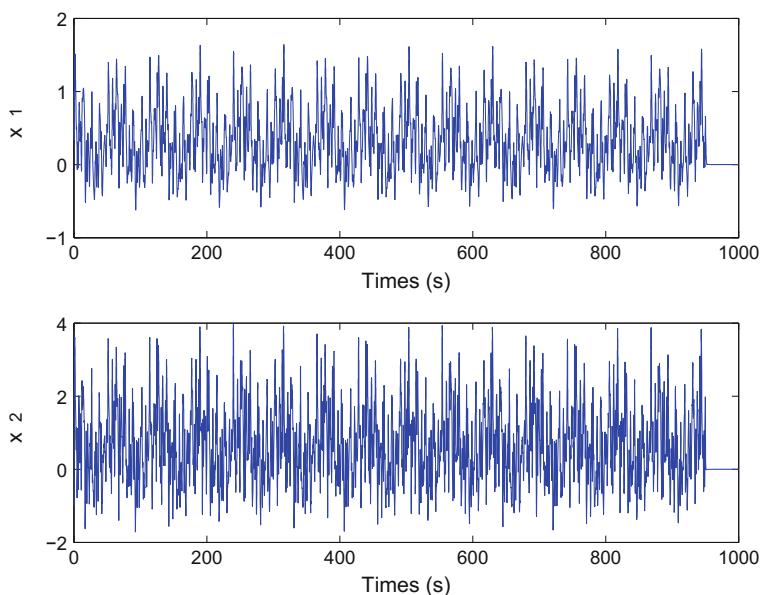


Fig. 11.18 Evolution of the states

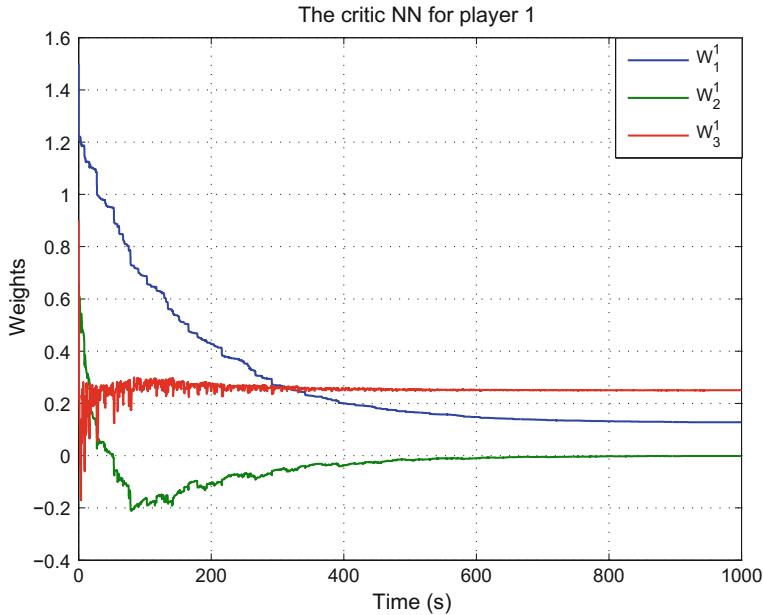


Fig. 11.19 Critic NN weights for player 1

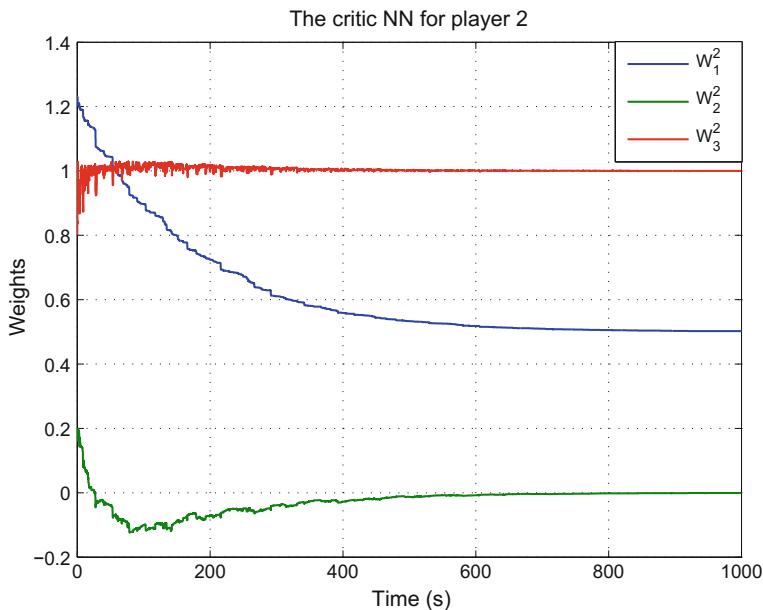


Fig. 11.20 Critic NN weights for player 2

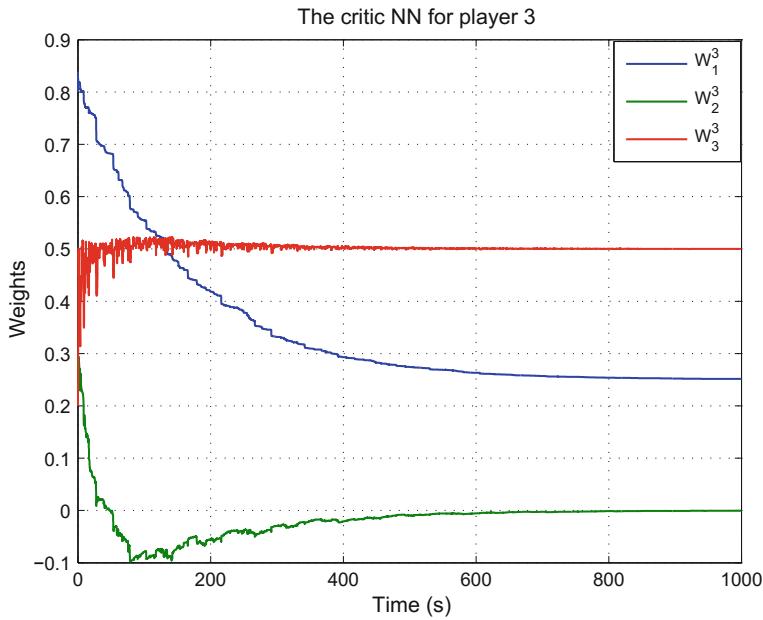


Fig. 11.21 Critic NN weights for player 3

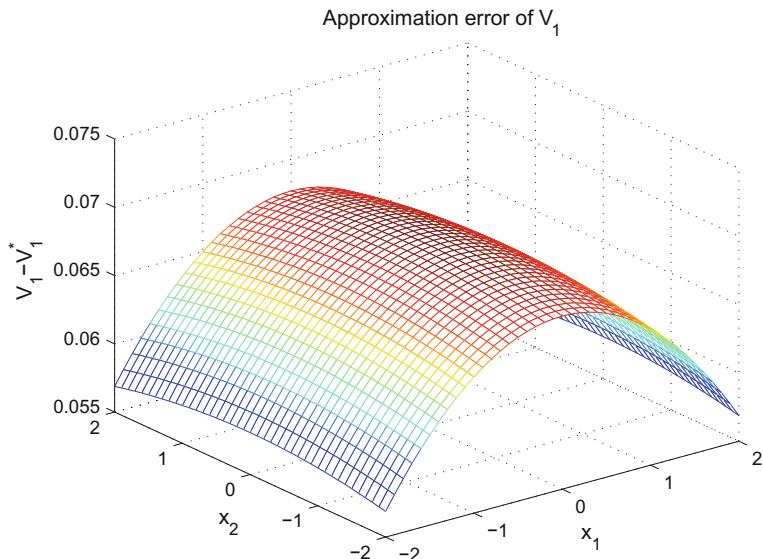


Fig. 11.22 3-D plot of the approximation error of the value function for player 1

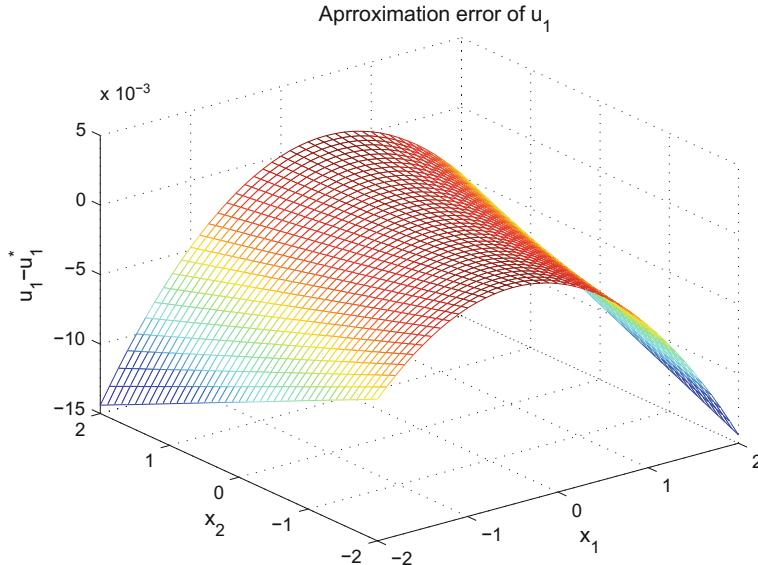


Fig. 11.23 3-D plot of the approximation error of the control policy for player 1

where

$$f(x) = \begin{bmatrix} -2x_1 + x_2 \\ -\frac{1}{2}x_1 - x_2 + x_1^2 x_2 + \frac{1}{4}x_2(\cos(2x_1) + 2)^2 + \frac{1}{4}x_2(\sin(4x_1^2) + 2)^2 \end{bmatrix},$$

$$g_1(x) = \begin{bmatrix} 0 \\ 2x_1 \end{bmatrix}, \quad g_2(x) = \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix}, \quad g_3(x) = \begin{bmatrix} 0 \\ \sin(4x_1^2) + 2 \end{bmatrix}.$$

Select $Q_1 = 0.5I_2$, $Q_2 = 2I_2$, $Q_3 = I_2$, $R_{11} = R_{12} = R_{13} = 0.5$, $R_{21} = R_{22} = R_{23} = 2$, and $R_{31} = R_{32} = R_{33} = 1$. The optimal value functions for the three players are

$$V_1^*(x) = \frac{1}{8}x_1^2 + \frac{1}{4}x_2^2, \quad V_2^*(x) = \frac{1}{2}x_1^2 + x_2^2, \quad V_3^*(x) = \frac{1}{4}x_1^2 + \frac{1}{2}x_2^2.$$

The corresponding optimal control policies for three players are

$$u_1^*(x) = -x_1 x_2, \quad u_2^*(x) = -\frac{1}{2}(\cos(2x_1) + 2)x_2, \quad (11.4.35)$$

and

$$u_3^*(x) = -\frac{1}{2}(\sin(4x_1^2) + 2)x_2. \quad (11.4.36)$$

First, we use a model NN to identify the unknown nonlinear system. The model NN is selected as in (11.4.18) with $A = [-10, 0; 0, -10]$. The activation functions are selected as hyperbolic tangent function $\tanh(\cdot)$. Select the parameters in Theorem 11.4.2 as $\Gamma_f = [1, 0.1; 0.1, 1]$, $\Gamma_{gj} = [1, 0.1; 0.1, 1]$. The curves of modeling error are shown in Fig. 11.17. We can observe that the obtained model NN can reconstruct the unknown nonlinear system successfully.

The activation functions for the critic NNs are selected as $\phi_1(x) = \phi_2(x) = \phi_3(x) = [x_1^2, x_1x_2, x_2^2]^\top$. The critic NN weight vectors for the three players are denoted as $\hat{W}^1 = [\hat{W}_1^1, \hat{W}_2^1, \hat{W}_3^1]^\top$, $\hat{W}^2 = [\hat{W}_1^2, \hat{W}_2^2, \hat{W}_3^2]^\top$, and $\hat{W}^3 = [\hat{W}_1^3, \hat{W}_2^3, \hat{W}_3^3]^\top$. The initial weights of the three critic NNs are randomly selected in $[0, 1.5]$, and the learning rates for the three critic NNs are all 0.1. The initial state is selected as $x_0 = [1, -1]^\top$. A small exploratory signal is used to satisfy the PE condition. After the exploratory signal is turned off at $t = 950$ sec, the states converge to zero, and Fig. 11.18 presents the evolution of the system states. From Figs. 11.19, 11.20 and 11.21, we can observe that the weight vector \hat{W}^1 converges to $[0.1277, -0.0012, 0.2503]^\top$, the weight vector \hat{W}^2 converges to $[0.5022, -0.0009, 1.0002]^\top$, and the weight vector \hat{W}^3 converges to $[0.2516, -0.0007, 0.5001]^\top$ at $t = 1000$ s. For player 1, Fig. 11.22 shows the 3-D plot of the difference between the approximated value function and the optimal one, and Fig. 11.23 shows the 3-D plot of the difference between the approximated control policy and the optimal one. We can find that these errors are close to zero, and other players also have similar results. Thus, the approximate value functions converge to the optimal ones within a small bound.

11.5 Conclusions

In this chapter, we developed some ADP algorithms for differential games of continuous-time systems with unknown dynamics. First, we developed an online model-free integral PI algorithm for two-player zero-sum differential games of continuous-time linear systems. Second, we developed an iterative ADP algorithm for multi-player zero-sum differential games of continuous-time nonlinear systems. Finally, we developed an online synchronous approximate optimal learning algorithm based on PI for multi-player nonzero-sum games of continuous-time nonlinear systems.

References

1. Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* 41(5):779–791
2. Abu-Khalaf M, Lewis FL, Huang J (2006) Policy iterations and the Hamilton-Jacobi-Isaacs equation for H_∞ state feedback control with input saturation. *IEEE Trans Autom Control* 51(12):1989–1995

3. Abu-Khalaf M, Lewis FL, Huang J (2008) Neurodynamic programming and zero-sum games for constrained control systems. *IEEE Trans Neural Netw* 19(7):1243–1252
4. Al-Tamimi A, Abu-Khalaf M, Lewis FL (2007) Adaptive critic designs for discrete-time zero-sum games with application to H_∞ control. *IEEE Trans Syst Man Cybern-Part B: Cybern* 37(1):240–247
5. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2007) Model-free Q-learning designs for linear discrete-time zero-sum games with application to H_∞ control. *Automatica* 43(3):473–481
6. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2008) Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans Syst Man Cybern-Part B: Cybern* 38(4):943–949
7. Bardi M, Capuzzo-Dolcetta I (1997) Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations. Birkhäuser, Boston
8. Basar T, Bernhard P (1995) H_∞ optimal control and related minimax design problems: a dynamic game approach. Birkhäuser, Boston
9. Basar T, Olsder GJ (1999) Dynamic noncooperative game theory. SIAM, Philadelphia
10. Bhasin S, Kamalapurkar R, Johnson M, Vamvoudakis KG, Lewis FL, Dixon WE (2013) A novel actor-critic-identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica* 49(1):82–92
11. Gupta SK (1995) Numerical methods for engineers. Wiley, New York
12. Hecht-Nielsen R (1989) Theory of the backpropagation neural network. In: Proceedings of the international joint conference on neural networks, pp 593–605
13. Jiang Y, Jiang ZP (2012) Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica* 48(10):2699–2704
14. Lee JY, Park JB, Choi YH (2012) Integral Q-learning and explorized policy iteration for adaptive optimal control of continuous-time linear systems. *Automatica* 48(11):2850–2859
15. Lewis FL, Liu D (2012) Reinforcement learning and approximate dynamic programming for feedback control. Wiley, Hoboken
16. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst Mag* 9(3):32–50
17. Li H, Liu D, Wang D (2014) Integral reinforcement learning for linear continuous-time zero-sum games with completely unknown dynamics. *IEEE Trans Autom Sci Eng* 11(3):706–714
18. Liu D, Wei Q (2014) Multi-person zero-sum differential games for a class of uncertain nonlinear systems. *Int J Adapt Control Signal Process* 28(3–5):205–231
19. Liu D, Li H, Wang D (2013) Neural-network-based zero-sum game for discrete-time nonlinear systems via iterative adaptive dynamic programming algorithm. *Neurocomputing* 110:92–100
20. Liu D, Wang D, Yang X (2013) An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs. *Inf Sci* 220:331–342
21. Liu D, Li H, Wang D (2014) Data-based online synchronous optimal learning algorithm for multi-player non-zero-sum games. *IEEE Trans Syst Man Cybern: Syst* 44(8):1015–1027
22. Marks RJ (1991) Introduction to Shannon sampling and interpolation theory. Springer, New York
23. Nevisti V, Prims JA (1996) Constrained nonlinear optimal control: a converse HJB approach. California Institute of Technology, TR96-021
24. Vamvoudakis KG, Lewis FL (2010) Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* 46(5):878–888
25. Vamvoudakis KG, Lewis FL (2011) Online solution of nonlinear two-player zero-sum games using synchronous policy iteration. *Int J Robust Nonlinear Control* 22(13):1460–1483
26. Vamvoudakis KG, Lewis FL (2011) Multi-player non-zero-sum games: online adaptive learning solution of coupled Hamilton-Jacobi equations. *Automatica* 47(8):1556–1569
27. Vrabie D, Lewis FL (2009) Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Netw* 22(3):237–246
28. Vrabie D, Lewis FL (2010) Integral reinforcement learning for online computation of feedback Nash strategies of nonzero-sum differential games. In: Proceedings of the IEEE conference on decision and control, pp 3066–3071

29. Varbie D, Lewis FL (2011) Adaptive dynamic programming for online solution of a zero-sum differential game. *J Control Theory Appl* 9(3):353–360
30. Vrabie D, Pastravanu O, Abu-Khalaf M, Lewis FL (2009) Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica* 45(2):477–484
31. Wang FY, Zhang H, Liu D (2009) Adaptive dynamic programming: an introduction. *IEEE Comput Intell Mag* 4(2):39–47
32. Wang D, Liu D, Wei Q, Zhao D, Jin N (2012) Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. *Automatica* 48(8):1825–1832
33. Werbos PJ (1992) Approximate dynamic programming for real-time control and neural modeling. In: White DA, Sofge DA (eds) *Handbook of intelligent control: neural, fuzzy, and adaptive approaches* (Chapter 13). Van Nostrand Reinhold, New York
34. Wu H, Luo B (2012) Neural network based online simultaneous policy update algorithm for solving the HJI equation in nonlinear H_∞ control. *IEEE Trans Neural Netw Learn Syst* 23(12):1884–1895
35. Wu H, Luo B (2013) Simultaneous policy update algorithms for learning the solution of linear continuous-time H_∞ state feedback control. *Inf Sci* 222(10):472–485
36. Zeidler E (1985) *Nonlinear functional analysis. Fixed point theorems*, vol 1. Springer, New York
37. Zhang H, Luo Y, Liu D (2009) Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints. *IEEE Trans Neural Netw* 20(9):1490–1503
38. Zhang H, Cui L, Zhang X, Luo Y (2011) Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Trans Neural Netw* 22(12):2226–2236
39. Zhang H, Wei Q, Liu D (2011) An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games. *Automatica* 47(1):207–214
40. Zhang H, Liu D, Luo Y, Wang D (2013) *Adaptive dynamic programming for control: algorithms and stability*. Springer, London

Part III

Applications

Chapter 12

Adaptive Dynamic Programming for Optimal Residential Energy Management

12.1 Introduction

With the rising cost, environmental concerns, and reliability issues, there is an increasing need to develop optimal control and management systems for residential environments. 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 operational efficiency of power systems [4, 23, 31]. On the other hand, with the rapidly evolving technology of electric storage devices, energy storage-based optimal management has attracted much attention [3, 19, 36]. Along with the development of smart grids, increasing intelligence is required in the optimal design of residential energy systems [10, 34, 39]. Hence, the intelligent optimization of battery management becomes a key tool for saving the power expense in smart residential environments.

Different techniques have been used to implement optimal controllers in residential energy management systems; for example, dynamic programming is used in [32, 37] and genetic algorithm is proposed in [11]. In addition, Liu and Huang [21] proposed an ADP scheme using only critic network and considering only three possible controls for the battery (charging mode, discharging mode, idle) to choose the best for every time slot, while in [18], a particle swarm optimization method and a mixed integer linear programming procedure are chosen in [33].

In this chapter, the optimal management of the total electrical system is viewed as optimal battery management for each time slot: Step by step, the controller provides the best decision of energy management whereby charging or discharging the battery and reducing the total cost according to the external environment. First, an action-dependent heuristic dynamic programming method is developed to obtain the optimal residential energy control law [21, 22]. Second, a dual iterative Q-learning algorithm is developed to solve the optimal battery management and control problem in smart residential environments where two iterations, internal and external iterations, are

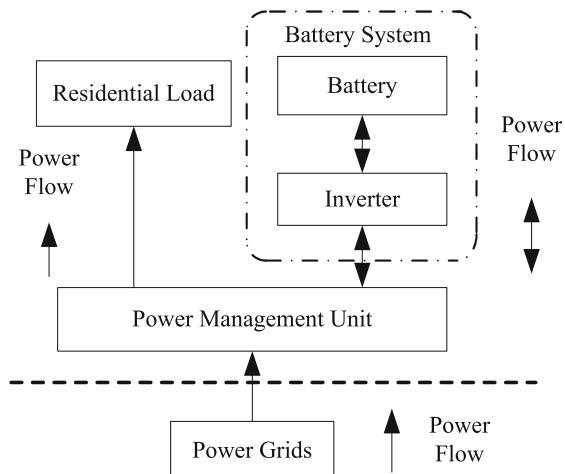
employed [42]. Based on the dual iterative Q-learning algorithm, the convergence property of iterative Q-learning method for the optimal battery management and control problem is proven. Finally, a distributed iterative ADP method is developed to solve the multi-battery optimal coordination control problems for home energy management systems [43].

12.2 A Self-learning Scheme for Residential Energy System Control and Management

The objective of this section is to apply an action-dependent heuristic dynamic programming (ADHDP) algorithm to obtain the optimal control for home energy management systems which minimizes the sum of system operational cost over the scheduling period, subject to technological and operational constraints of power grids and storage resources and subject to the system constraints such as power balance and reliability. For this purpose, we focus our research on finding the optimal battery charge/discharge strategy of the residential energy system including batteries and power grids [21, 22].

The residential energy system uses AC utility grid as the primary source of electricity and is intended to operate in parallel with the battery storage system. Figure 12.1 depicts the schematic diagram of a residential energy system. The system consists of power grids, a sine wave inverter, a battery system, and a power management unit. The battery storage system is connected to power management system through an inverter. The inverter functions as both charger and discharger for the battery. The construction of the inverter is based upon power MOSFET technology and pulse width modulation technique [16]. The quality of the inverter output is

Fig. 12.1 Grid-connected residential energy system with battery storage



comparable to that delivered from the power grids. The battery storage system consists of lead acid batteries, which are the most commonly used rechargeable battery type. The optimum battery size for a particular residential household can be obtained by performing various test scenarios, which is beyond the scope of the present book. Generally speaking, the battery is sized to enable it to supply power to the residential load for a period of twelve hours.

There are three operational modes for the batteries in residential energy system under consideration.

- (1) Charging mode: When system load is low and the electricity price is inexpensive, the power grids will supply the residential load directly and, at the same time, charge the batteries.
- (2) Idle mode: the power grids will directly supply the residential load at certain hours when, from the economical point of view, it is more cost effective to use the fully charged batteries in the evening peak hours.
- (3) Discharging mode: By taking the subsequent load demands and time-varying electricity rate into account, batteries alone supply the residential load at hours when the price of electricity from the grid is high.

This system can easily be expanded; i.e., other power sources along with the power grid and sources such as photovoltaic (PV) panels or wind generators can be integrated into the system when they are available.

For this section, the optimal scheduling problem is treated as a discrete-time problem with the time step as one hour, and it is assumed that the residential load over each hourly time step varies with noise. Thus, the daily load profile is divided into twenty-four hour period to represent each hour of the day. Each day can be divided into a greater number of periods to have higher resolution. However, for simplicity and agreement with existing literature [6, 9, 13, 30], we use a twenty-four-hour period each day in this work. A typical weekday load profile is shown in Fig. 12.2. The load factor P_L is expressed as P_{Lt} during hour t ($t = 1, 2, \dots, 24$). For instance, at time $t = 19$, the load is 7.8 kW which would require 7.8 kWh of energy. Since the load profile is divided into one-hour steps, the units of the power of energy sources can be represented equally by kW or kWh. Residential real-time pricing is one of the load management policies used to shift electricity usage from peak load hours to light load hours in order to improve the power system efficiency and allow the new power system construction projects [24]. With real-time pricing, the electricity rate varies from hour to hour based on the wholesale market prices. Hourly, market-based electricity prices typically change as the demand for electricity changes; higher demand usually means higher hourly prices. In general, there tends to be a small price spike in the morning and another slightly larger spike in the evening when the corresponding demand is high. Figure 12.3 demonstrates a typical daily real-time pricing from [12]. The varying electricity rate is expressed as C_t , representing the energy cost during the hour t in cents. For the residential customer with real-time pricing, energy charges are functions of the time of electricity usage. Therefore, for the situation where batteries are charged during the low rate hours

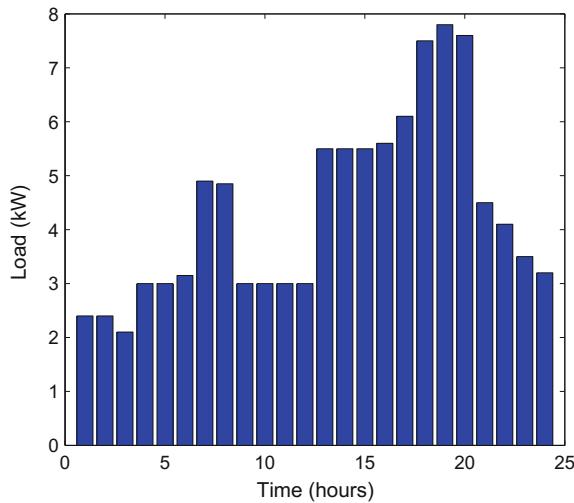


Fig. 12.2 A typical residential load profile

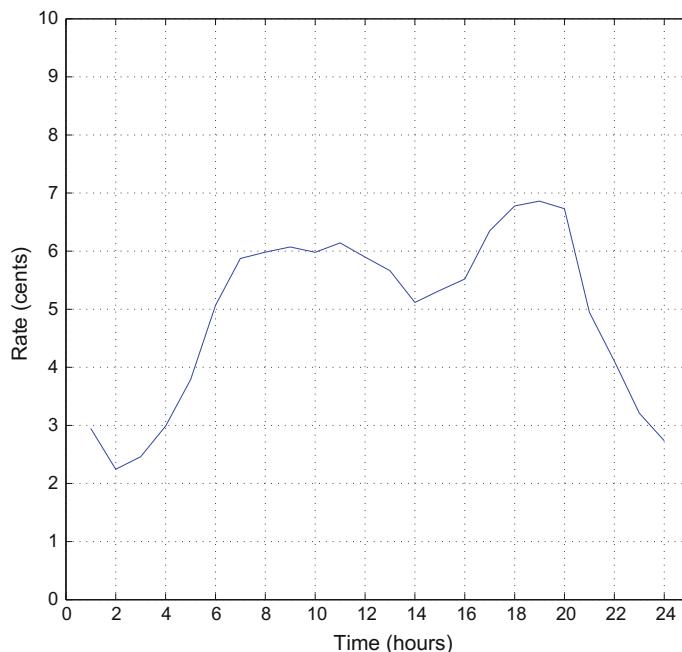


Fig. 12.3 A typical daily real-time pricing

and discharged during high rate hours, one may expect, from an economical point of view, the cost savings will be made by storing energy during low rate hours and releasing it during the high rate hours. In this way, the battery storage system can be used to reduce the total electricity cost for residential household. The energy stored in a battery can be expressed as [24, 44]:

$$E_{bt} = E_{b0} - \sum_{\tau=0}^t P_{b\tau},$$

$$P_{b\tau} = V_b I_b \alpha_\tau, \quad \tau = 0, 1, \dots, t,$$

$$\alpha_\tau = \begin{cases} 1, & \tau \leq \tau_0, \\ K_1(I_b) dV_b/d\tau, & \tau > \tau_0, \end{cases}$$

$$V_{b\tau} = V_s - (K_c(\Delta/(\Delta - J_c\tau) + N)J_c + A \exp(-B\Delta^{-1}J_c\tau)),$$

where E_{bt} is the battery energy at time t , E_{b0} is the peak energy level when the battery is fully charged (capacity of the battery), $P_{b\tau}$ is the battery power output at time τ , $V_{b\tau}$ is the terminal voltage of the battery, I_b is the battery discharge current, α_τ is the current weight factor as a function of discharge time, τ_0 is the battery manufacturer specified length of time for constant power output under constant discharge current rate, $K_1(I_b)$ is the weight factor as a function of the magnitude of the current, V_s is the battery internal voltage, K_c is the polarization coefficient (ohm \times cm 2), Δ is the available amount of active material (coulombs per cm 2), J_c is the apparent current density (amperes per cm 2), N is the internal resistance per cm 2 , and A and B are constants.

Apart from the battery itself, the loss of other equipments such as inverters, transformers, and transmission lines should also be considered in the battery model. The efficiency of these devices was derived in [44] as follows:

$$\eta(P_{bt}) = 0.898 - 0.173 \frac{|P_{bt}|}{P_{\text{rate}}}, \quad P_{\text{rate}} > 0, \quad (12.2.1)$$

where P_{rate} is the rated power output of the battery and $\eta(P_{bt})$ is the total efficiency of all the auxiliary equipments in the battery system.

Assume that all the loss caused by these equipments occur during the charging period. The battery model used in this work is expressed as follows: When the battery charges,

$$E_{b(t+1)} = E_{bt} - P_{b(t+1)} \times \eta(P_{b(t+1)}), \quad P_{b(t+1)} < 0,$$

and when the battery discharges,

$$E_{b(t+1)} = E_{bt} - P_{b(t+1)} \times \eta(P_{b(t+1)}), \quad P_{b(t+1)} > 0.$$

In general, to improve the battery efficiency and extend the battery's lifetime as far as possible, two constraints need to be considered:

- (1) Battery has storage limit. A battery lifetime may be reduced if it operates at lower amount of charge. In order to avoid damage, the energy stored in the battery must always meet constraint as follows:

$$E_b^{\min} \leq E_{bt} \leq E_b^{\max}. \quad (12.2.2)$$

- (2) For safety, battery cannot be charged or discharged at rate exceeding the maximum and minimum values to prevent damage. This constraint represents the upper and lower limit for the hourly charging and discharging power. A negative P_{bt} means that the battery is being charged, while a positive P_{bt} means the battery is discharging,

$$P_b^{\min} \leq P_{bt} \leq P_b^{\max}. \quad (12.2.3)$$

At any time, the sum of the power from the power grids and the batteries must be equal to the demand of residential user

$$P_{Lt} = P_{bt} + P_{gt}, \quad (12.2.4)$$

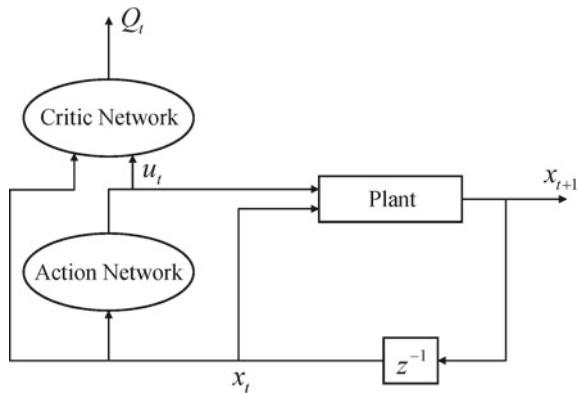
where P_{gt} is the power from the power grids, P_{bt} can be positive (in the case of batteries discharging) or negative (batteries charging) or zero (idle). It explains the fact that the power generation (power grids and batteries) must balance the load demand for each hour in the scheduling period. We assume here that the supply from power grids is enough for the residential demand. The objective of the optimization policy is, given the residential load profile and real-time pricing, to find the optimal battery charge/discharge/idle schedule at each time step which minimizes the total cost $C_T = \sum_{t=1}^T C_t \times P_{gt}$, while satisfying the load balance equation (12.2.4) and the operational constraints (12.2.1)–(12.2.3). C_T represents the operational cost to the residential customer in a period of T hours. To make the best possible use of batteries for the benefit of residential customers, with time of day pricing signals given by C_t , it is a complex multistage stochastic optimization problem. Adaptive dynamic programming (ADP) which provides approximate optimal solutions to dynamic programming is applicable to this problem. Using ADP, we will develop a self-learning optimization strategy for residential energy system control and management. During real-time operations under uncertain changes in the environment, the performance of the optimal strategy can be further refined and improved through continuous learning and adaptation.

12.2.1 The ADHDP Method

In this section, we consider the following discrete-time nonlinear systems:

$$x_{t+1} = F(x_t, u_t), \quad t = 0, 1, 2, \dots, \quad (12.2.5)$$

Fig. 12.4 A typical scheme of an ADHDP



where $x_t \in \mathbb{R}^n$ denotes the state vector of the system, $u_t \in \mathbb{R}^m$ represents the control action, and F is a transition from the current state x_t to the next state x_{t+1} under given state feedback control action $u_t = u(x_t)$ at time t . Suppose that this system is associated with the performance index

$$J(x_{t_0}, u) = J^u(x_{t_0}) = \sum_{k=t_0}^{\infty} \gamma^{k-t_0} U(x_k, u_k), \quad (12.2.6)$$

where U is called the utility function and γ is the discount factor with $0 < \gamma \leq 1$. It is important to realize that J depends on the initial time t_0 and the initial state x_{t_0} . The performance index J is also referred to as the cost-to-go of state x_{t_0} . The objective of dynamic programming problem is to choose a sequence of control actions $u_t = u(x_t)$, $t = t_0, t_0 + 1, \dots$, so that the performance index J in (12.2.6) is minimized. According to Bellman, the optimal cost from the time t on is equal to

$$J^*(x_t) = \min_{u_t} \{U(x_t, u_t) + \gamma J^*(F(x_t, u_t))\}.$$

The optimal control u_t^* at time t is the u_t that achieves this minimum.

Generally speaking, there are three design families of ADP: heuristic dynamic programming (HDP), dual heuristic programming (DHP), and globalized dual heuristic programming (GDHP) as well as their action-dependent versions. A typical ADHDP is shown in Fig. 12.4 [28]. Both the critic and action networks can be trained using the strategy in [25] as described in Sect. 1.3.1 of this book.

12.2.2 A Self-learning Scheme for Residential Energy System

The learning control architecture for residential energy system control and management is based on ADHDP. However, as described below, only a single module (single

critic approach) will be used instead of two or three modules in the original scheme. The single critic module technique retains all the powerful features of the original ADP, while eliminating the action module completely. There is no need for the iterative training loops between the action and the critic networks and, thus, greatly simplifies the training process.

There exists a class of problems in realistic applications that have a finite-dimensional control action space. Typical examples include inverted pendulum or the cart-pole problem, where the control action only takes a few finite values. When there is only a finite control action space in the application, the decisions that can be made are constrained to a limited number of choices, e.g., a ternary choice in the case of residential energy control and management problem. When there is a power demand from the residential household, the decisions can be made are constrained to three choices, i.e., to discharge batteries, to charge batteries, or to do nothing to batteries. Let us denote the three options using $u_t = 1$ for “discharge,” $u_t = -1$ for “charge,” and $u_t = 0$ for “idle.” In the present case, we note that the control actions are limited to a ternary choice or to only three possible options. Therefore, we can further simplify the ADHDP introduced in Fig. 12.4 so that only the critic network is needed in the design.

Figure 12.5 illustrates our self-learning control scheme for residential energy system control and management using ADHDP. The control scheme works as follows: When there is a power demand from the residential household, we will first ask the critic network to see which action (discharge, charge, and idle) generates the smallest output value of the critic network; then, the control action from $u_t = 1, -1, 0$ that generates the smallest critic network output will be chosen. As in the case of Fig. 12.4, the critic network in our design will also need the system states as input variables. It is important to realize that Fig. 12.5 is only a diagrammatic layout that illustrates how the computation takes place while making battery control and management decisions. In Fig. 12.5, the three blocks for the critic network stand for the same critic network or computer program. From the block diagram in Fig. 12.5, it is clear that the critic network will be utilized three times in calculations with different values of u_t to make a decision about whether to discharge or charge batteries or keep it idle. The previous description is based on the assumption that the critic network has been successfully trained. Once the critic network is learned and obtained (off-line or online), it will be applied to perform the task of residential energy system control and management as in Fig. 12.5. The performance of the overall system can be further refined and improved through continuous learning as it learns more experience in real-time operations when needed. In this way, the overall residential energy system will achieve optimal individual performance now and in the future environments under uncertain changes.

In stationary environment, where residential energy system configuration remains unchanged, a set of simple static if-then rules will be able to achieve the optimal scheduling as described previously. However, system configuration, including user power demand, capacity of the battery, and power rate, may be significantly different from time to time. To cope with uncertain changes in environments, static energy control and management algorithm would not be proper. The present control and

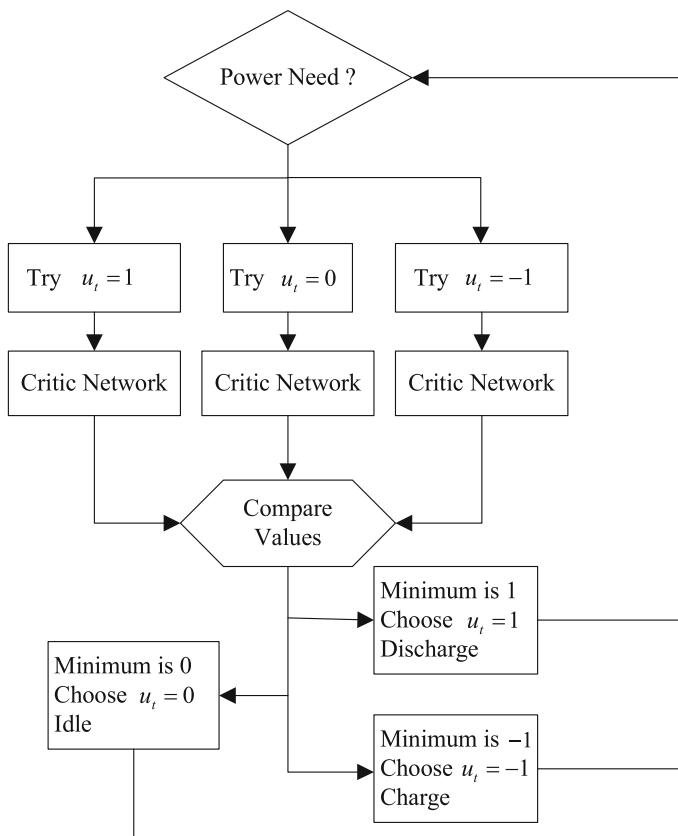


Fig. 12.5 Block diagram of the single critic approach

management scheme based on ADHDP will be capable of coping with uncertain changes in the environment through continuous learning. Another advantage of the present self-learning scheme is that through further learning as it gains more and more experience in real-time operations, the algorithm has the capability to adapt itself and improve performance. We note that continuous learning and adaptation over the entire operating regime and system conditions to improve the performance of the overall system is one of the key promising attributes of the present method.

The development of the present self-learning scheme for residential energy system control and management involves the following four steps.

Step 1: Collecting data: During this stage, whenever there is a power demand from residential household, we can take any of the following actions: discharge batteries, charge batteries, or keep batteries idle and calculate the utility function for the system. The utility function is chosen as follows:

$$U_t = \frac{\text{the electricity cost at time } t}{\text{the possible maximum cost}}.$$

During the data collection step, we simply choose actions 1, -1 , 0 randomly with the same probability of $1/3$. Meanwhile, the states corresponding to each action are collected. The environmental states we collect for each action are the electricity rate, the residential load, and the energy level of the battery.

- Step 2: Training the critic network: We use the data collected to train the critic network as presented in Chap. 1. The input variables chosen for the critic network are states including the electricity rate, the residential load, the energy level of the battery, and the action.
- Step 3: Applying the critic network: We apply the trained critic network as illustrated in Fig. 12.5. Three values of action u_t will be provided to the critic network at each time step. The action with the smallest output of the critic network is the one the system is going to take.
- Step 4: Further updating the critic network: We will update the critic network as needed while it is applied in the residential energy system to cope with environmental changes, for example, user demand changes or new requirements for the system. We note that the data has to be collected again and the training of critic network has to be performed as well. In such a case, the previous three steps will be repeated.

12.2.3 *Simulation Study*

The performance of the present algorithm is demonstrated by simulation studies for a typical residential family. The objective is to minimize the electricity cost from power grids over one-week horizon by finding the optimal battery operational strategy of the energy system while satisfying the load conditions and the system constraints. The focus of the present section is on residential energy system with home batteries connected to the power grids. For the residential energy system, the cost to be minimized is a function of real-time pricing and residential power demands. The optimal battery operation strategy refers to the strategy of when to charge batteries, when to discharge batteries, and when to keep batteries idle to achieve minimum electricity cost for the residential user.

The residential energy system consists of power grids, an inverter, batteries, and a power management unit as shown in Fig. 12.1. We assume that the supply from power grid is guaranteed for the residential user demand at any time. The capacity of batteries used in the simulations is 100 kWh, and a minimum of 20% of the charge is to be retained. The rated power output of batteries and the maximum charge/discharge rate is 16 kWh. The initial charge of batteries is at 80% of batteries' full charge. We assume that the batteries and the power grids will not simultaneously provide power to the residential user. At any time, residential power demand is supplied by either batteries or power grids. The power grids would provide the supply to the residential

user and, at the same time, charge batteries. It is expected that batteries are charged during the low rate hours, idle in some mid-rate hours, and discharged during high rate hours. In this way, both energy and cost savings are achieved.

The critic network in the present application is a multilayer feedforward neural network with 4–9–1 structure, i.e., four neurons at the input layer, nine neurons at the hidden layer, and one linear neuron at the output layer. The hidden layer uses the hyperbolic tangent function as the activation function. The critic network outputs function Q , which is an approximation to the function $J(x_t, u)$ as defined in (12.2.6). The four inputs to the critic network are as follows: energy level of batteries, residential power demand, real-time pricing, and the action of operation (1 for discharging batteries, -1 for charging batteries, 0 for keeping batteries idle). The local utility function defined in (12.2.6) is

$$U_t = \frac{C_t \times P_{gt}}{U_{\max}},$$

where C_t is real-time electricity rate, P_{gt} is the supply from power grids for residential power demand, and U_{\max} is the possible maximum cost for all time. The utility function chosen in this way will lead to a control objective of minimizing the overall cost for the residential user.

The typical residential load profile in one week is shown in Fig. 12.6 [12] random noise in the load curve. From the load curve, we can see that, during weekdays, there are two load peaks occurring in the period of 7:00–8:00 and 18:00–20:00, while during weekend, the residential demand gradually increases until the peak appears at 19:00. Thus, the residential demand pattern during weekdays and during weekend is different. Figure 12.7 shows the change in the electrical energy level in batteries

Fig. 12.6 A typical residential load profile in one week

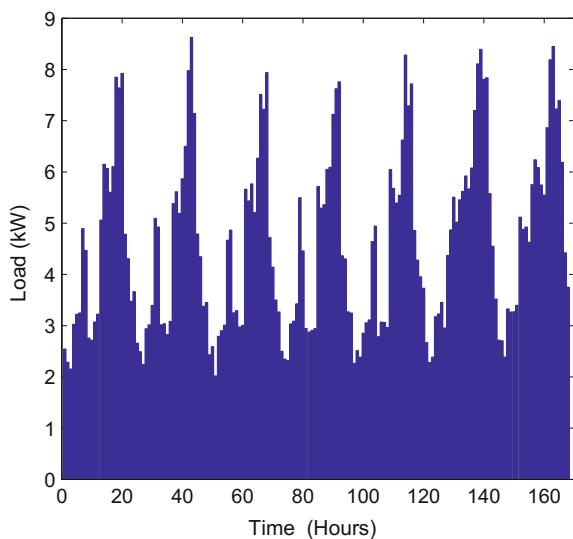


Fig. 12.7 Energy changes in batteries

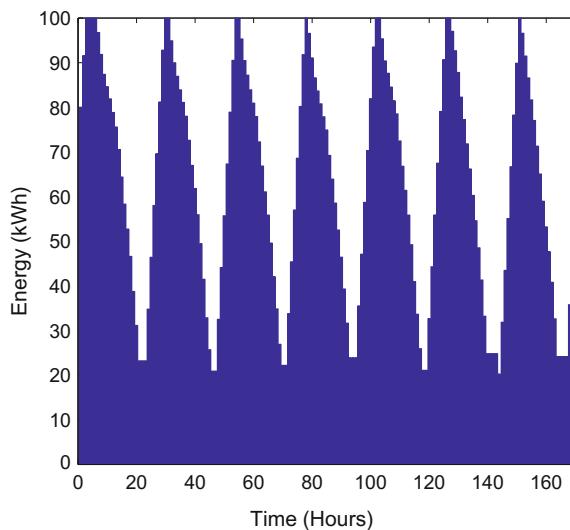
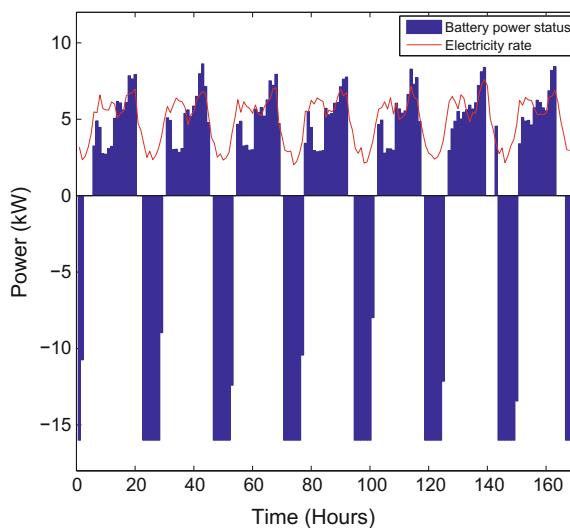


Fig. 12.8 Optimal scheduling of batteries in one week

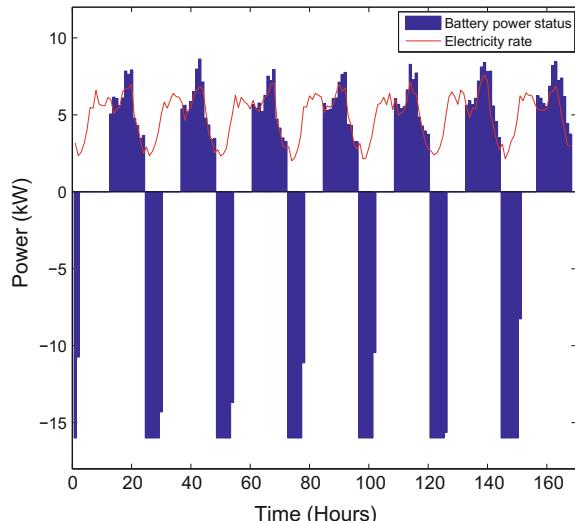


during a typical one-week residential load cycle. From Fig. 12.7, it is shown that batteries are fully charged during the midnight when the price of electricity is low. After that, batteries discharge during peak load hours or medium load hours and are charged again during the midnight light load hours. This cycle repeats, which means that the scheme is optimized with evenly charging and discharging. Therefore, the peak of the load curve is shaved by the output of batteries, which results in less cost of power from the power grids. Figure 12.8 illustrates the optimal scheduling of home batteries. The bars in Fig. 12.8 represent the power output of batteries,

while the dotted line denotes the electricity rate in real time. From Fig. 12.8, we can see that batteries are charged during hours from 23:00 to 5:00 next day when the electricity rate is in the lowest range and discharge when the price of electricity is expensive. It is observed that batteries discharge from 6:00 to 20:00 during weekdays and from 7:00 to 19:00 during weekend to supply the residential power demand. The difference lies in the fact that the power demand during the weekend is generally bigger than the weekdays' demand, which demonstrates that the present scheme can adapt to varying load conditions. From Fig. 12.8, we can also see that there are some hours that the batteries are idle, such as from 3:00 to 5:00 and from 21:00 to 22:00. Obviously, the self-learning algorithm believe that considering the subsequent load demand and electricity rate, keeping batteries idle during these hours will achieve the most economic return which result in the lowest overall cost to the customer. The cost of serving this typical residential load in one week is 2866.64 cents. Compared to the cost using the power grids alone to supply the residential load which is 4124.13 cents, it gives a savings of 1257.49 cents in a week period. This illustrates that a considerable saving on the electricity cost is achieved. In this case, the self-learning scheme has the ability to learn the system characteristics and provide the minimum cost to the residential user.

In order to better evaluate the performance of the self-learning scheme, we conduct comparison studies with a fixed daily cycle scheme. The daily cycle scheme charges batteries during the day time and releases the energy into the residential user load when required during the expensive peak hours at night. Figure 12.9 shows the scheduling of batteries by the fixed daily cycle scheme. The overall cost is 3284.37 cents. This demonstrates that the present ADHDP scheme has lower cost. Comparing Fig. 12.8 with 12.9, we can see the self-learning scheme is able to discharge batteries one hour late from 7:00 to 19:00 during the weekend instead of from 6:00 to

Fig. 12.9 Scheduling of batteries of fixed daily cycle scheme



20:00 during weekdays to achieve optimal performance, while the fixed daily cycle scheme ignores the differences in the demand between weekdays and weekend due to the static nature of the algorithm. Therefore, we conclude that the present self-learning algorithm performs better than the fixed algorithm due to the fact that the self-learning scheme can adapt to the varying load consideration and environmental changes.

12.3 A Novel Dual Iterative Q-Learning Method for Optimal Battery Management

12.3.1 Problem Formulation

The smart residential energy system described by (12.2.5) is composed of the power grid, the residential load, a battery system, which is located at the side of residential load (including a battery and a sine wave inverter), and a power management unit (controller). The schematic diagram of the smart residential energy system can be described in Fig. 12.1. The battery model used in this work is based on [22, 24, 44], where the battery model is expressed by

$$E_{b(t+1)} = E_{bt} - P_{bt} \times \eta(P_{bt}).$$

Let $P_{bt} > 0$ denote battery discharging; let $P_{bt} < 0$ denote battery charging; and let $P_{bt} = 0$ denote the battery idle. Let the efficiency of battery charging/discharging be derived as in (12.2.1).

In this section, 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. For convenience of analysis, we introduce delays in P_{bt} and P_{Lt} , and then, we can define the load balance as $P_{L(t-1)} = P_{b(t-1)} + P_{gt}$. The total cost function expected to be minimized is defined as

$$\sum_{t=0}^{\infty} \gamma^t [m_1(C_t P_{gt})^2 + m_2(E_{bt} - E_b^o)^2 + r(P_{bt})^2], \quad (12.3.1)$$

where $0 < \gamma < 1$ and $E_b^o = \frac{1}{2}(E_b^{\min} + E_b^{\max})$. The physical meaning of the first term of the cost function is to minimize the total cost from the grid. The second term aims to guarantee the stored energy of the battery to be 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 $x_{1t} = P_{gt}$ and $x_{2t} = E_{bt} - E_b^o$. Letting $x_t = [x_{1t}, x_{2t}]^T$ and $u_t = P_{bt}$, the equation of the residential energy system can then be written as

$$x_{t+1} = F(x_t, u_t, t) = \begin{pmatrix} P_{Lt} - u_t \\ x_{2t} - \eta(u_t)u_t \end{pmatrix}. \quad (12.3.2)$$

Let $\underline{u}_t = (u_t, u_{t+1}, \dots)$ 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 state. Then, the cost function (12.3.1) can be written as

$$J(x_0, \underline{u}_0, 0) = \sum_{t=0}^{\infty} \gamma^t U(x_t, u_t, t),$$

where $U(x_t, u_t, t) = x_t^T M_t x_t + r u_t^2$. Generally speaking, 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 12.3.1 The residential load P_{Lt} and the electricity rate C_t are periodic functions with the period $\lambda = 24$ h.

Define the control sequence set as $\underline{\mathcal{U}}_t = \{\underline{u}_t : \underline{u}_t = (u_t, u_{t+1}, \dots), u_{t+i} \in \mathbb{R}, i = 0, 1, \dots\}$. Then, the optimal cost function can be defined as follows:

$$J^*(x_t, t) = \inf_{\underline{u}_t} \{J(x_t, \underline{u}_t, t) : \underline{u}_t \in \underline{\mathcal{U}}_t\}.$$

Define the optimal Q-function as $Q^*(x_t, u_t, t)$ such that $\min_{u_t} Q^*(x_t, u_t, t) = J^*(x_t, t)$. Hence, the Q-function is also an action-dependent value function. According to [40, 41], the optimal Q-function satisfies the following Bellman equation

$$Q^*(x_t, u_t, t) = U(x_t, u_t, t) + \gamma \min_{u_{t+1}} Q^*(x_{t+1}, u_{t+1}, t+1). \quad (12.3.3)$$

12.3.2 Dual Iterative Q-Learning Algorithm

In this section, a novel dual iterative Q-learning algorithm is developed to obtain the optimal control law for residential energy systems [42]. A new convergence analysis method will also be developed in this section. From (12.3.3), we can see that the optimal Q-function $Q^*(x_t, u_t, t)$ is a nonlinear function which is difficult to obtain. According to Assumption 12.3.1, there exist $\rho = 0, 1, \dots$ and $\theta = 0, 1, \dots, 23$ such that $t = \rho\lambda + \theta, \forall t = 0, 1, \dots$. Let $k = \rho\lambda$. Then, $P_{Lt} = P_{L(k+\theta)} = P_{L\theta}$ and $C_t = C_{k+\theta} = C_\theta$, respectively. Define \mathcal{U}_k as the control sequence in 24 h from k to $k + \lambda - 1$, i.e., $\mathcal{U}_k = (u_k, u_{k+1}, \dots, u_{k+\lambda-1})$. We can define a new utility function as

$$\Pi(x_k, \mathcal{U}_k) = \sum_{\theta=0}^{\lambda-1} \gamma^\theta U(x_{k+\theta}, u_{k+\theta}, \theta), \quad \forall k \in \{0, \lambda, 2\lambda, \dots\}. \quad (12.3.4)$$

The utility function in (12.3.4) is time-invariant for $k = 0, \lambda, 2\lambda, \dots$, since the matrix M_t used in the definition of $U(x_t, u_t, t)$ is periodic with period of λ . Then, (12.3.3) can be expressed as

$$\mathcal{D}^*(x_k, \mathcal{U}_k) = \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \mathcal{D}^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda}),$$

where $\tilde{\gamma} = \gamma^\lambda$ and $\mathcal{D}(x_k, \mathcal{U}_k)$ is defined according to $\Pi(x_k, \mathcal{U}_k)$. The optimal control law sequence can be expressed as

$$\mathcal{U}^*(x_k) = \arg \min_{\mathcal{U}_k} \mathcal{D}^*(x_k, \mathcal{U}_k).$$

Based on the preparations above, a new dual iterative Q-learning algorithm can be developed. In the present algorithm, two iterations are utilized, which are external iterations (i -iterations in brief) and internal iterations (j -iterations in brief), respectively. Let $i = 0, 1, \dots$ be the external iteration index.

Let $\Psi(x_k, u_k)$ be an arbitrary positive-semidefinite function. Define the initial Q-function $\mathcal{Q}_0(x_k, \mathcal{U}_k)$ as

$$\mathcal{Q}_0(x_k, \mathcal{U}_k) = \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{u_{k+\lambda}} \Psi(x_{k+\lambda}, u_{k+\lambda}). \quad (12.3.5)$$

The control law \mathfrak{A}_0 can be computed as

$$\mathfrak{A}_0(x_k) = \arg \min_{\mathcal{U}_k} \mathcal{Q}_0(x_k, \mathcal{U}_k). \quad (12.3.6)$$

For $i = 1, 2, \dots$, the i -iteration will proceed between

$$\begin{aligned} \mathcal{Q}_i(x_k, \mathcal{U}_k) &= \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \mathcal{Q}_{i-1}(x_{k+\lambda}, \mathcal{U}_{k+\lambda}) \\ &= \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \mathcal{Q}_{i-1}(x_{k+\lambda}, \mathfrak{A}_{i-1}(x_{k+\lambda})), \end{aligned} \quad (12.3.7)$$

and

$$\mathfrak{A}_i(x_k) = \arg \min_{\mathcal{U}_k} \mathcal{Q}_i(x_k, \mathcal{U}_k), \quad (12.3.8)$$

where $\mathcal{Q}_0(\cdot, \cdot)$ and $\mathfrak{A}_0(\cdot)$ are determined by (12.3.5) and (12.3.6). Let $j = 0, 1, \dots, 24$ be the internal iteration index. For $i = 0$ and $j = 0$, let the initial Q-function be

$$Q_0^0(x_k, u_k) = \Psi(x_k, u_k) \quad (12.3.9)$$

and the corresponding control law is obtained by

$$u_0^0(x_k) = \arg \min_{u_k} Q_0^0(x_k, u_k). \quad (12.3.10)$$

For $i = 0$ and $j = 1, 2, \dots, 24$, the j -iteration will proceed between

$$\begin{aligned} Q_0^j(x_k, u_k) &= U(x_k, u_k, \lambda - j) + \gamma \min_{u_{k+1}} Q_0^{j-1}(x_{k+1}^{(j)}, u_{k+1}) \\ &= U(x_k, u_k, \lambda - j) + \gamma Q_0^{j-1}(x_{k+1}^{(j)}, u_0^{j-1}(x_{k+1}^{(j)})), \end{aligned} \quad (12.3.11)$$

and

$$u_0^j(x_k) = \arg \min_{u_k} Q_0^j(x_k, u_k), \quad (12.3.12)$$

where

$$x_{k+1}^{(j)} = F(x_k, u_k, \lambda - j) = \begin{pmatrix} P_{L(\lambda-j)} - u_k \\ x_{2k} - \eta(u_k)u_k \end{pmatrix} \quad (12.3.13)$$

and

$$U(x_k, u_k, \lambda - j) = x_k^\top M_{\lambda-j} x_k + r u_k^2.$$

Note that there are 24 such systems in (12.3.13) according to $j = 1, 2, \dots, 24$, and they are system (12.3.2) working at different hours according to $\lambda - j$.

For $i = 1, 2, \dots$, let $Q_i^0(x_k, u_k) = Q_{i-1}^{24}(x_k, u_k)$. For $j = 0$, we calculate

$$u_i^0(x_k) = \arg \min_{u_k} Q_i^0(x_k, u_k).$$

For $j = 1, 2, \dots, 24$, the j -iteration will proceed between

$$\begin{aligned} Q_i^j(x_k, u_k) &= U(x_k, u_k, \lambda - j) + \gamma \min_{u_{k+1}} Q_i^{j-1}(x_{k+1}^{(j)}, u_{k+1}) \\ &= U(x_k, u_k, \lambda - j) + \gamma Q_i^{j-1}(x_{k+1}^{(j)}, u_i^{j-1}(x_{k+1}^{(j)})), \end{aligned} \quad (12.3.14)$$

and

$$u_i^j(x_k) = \arg \min_{u_k} Q_i^j(x_k, u_k). \quad (12.3.15)$$

Then, we can obtain the iterative control law sequence as

$$\mathfrak{A}_i(x_k) = \{u_i^0(x_k), u_i^{23}(x_k), u_i^{22}(x_k), \dots, u_i^1(x_k)\}, \quad \forall i = 0, 1, \dots \quad (12.3.16)$$

Such an ordering of control laws can be understood with the proof of the next theorem (cf. (12.3.17) when $j = 24$).

In the present section, the convergence property of the dual iterative Q-learning algorithm will be established. First, we will show that iterative control law sequence $\mathfrak{A}_i(x_k)$ obtained by the j -iteration can minimize the total cost in each 24-h period.

Theorem 12.3.1 For $i = 0, 1, \dots$ and $j = 0, 1, \dots, 24$, let the iterative Q-functions $\mathcal{Q}_i(x_k, \mathcal{U}_k)$ and $Q_i^j(x_k, u_k)$ be obtained by (12.3.5)–(12.3.15). Then,

$$\min_{\mathcal{U}_k} \mathcal{Q}_i(x_k, \mathcal{U}_k) = \min_{u_k} Q_i^{24}(x_k, u_k).$$

Proof The statement can be proven by mathematical induction. First, for $i = 0$, we have

$$\begin{aligned} \min_{u_k} Q_0^j(x_k, u_k) &= \min_{u_k} \left\{ U(x_k, u_k, \lambda - j) + \gamma \min_{u_{k+1}} Q_0^{j-1}(x_{k+1}^{(j)}, u_{k+1}) \right\} \\ &= \min_{u_k} \left\{ U(x_k, u_k, \lambda - j) + \gamma \min_{u_{k+1}} \left\{ U(x_{k+1}^{(j)}, u_{k+1}, \lambda - j + 1) \right. \right. \\ &\quad \left. \left. + \gamma \min_{u_{k+2}} \left\{ U(x_{k+2}^{(j-1)}, u_{k+2}, \lambda - j + 2) + \dots \right. \right. \right. \\ &\quad \left. \left. \left. + \gamma \min_{u_{k+j-1}} \left\{ U(x_{k+j-1}^{(2)}, u_{k+j-1}, \lambda - 1) + \gamma \min_{u_{k+j}} \Psi(x_{k+j}^{(1)}, u_{k+j}) \right\} \right\} \right\} \right\} \\ &= \min_{(u_k, u_{k+1}, \dots, u_{k+j-1})} \left\{ U(x_k, u_k, \lambda - j) + \sum_{l=1}^{j-1} \gamma^l U(x_{k+l}^{(j-l+1)}, u_{k+l}, \lambda - j + l) \right. \\ &\quad \left. + \gamma^j \min_{u_{k+j}} \Psi(x_{k+j}^{(1)}, u_{k+j}) \right\}. \end{aligned} \quad (12.3.17)$$

Let $j = 24$. According to (12.3.4) and (12.3.5), we have

$$\min_{u_k} Q_0^{24}(x_k, u_k) = \min_{\mathcal{U}_k} \left\{ \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{u_{k+\lambda}} \Psi(x_{k+\lambda}, u_{k+\lambda}) \right\} = \min_{\mathcal{U}_k} \mathcal{Q}_0(x_k, \mathcal{U}_k).$$

Note that the superscripts used for x_{k+1}, x_{k+2}, \dots , can be dropped when $j = 24$. For example, when $j = 24$, the calculation of $x_{k+2}^{(j-1)} = x_{k+2}^{(23)}$ requires system (12.3.13) at hour = 1 (or equivalently, at $k + 1$), and the subscript 23 indicates exactly the same. The conclusion holds for $i = 0$. Assume that the conclusion holds for $i = \tau - 1$, i.e.,

$$\min_{\mathcal{U}_k} \mathcal{Q}_{\tau-1}(x_k, \mathcal{U}_k) = \min_{u_k} Q_{\tau-1}^{24}(x_k, u_k).$$

Then, for $i = \tau$, we have

$$\begin{aligned} \min_{u_k} Q_{\tau}^{24}(x_k, u_k) &= \min_{u_k} \left\{ U(x_k, u_k, 0) + \gamma \min_{u_{k+1}} \left\{ U(x_{k+1}^{(24)}, u_{k+1}, 1) \right. \right. \\ &\quad \left. \left. + \gamma \min_{u_{k+2}} Q_{\tau}^{22}(x_{k+2}^{(23)}, u_{k+2}) \right\} \right\} \\ &= \min_{u_k} \left\{ U(x_k, u_k, 0) + \gamma \min_{u_{k+1}} \left\{ U(x_{k+1}^{(24)}, u_{k+1}, 1) \right. \right. \\ &\quad \left. \left. + \gamma \min_{u_{k+2}} \left\{ U(x_{k+2}^{(23)}, u_{k+2}, 2) + \gamma \min_{u_{k+3}} \left\{ U(x_{k+3}^{(22)}, u_{k+3}, 3) + \dots \right. \right. \right. \right. \\ &\quad \left. \left. \left. \left. + \gamma \min_{u_{k+\tau}} \Psi(x_{k+\tau}^{(1)}, u_{k+\tau}) \right\} \right\} \right\} \end{aligned}$$

$$\begin{aligned}
& + \gamma \min_{u_{k+23}} \left\{ U(x_{k+23}^{(2)}, u_{k+23}, 23) + \gamma \min_{u_{k+\lambda}} Q_{\tau}^0(x_{k+\lambda}^{(1)}, u_{k+\lambda}) \right\} \} \} \} \\
& = \min_{\mathcal{U}_k} \left\{ \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{u_{k+\lambda}} Q_{\tau-1}^{24}(x_{k+\lambda}, u_{k+\lambda}) \right\} \\
& = \min_{\mathcal{U}_k} \left\{ \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \mathcal{Q}_{\tau-1}(x_{k+\lambda}, \mathcal{U}_{k+\lambda}) \right\} \\
& = \min_{\mathcal{U}_k} \mathcal{Q}_{\tau}(x_k, \mathcal{U}_k).
\end{aligned}$$

The mathematical induction is complete.

From Theorem 12.3.1, we can obtain the following corollary.

Corollary 12.3.1 *Let $\mu(x_k)$ be an arbitrary control law. For $i = 0, 1, \dots$ and $j = 0, 1, \dots, 24$, define a new value function as*

$$P_i^j(x_k, u_k) = U(x_k, u_k) + \gamma P_i^{j-1}(x_{k+1}, \mu(x_{k+1})),$$

and define $Q_i^j(x_k, u_k)$ as in (12.3.14). For $i = 0, 1, \dots$, let $P_i^0(x_k, u_k) = Q_i^0(x_k, u_k)$. Then, for $j = 0, 1, \dots, 24$, we have

$$Q_i^j(x_k, u_k) \leq P_i^j(x_k, u_k).$$

From Theorem 12.3.1 and Corollary 12.3.1, for $i = 0, 1, \dots$, the total cost in each period can be minimized by the iterative control law sequence $\mathfrak{A}_i(x_k)$ according to j -iteration (12.3.9)–(12.3.16). Next, the convergence property of i -iteration will be developed.

Theorem 12.3.2 *For $i = 0, 1, \dots$, let $\mathcal{Q}_i(x_k, \mathcal{U}_k)$ and $\mathfrak{A}_i(x_k)$ be obtained by i -iteration (12.3.5)–(12.3.8). Then, the iterative Q-function $\mathcal{Q}_i(x_k, \mathcal{U}_k)$ converges to its optimum, i.e.,*

$$\lim_{i \rightarrow \infty} \mathcal{Q}_i(x_k, \mathcal{U}_k) = \mathcal{Q}^*(x_k, \mathcal{U}_k). \quad (12.3.18)$$

Proof For functions $\mathcal{Q}^*(x_k, \mathcal{U}_k)$, $\Pi(x_k, \mathcal{U}_k)$, and $\mathcal{Q}_0(x_k, \mathcal{U}_k)$, inspired by [26], let $\underline{\varsigma}$, $\bar{\varsigma}$, $\underline{\delta}$, and $\bar{\delta}$ be constants such that

$$\underline{\varsigma} \Pi(x_k, \mathcal{U}_k) \leq \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \mathcal{Q}^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda}) \leq \bar{\varsigma} \Pi(x_k, \mathcal{U}_k),$$

and

$$\underline{\delta} \mathcal{Q}^*(x_k, \mathcal{U}_k) \leq \mathcal{Q}_0(x_k, \mathcal{U}_k) \leq \bar{\delta} \mathcal{Q}^*(x_k, \mathcal{U}_k),$$

respectively, where $0 < \underline{\varsigma} \leq \bar{\varsigma} < \infty$ and $0 \leq \underline{\delta} \leq \bar{\delta} < \infty$. Since $\mathcal{Q}^*(x_k, \mathcal{U}_k)$ is unknown, the values of $\underline{\varsigma}$, $\bar{\varsigma}$, $\underline{\delta}$, and $\bar{\delta}$ cannot be obtained directly. We will prove

that for the given constants $\underline{\zeta}$, $\bar{\zeta}$, $\underline{\delta}$, and $\bar{\delta}$, the iterative Q-function $\mathcal{Q}_i(x_k, \mathcal{U}_k)$ will converge to the optimum and the estimations of these constants can be omitted. The proof proceeds in four steps. First, we show that if $0 \leq \underline{\delta} \leq \bar{\delta} < 1$, and then for $i = 0, 1, \dots$, the iterative value function $\mathcal{Q}_i(x_k, \mathcal{U}_k)$ satisfies

$$\left(1 + \frac{\underline{\delta} - 1}{(1 + \bar{\zeta}^{-1})^i}\right) \mathcal{Q}^*(x_k, \mathcal{U}_k) \leq \mathcal{Q}_i(x_k, \mathcal{U}_k) \leq \left(1 + \frac{\bar{\delta} - 1}{(1 + \underline{\zeta}^{-1})^i}\right) \mathcal{Q}^*(x_k, \mathcal{U}_k). \quad (12.3.19)$$

The inequality (12.3.19) can be proven by mathematical induction. Let $i = 0$. We have

$$\begin{aligned} \mathcal{Q}_1(x_k, \mathcal{U}_k) &= \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \{\mathcal{Q}_0(x_{k+\lambda}, \mathcal{U}_{k+\lambda})\} \\ &\geq \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \{\mathcal{Q}^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda})\} \\ &\geq \left(1 + \bar{\zeta} \frac{\underline{\delta} - 1}{1 + \bar{\zeta}}\right) \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \left(\underline{\delta} - \frac{\underline{\delta} - 1}{1 + \bar{\zeta}}\right) \min_{\mathcal{U}_{k+\lambda}} \{\mathcal{Q}^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda})\} \\ &= \left(1 + \frac{\bar{\zeta}(\underline{\delta} - 1)}{(1 + \bar{\zeta})}\right) \left\{ \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \{\mathcal{Q}^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda})\} \right\} \\ &= \left(1 + \frac{\underline{\delta} - 1}{(1 + \bar{\zeta}^{-1})}\right) \mathcal{Q}^*(x_k, \mathcal{U}_k). \end{aligned} \quad (12.3.20)$$

Similarly, we can get

$$\mathcal{Q}_1(x_k, \mathcal{U}_k) \leq \left(1 + \frac{\bar{\delta} - 1}{(1 + \underline{\zeta}^{-1})}\right) \mathcal{Q}^*(x_k, \mathcal{U}_k).$$

Thus, (12.3.19) holds for $i = 0$. Assume that (12.3.19) holds for $i = l - 1$, $l = 1, 2, \dots$. Then, for $i = l$, we have

$$\begin{aligned} \mathcal{Q}_l(x_k, \mathcal{U}_k) &= \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \{\mathcal{Q}_{l-1}(x_{k+\lambda}, \mathcal{U}_{k+\lambda})\} \\ &\geq \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \left(1 + \frac{\bar{\zeta}^{l-1}(\underline{\delta} - 1)}{(1 + \bar{\zeta})^{l-1}}\right) \min_{\mathcal{U}_{k+\lambda}} \{\mathcal{Q}^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda})\} \\ &\geq \left(1 + \frac{\bar{\zeta}^l(\underline{\delta} - 1)}{(1 + \bar{\zeta})^l}\right) \left\{ \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \{\mathcal{Q}^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda})\} \right\} \\ &= \left(1 + \frac{\underline{\delta} - 1}{(1 + \bar{\zeta}^{-1})^l}\right) \mathcal{Q}^*(x_k, \mathcal{U}_k). \end{aligned} \quad (12.3.21)$$

Similarly, we can also get

$$\mathcal{Q}_l(x_k, \mathcal{U}_k) \leq \left(1 + \frac{\bar{\delta} - 1}{(1 + \underline{\zeta}^{-1})^l}\right) \mathcal{Q}^*(x_k, \mathcal{U}_k).$$

Hence, (12.3.19) holds for $i = 0, 1, \dots$. The mathematical induction is complete.

Second, we show that if $0 \leq \underline{\delta} \leq 1 \leq \bar{\delta} < \infty$, then the iterative Q-function $\mathcal{Q}_i(x_k, \mathcal{U}_k)$ satisfies

$$\left(1 + \frac{\underline{\delta} - 1}{(1 + \bar{\delta}^{-1})^i}\right) \mathcal{Q}^*(x_k, \mathcal{U}_k) \leq \mathcal{Q}_i(x_k, \mathcal{U}_k) \leq \left(1 + \frac{\bar{\delta} - 1}{(1 + \bar{\delta}^{-1})^i}\right) \mathcal{Q}^*(x_k, \mathcal{U}_k). \quad (12.3.22)$$

The lower bound of (12.3.22) can be proven according to the steps similar to (12.3.20) and (12.3.21). For the upper bound of (12.3.22), letting $i = 0$, we have

$$\begin{aligned} \mathcal{Q}_1(x_k, \mathcal{U}_k) &= \Pi(x_k, \mathcal{U}_k) + \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \{\mathcal{Q}_0(x_{k+\lambda}, \mathcal{U}_{k+\lambda})\} \\ &\leq \Pi(x_k, \mathcal{U}_k) + \bar{\delta} \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \{\mathcal{Q}^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda})\} \\ &\quad + \frac{\bar{\delta} - 1}{(1 + \bar{\delta})} \left(\bar{\delta} \Pi(x_k, \mathcal{U}_k) - \tilde{\gamma} \min_{\mathcal{U}_{k+\lambda}} \{\mathcal{Q}^*(x_{k+\lambda}, \mathcal{U}_{k+\lambda})\} \right) \\ &\leq \left(1 + \frac{\bar{\delta} - 1}{(1 + \bar{\delta}^{-1})}\right) \mathcal{Q}^*(x_k, \mathcal{U}_k). \end{aligned}$$

According to mathematical induction, we can obtain the upper bound of (12.3.22).

Third, for the situation $1 \leq \underline{\delta} \leq \bar{\delta} < \infty$, according to the steps similar to (12.3.20) and (12.3.21), we can prove that, for $i = 0, 1, \dots$, the iterative value function $\mathcal{Q}_i(x_k, \mathcal{U}_k)$ satisfies (12.3.19).

Finally, considering the three situations above, for the given constants $\underline{\delta}, \bar{\delta}, \underline{\delta}$, and $\bar{\delta}$, according to (12.3.19) and (12.3.22), we obtain (12.3.18), as $i \rightarrow \infty$. The proof is complete.

Corollary 12.3.2 For $i = 0, 1, \dots$, let $\mathcal{Q}_i(x_k, \mathcal{U}_k)$ and $\mathcal{U}_i(x_k)$ be obtained by i -iteration (12.3.5)–(12.3.8). Then, the iterative control law sequence $\mathcal{U}_i(x_k)$ converges to the optimal control law sequence, i.e.,

$$\lim_{i \rightarrow \infty} \mathcal{U}_i(x_k) = \mathcal{U}^*(x_k).$$

12.3.3 Neural Network Implementation

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 backpropagation (BP) networks. The whole structural diagram is shown in Fig. 12.10. The role of the action network is to approximate the

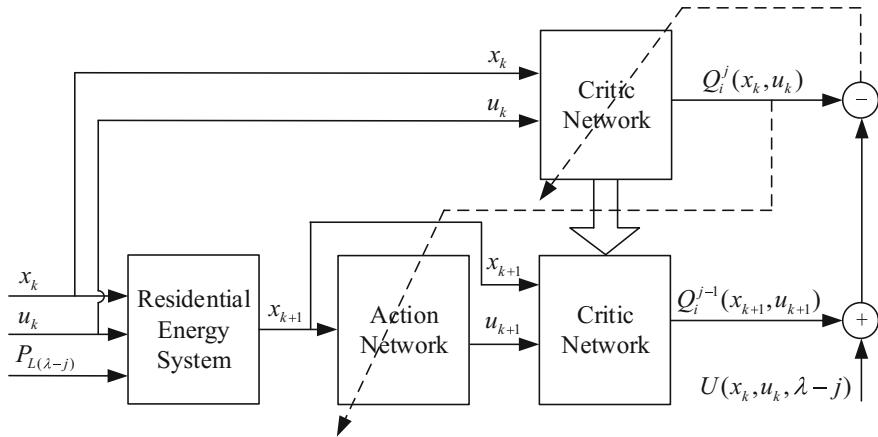


Fig. 12.10 The structural diagram of the dual iterative Q-learning algorithm (the discount factor γ is not shown in the diagram)

iterative control law sequence $\mathfrak{U}_i(x_k)$, $i = 0, 1, \dots$, defined in (12.3.8). The target of the action network can be defined as in (12.3.12) and (12.3.15). The action network can be constructed by 2 input neurons, 10 sigmoidal hidden neurons, and 1 linear output neuron. Let $l = 0, 1, \dots$ be the training step. The output of the action network can be expressed as

$$\hat{u}_i^{j,l}(x_k) = W_a^{ij\top}(l)\sigma(\mathcal{Z}_a(x_k)),$$

where $\mathcal{Z}_a(x_k) = Y_a^\top x_k$ and $\sigma(\cdot)$ is a sigmoid function [38]. To enhance the training speed, only the hidden–output weight $W_a^{ij}(l)$ is updated during the neural network training, while the input-hidden weight is fixed [15]. According to [38], the action network’s weight update is expressed as follows:

$$W_a^{ij}(l+1) = W_a^{ij}(l) - \beta_a \left[\frac{\partial E_{ai}^j(l)}{\partial W_a^{ij}(l)} \right],$$

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. The goal of the critic network is to obtain $Q_i(x_k, \mathcal{U}_k)$ by updating $Q_i^j(x_k, u_k)$ in (12.3.14) for $i = 0, 1, \dots$ and $j = 0, 1, \dots, 24$, iteratively. The critic network can be constructed by 3 input neurons, 15 sigmoidal hidden neurons, and 1 linear output neuron. Let $Z_{ck} = [x_k^\top, u_k]^\top$ be the input vector of the critic network. Then, the output of the critic network can be expressed as

$$\hat{Q}_i^{j,l}(x_k, u_k) = W_c^{ij\top}(l)\sigma(\mathcal{Z}_{ck}),$$

where $\mathcal{Z}_{ck} = Y_c^\top Z_{ck}$ and $\sigma(\cdot)$ is a sigmoid function [38]. During the neural network training, the hidden–output weight $W_{ci}^j(l)$ is updated, while the input-hidden weight Y_c is fixed. According to [38], the critic network weight update is expressed as follows:

$$W_c^{ij}(l+1) = W_c^{ij}(l) - \alpha_c \left[\frac{\partial E_{ci}^j(l)}{\partial W_c^{ij}(l)} \right],$$

where

$$E_{ci}^j(l) = \frac{1}{2}(e_{ci}^j(l))^2,$$

$$e_{ci}^j(l) = \hat{Q}_i^{j,l}(x_k, u_k) - Q_i^j(x_k, u_k),$$

and $\alpha_c > 0$ is the learning rate of the critic network. The dual iterative Q-learning algorithm implemented by action and critic networks is explained step by step and shown in Algorithm 12.3.1.

Algorithm 12.3.1 Dual iterative Q-learning algorithm

Initialization:

Collect an array of system data for the residential energy system (12.3.2).

Give a positive semidefinite function $\Psi(x_k, u_k)$.

Give the computation precision $\varepsilon > 0$.

Iteration:

Step 1. Let $i = 0$. For $j = 0$, let $Q_0^0(x_k, u_k) = \Psi(x_k, u_k)$.

Step 2. For $j = 0, 1, \dots, 24$, train the critic and action networks to obtain $Q_0^j(x_k, u_k)$ and $u_0^j(x_k)$ that satisfy (12.3.9)–(12.3.12).

Step 3. Let $\mathcal{Q}_0(x_k, \mathcal{U}_k) = Q_0^{24}(x_k, u_k)$. Obtain $\mathcal{A}_0(x_k)$ by (12.3.6).

Step 4. Let $i = i + 1$.

Step 5. For $j = 0, 1, \dots, 24$, train the critic and action networks to obtain $Q_i^j(x_k, u_k)$ and $u_i^j(x_k)$ that satisfy (12.3.14) and (12.3.15), respectively.

Step 6. Let $\mathcal{Q}_i(x_k, \mathcal{U}_k) = Q_i^{24}(x_k, u_k)$. Obtain $\mathcal{Q}_i(x_k, \mathcal{U}_k)$ and $\mathcal{A}_i(x_k)$ by (12.3.7) and (12.3.8), respectively.

Step 7. If $|\mathcal{Q}_i(x_k, \mathcal{U}_k) - \mathcal{Q}_{i-1}(x_k, \mathcal{U}_k)| \leq \varepsilon$, then goto next step; else, goto Step 4.

Step 8. Obtain $\mathcal{U}_i(x_k) = [u_i^{23}(x_k), \dots, u_i^0(x_k)]$.

Step 9. Return $\mathcal{Q}_i(x_k, \mathcal{U}_k)$ and $\mathcal{U}_i(x_k)$.

12.3.4 Numerical Analysis

In this section, the performance of the dual iterative Q-learning algorithm will be examined by numerical experiments. Comparisons will also be given to show the superiority of the present algorithm. The profiles of the residential load demand and the real-time electricity rate are taken from [8, 17, 22], where the real-time electricity rate and the residential load demand for one week (168 h) are shown in Fig. 12.11a and c, respectively. We can see that the real-time electricity rate and the residential load demand are both periodic-like functions with the period $\lambda = 24$. The average trajectories of the electricity rate and the residential load demand are shown in Fig. 12.11b, d. In this section, we use average residential load demand and average electricity rate as the periodic residential load demand and electricity rate.

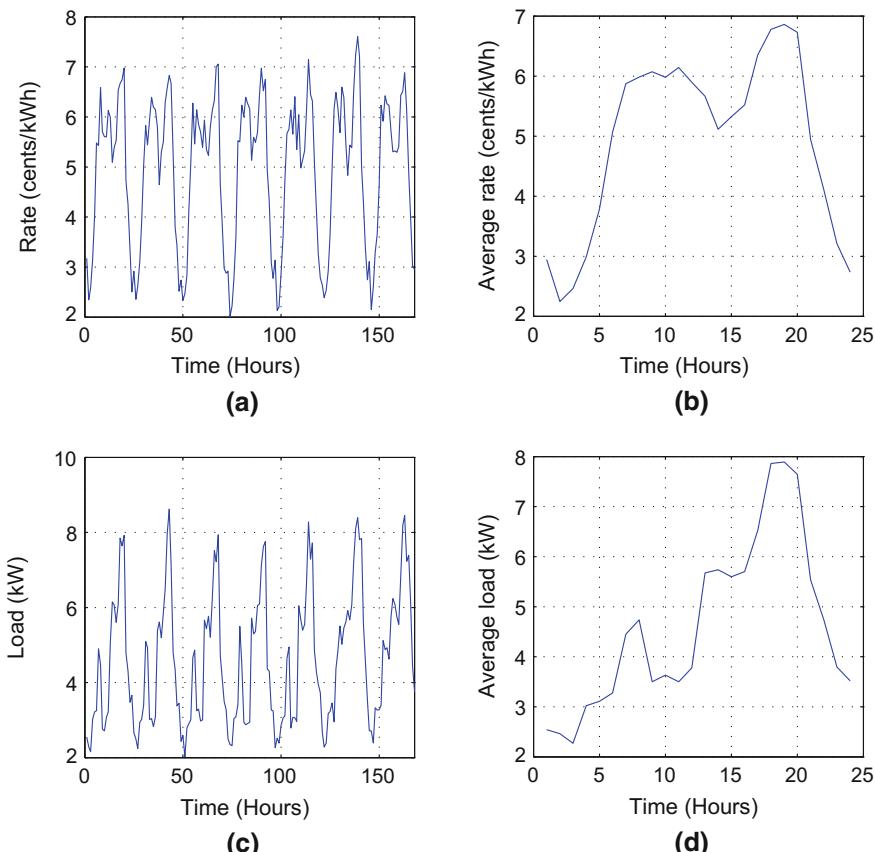


Fig. 12.11 Residential electricity rate and load demand. **a** Real-time electricity rate for 168 h. **b** Average electricity rate. **c** Residential load demand for 168 h. **d** Average residential load demand

We assume that the supply from the power grid guarantees the residential load demand at any time. Define the capacity of the 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 charging/discharging rate is 16 kW. The initial level of the battery is 60 kWh. Let the cost function be expressed as in (12.3.1), where we set $m_1 = 1$, $m_2 = 0.2$, $r = 0.1$, and $\gamma = 0.995$. Let the initial function $\Psi(x_k, u_k) = [x_k^T, u_k^T]P[x_k^T, u_k^T]^T$, where $P = I$ is the identity matrix with a suitable dimension. Let the initial state be $x_0 = [8, 60]^T$. After normalizing the data of the residential load demand and the electricity rate [1, 38], we implement the present 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. 12.12. After $i = 20$ iterations, we get $Q_i^j(x_0, \bar{u}) = Q_{i-1}^j(x_0, \bar{u}), j = 0, 1, \dots, 24$, which means the iterative Q-function is convergent to the optimum. According to the one week's residential load demand and electricity rate, the optimal control of the battery is shown in Fig. 12.13.

In the present study, time-based Q-learning (TBQL) algorithm [22] and particle swarm optimization (PSO) algorithm [17] will be compared to illustrate the superiority of the present dual iterative Q-learning algorithm. For $t = 0, 1, \dots$, the goal of TBQL algorithm [22, 38] is to design an iterative control that satisfies the following optimality equation

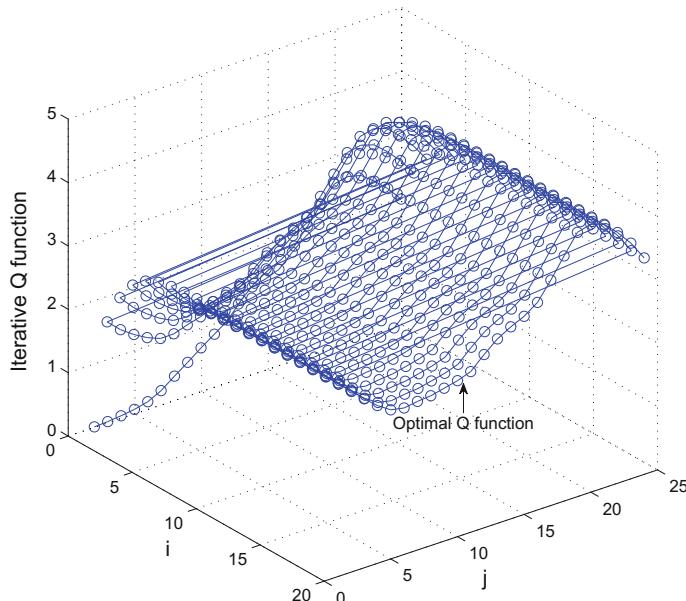


Fig. 12.12 The trajectory of the iterative Q-function

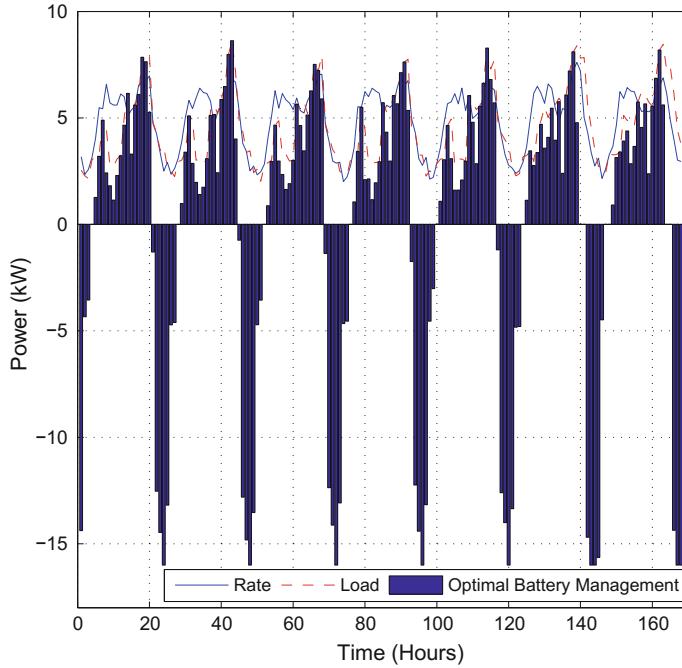


Fig. 12.13 Optimal control of the battery in one week

$$Q(x_{t-1}, u_{t-1}) = U(x_t, u_t) + \gamma Q(x_t, u_t).$$

Let the initial function and the structures of the action and critic networks which implement the TBQL algorithm be the same as those in our example. For PSO algorithm [17], let $\mathcal{G} = 30$ be the swarm size. The position of each particle at time t is represented by $x_{\ell t}$, $\ell = 1, 2, \dots, \mathcal{G}$ and its movement by the velocity vector $v_{\ell t}$. Then, the update rule of PSO can be expressed as

$$\begin{aligned} x_{\ell t} &= x_{\ell(t-1)} + v_{\ell t}, \\ v_{\ell t} &= \omega v_{\ell(t-1)} + \varphi_1 \rho_1^\top (p_\ell - x_{\ell(t-1)}) + \varphi_2 \rho_2^\top (p_g - x_{\ell(t-1)}). \end{aligned}$$

Let the inertia factor be $\omega = 0.7$. Let the correction factors $\rho_1 = \rho_2 = [1, 1]^\top$. Let φ_1 and φ_2 be random numbers in $[0, 1]$. Let p_ℓ be the best position of particles, and let p_g be the global best position. Implement the TBQL for 100 time steps, and implement PSO algorithm for 100 iterations. Let the real-time cost function be $R_{ct} = C_t P_{gt}$, and the corresponding real-time cost functions are shown in Fig. 12.14a, where the term “original” denotes “no battery system.” The comparison of the total cost for 168 h is displayed in Table 12.1. From Table 12.1, the superiority of our dual iterative Q-learning algorithm can be verified. The trajectories of the battery energy by dual

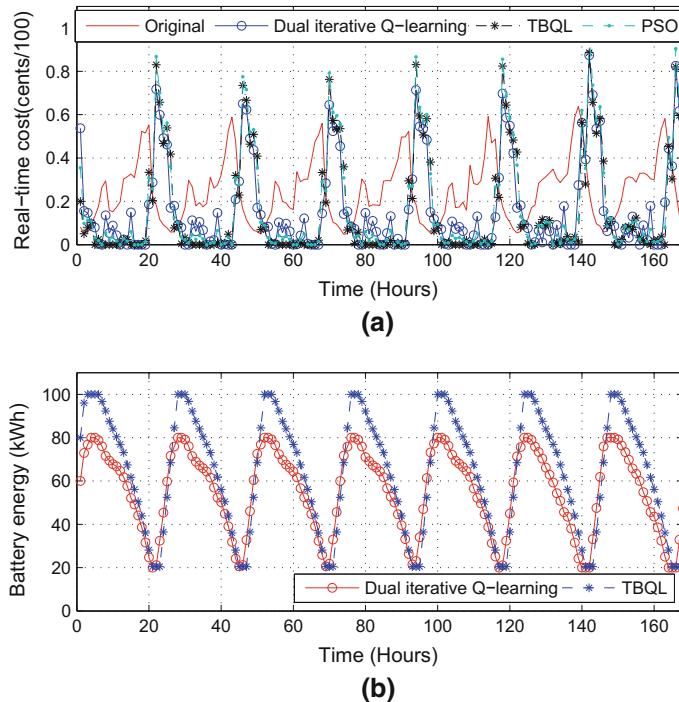


Fig. 12.14 Numerical comparisons. **a** Real-time cost comparison among dual iterative Q-learning, TBQL, and PSO algorithms. **b** Battery energy comparison between dual iterative Q-learning and TBQL algorithms

Table 12.1 Cost comparison

Horizon	Original	PSO	TBQL	Dual iterative Q-learning
Total cost (cents)	4124.13	3029.96	2866.64	2797.86
Saving (%)		26.53	30.49	32.16

iterative Q-learning and TBQL algorithms are shown in Fig. 12.14b. We can see that using the TBQL algorithm, the battery is fully charged each day, while the battery level is more reasonable by the dual iterative Q-learning algorithm.

In the above optimizations, we give more importance to the electricity rate than the cost of the battery system, i.e., m_1 in the cost function is large. On the other hand, the discharging rate and depth are also important for the battery system to be kept “alive” for as long as possible. Hence, we enlarge the parameters m_2 and r in the cost function. Let $m_2 = 1$, $r = 1$, and let m_1 be unchanged. The iterative Q-function is shown in Fig. 12.15. The optimal battery control is shown in Fig. 12.16, and the battery energy under the new cost function is shown in Fig. 12.17a. Enlarging m_2

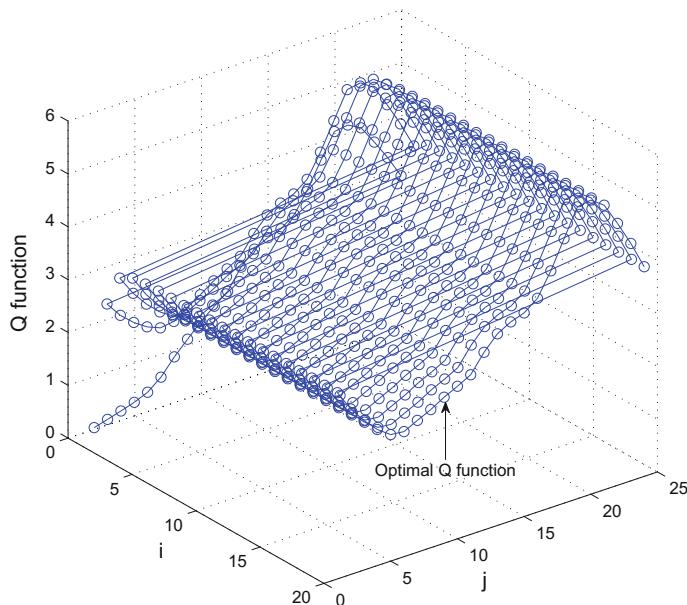


Fig. 12.15 The trajectory of the iterative Q-function

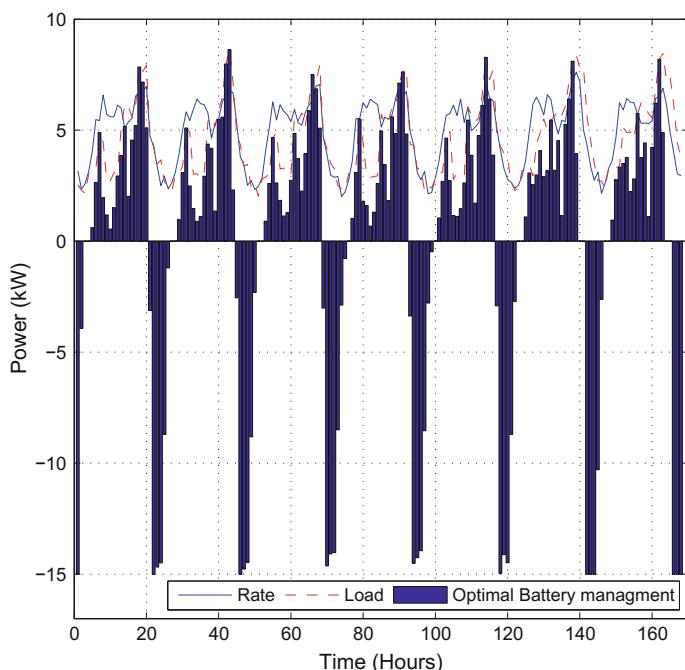


Fig. 12.16 Optimal control of the battery under $m_1 = 1$, $m_2 = 1$ and $r = 1$

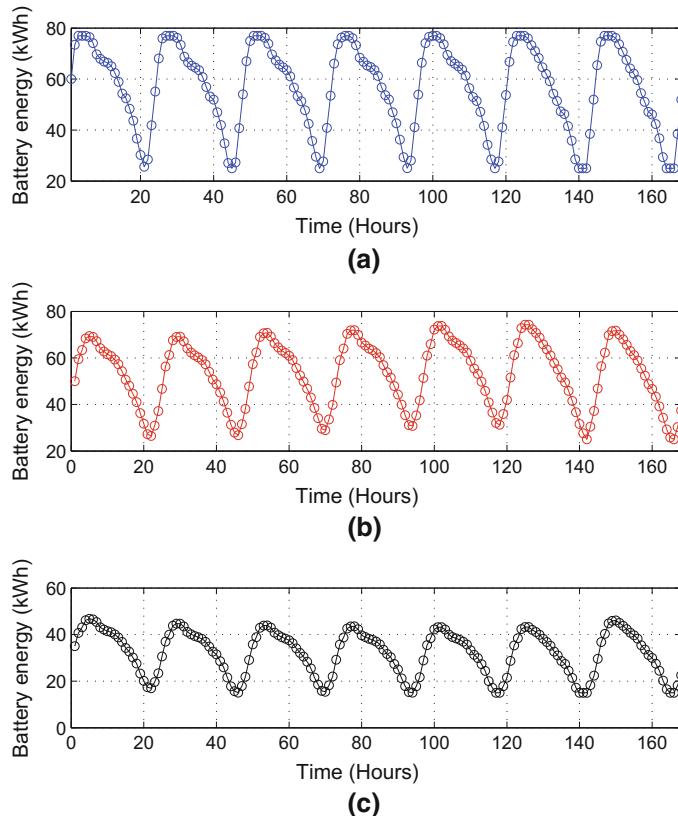


Fig. 12.17 Batteries' energy. **a** New cost function with $m_1 = 1$, $m_2 = 1$ and $r = 1$. **b** Battery I. **c** Battery II

and r , we can see that the value of the iterative Q-function is enhanced. The battery output power is reduced, and the battery energy is closer to E^o , which extends the lifetime of the battery. However, the total cost of one week is 2955.35 cents, which means the cost saving is reduced.

On the other hand, the battery model is important to the optimal control law of the battery. To illustrate the effectiveness of the present algorithm, different elements of the battery will be considered.

For convenience of analysis, we let $m_1 = 1$, $m_2 = 0.2$, $r = 0.1$. First, let the efficiency of battery charging/discharging be $\eta(P_{br}) = 0.698 - 0.173|P_{br}|/P_{rate}$ and let the capacity of the battery be 80 kWh. Define the battery as Battery I. Implementing the dual iterative Q-learning algorithm with Battery I, the trajectory of $Q_i^j(x_0, \bar{u})$ is shown in Fig. 12.18. We can see that the iterative Q-function is also convergent to the optimum after $i = 20$ iterations and the values of the Q-functions are larger than the ones in Fig. 12.12, which indicates that the optimization ability decreases. The

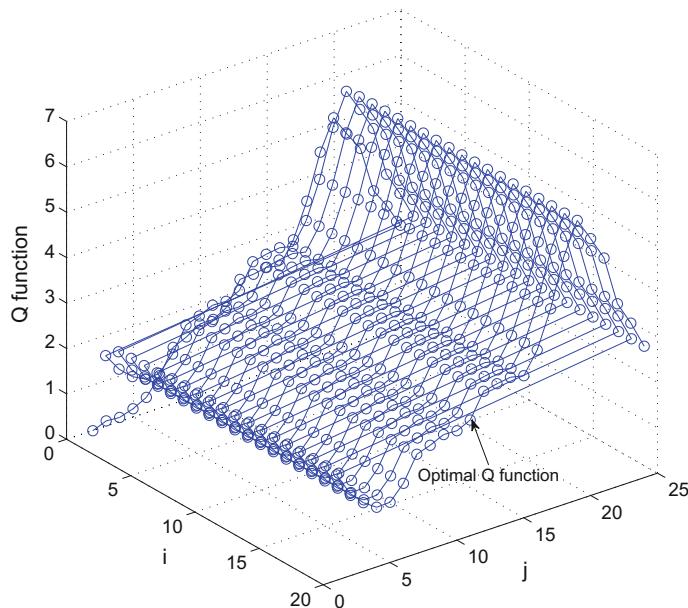


Fig. 12.18 The trajectory of the iterative Q-function

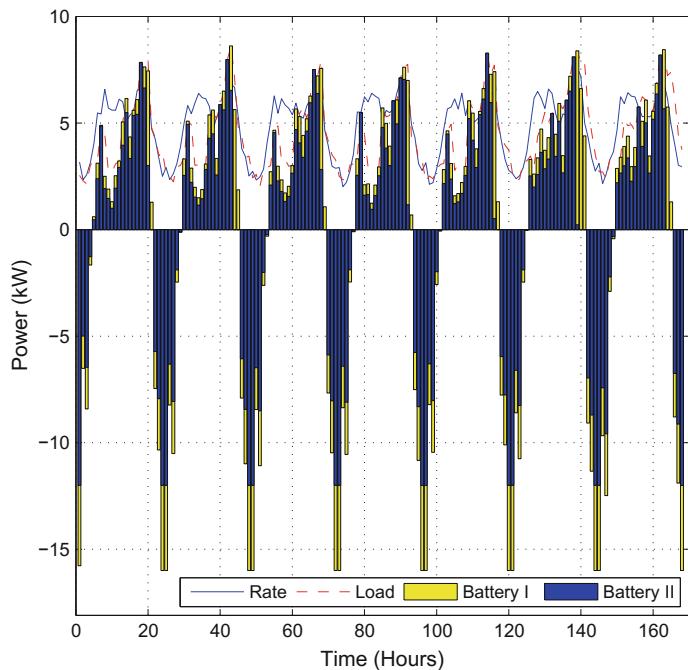


Fig. 12.19 Optimal control of the battery in one week

optimal control trajectory for Battery I is shown in Fig. 12.19. The battery energy of Battery I is shown in Fig. 12.17b, and the total cost in one week is 2914.70 cents.

Next, we keep on reducing the performance of the battery. Let the capacity of the battery decrease to 60 kWh. Let the rated power output of the battery and the maximum charging/discharging rate be 12 kW. Define the battery as Battery II. The optimal control trajectory for Battery II is shown in Fig. 12.19. The battery energy of Battery II is shown in Fig. 12.17c, and the total cost in one week is 3027.17 cents.

From the numerical results, we can see that for different battery models, the present dual iterative Q-learning algorithm will guarantee the iterative value function to converge to the optimum and obtain the optimal battery control law. We can also see that as the performance of the battery decreases, the optimization ability of the battery also decreases.

12.4 Multi-battery Optimal Coordination Control for Residential Energy Systems

In this section, the multi-battery home energy management system will be described and the optimization objective of the multi-battery coordination control will be introduced [43].

The optimal multi-battery control problem is treated as a discrete-time problem with the time step of 1 h, and it is assumed that the load demand varies hourly. The schematic diagram of the smart home energy system is described in Fig. 12.20, which is composed of the power grid, the load demand, the multi-battery system (including N batteries, $N \in \mathbb{Z}^+$, and sine wave inverters), and the power management unit

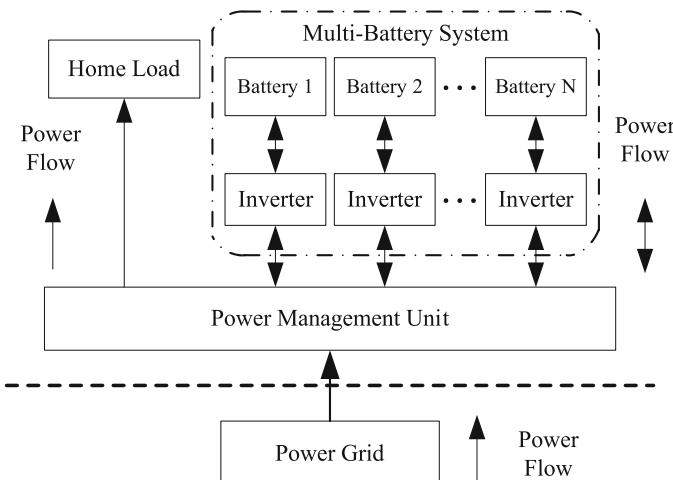


Fig. 12.20 Smart home energy system

(controller). Given the real-time home load and electricity rate, our goal is to find the optimal coordination control laws for the N batteries which minimize the total expense of the power from the grid.

The models of batteries are taken from [22, 24, 44]. The battery characteristics, battery hardware, battery software, and inverter/rectifier model of the battery were presented in [44], and the derivation of battery model was presented in [24]. Define N_ς as $N_\varsigma = \{\varsigma : \varsigma \in \mathbb{Z}^+, \varsigma \leq N\}$. Let $E_{b\varsigma,t}$ be the energy of battery ς at time t , and let $\eta_\varsigma(\cdot)$ be the charging/discharging efficiency of battery ς , $\forall \varsigma \in N_\varsigma$. Then, the model of battery ς can be expressed as $E_{b\varsigma,t+1} = E_{b\varsigma,t} - P_{b\varsigma,t} \times \eta(P_{b\varsigma,t})$, where $P_{b\varsigma,t}$ is the power output of battery ς at time t . Let $P_{b\varsigma,t} > 0$ denote battery ς discharging, $\forall \varsigma \in N_\varsigma$. Let $P_{b\varsigma,t} < 0$ denote battery ς charging, and let $P_{b\varsigma,t} = 0$ denote battery ς idle. Let the efficiency of battery charging/discharging be derived as $\eta(P_{b\varsigma,t}) = \bar{\eta}_\varsigma - 0.173|P_{b\varsigma,t}|/P_\varsigma^{\text{rate}}$, where $P_\varsigma^{\text{rate}} > 0$ is the rated power output of battery ς and $\bar{\eta}_\varsigma$ is the maximum efficiency constant such that $\bar{\eta}_\varsigma \leq 0.898$. The storage limit is defined as $E_{b\varsigma}^{\min} \leq E_{b\varsigma,t} \leq E_{b\varsigma}^{\max}$, where $E_{b\varsigma}^{\min}$ and $E_{b\varsigma}^{\max}$ are the minimum and maximum storage energies of battery ς , respectively. The corresponding charging and discharging power limits are defined as $P_{b\varsigma}^{\min} \leq P_{b\varsigma,t} \leq P_{b\varsigma}^{\max}$, where $P_{b\varsigma}^{\min}$ and $P_{b\varsigma}^{\max}$ are the minimum and maximum charging/discharging powers of battery ς , respectively.

Based on the models of batteries, the optimization objectives can be formulated. Let C_t be the electricity rate at time t . Let $P_{L,t}$ be the power of the home load at time t , and let $P_{g,t}$ be the power from the power grid. For the convenience of analysis, we assume that the home load $P_{L,t}$ and the electricity rate C_t are periodic functions with the period $\lambda = 24\text{h}$. For $\varsigma \in N_\varsigma$, with delays in $P_{L,t}$ and $P_{b\varsigma,t}$ for battery ς , the load balance can be expressed as

$$P_{L,t-1} = \sum_{\varsigma=1}^N P_{b\varsigma,t-1} + P_{g,t}.$$

In this section, power flow from the batteries to the grid is not permitted, i.e., we define $P_{g,t} \geq 0$, to guarantee the power quality of the grid. To extend the lifetime of the batteries, for $\varsigma \in N_\varsigma$, we desire that the stored energy of the battery ς is near the middle of storage limit $E_{b\varsigma}^o = \frac{1}{2}(E_{b\varsigma}^{\min} + E_{b\varsigma}^{\max})$. Define $\Delta E_{b\varsigma,t}$ as $\Delta E_{b\varsigma,t} = E_{b\varsigma,t} - E_{b\varsigma}^o$. Let $\alpha, \beta_\varsigma, r_\varsigma, \varsigma \in N_\varsigma$ be given positive constants, and let $0 < \gamma < 1$ be the discount factor. Then, the total cost function to be minimized can be defined as

$$\sum_{t=0}^{\infty} \gamma^t \left(\alpha(C_t P_{g,t})^2 + \sum_{\varsigma=1}^N (\beta_\varsigma \Delta E_{b\varsigma,t}^2 + r_\varsigma P_{b\varsigma,t}^2) \right), \quad (12.4.1)$$

The physical meaning of the first term of the cost function is to minimize the total cost from the grid. The second term aims to guarantee the stored energy of batteries to be close to the middle of storage limit, which avoids fully charging/discharging of the batteries. The third term is to prevent large charging/discharging power of

the batteries. Hence, the second and third terms aim to extend the lifetime of the batteries. Let $x_{1,t} = P_{g,t}$ and $x_{2\varsigma,t} = \Delta E_{b\varsigma,t}$, $\varsigma \in N_\varsigma$, be the system states. Let $u_{\varsigma,t} = P_{b\varsigma,t}$ be the control input. Let $M_2 = \text{diag}\{\beta_\varsigma\}$ and $M_t = \text{diag}\{\alpha C_t^2, M_2\}$. Let $x_t = [x_{1,t}, x_{21,t}, \dots, x_{2N,t}]^\top$ be the state vector and $u_t = [u_{1,t}, \dots, u_{N,t}]^\top$ be the control vector. The home energy management system is defined as

$$x_{t+1} = F(x_t, u_t, t) = \begin{bmatrix} P_{L,t} - \sum_{\varsigma=1}^N u_{\varsigma,t} \\ x_{21,t} - \eta(u_{1,t})u_{1,t} \\ \vdots \\ x_{2N,t} - \eta(u_{N,t})u_{N,t} \end{bmatrix}. \quad (12.4.2)$$

Let x_0 be the initial state. Then, the cost function (12.4.1) can be written as $J(x_0, \underline{u}_0, 0) = \sum_{t=0}^{\infty} \gamma^t U(x_t, u_t, t)$, where $U(x_t, u_t, t) = x_t^\top M_t x_t + u_t^\top R u_t$, $R = \text{diag}\{r_\varsigma\}$ and $\underline{u}_t = (u_t, u_{t+1}, \dots)$ denotes the control sequence from t to ∞ . The optimal cost function can be defined as $J^*(x_t, t) = \inf_{\underline{u}_t} \{J(x_t, \underline{u}_t, t)\}$. According to Bellman's principle of optimality [7], we can obtain the following Bellman equation

$$J^*(x_t, t) = \min_{u_t} \{U(x_t, u_t, t) + \gamma J^*(x_{t+1}, t+1)\}. \quad (12.4.3)$$

12.4.1 Distributed Iterative ADP Algorithm

In this section, a novel distributed iterative ADP algorithm is developed to solve the optimal multi-battery coordination control problem for home energy management systems. Convergence analysis results will be developed. From (12.4.2), the system state x_t is an $(N+1)$ -dimensional vector and the control u_t is an N -dimensional vector. Generally speaking, $J^*(x_t, t)$ in (12.4.3) is a highly nonlinear and nonanalytic function. For the multi-battery home energy management system (12.4.2), $J^*(x_t, t)$ is highly complex which is computationally untenable to obtain by directly solving the Bellman equation (12.4.3). To overcome these difficulties, system transformations are necessary to implement. First, for $t = 0, 1, \dots$, there exist $\rho = 0, 1, \dots$ and $\theta = 0, 1, \dots, 23$ such that $t = \rho\lambda + \theta$. Let $k = \rho\lambda$. Then, $P_{L,t} = P_{L,k+\theta} = P_{L,\theta}$ and $C_t = C_{k+\theta} = C_\theta$, respectively. Let \mathfrak{U}_k denote the control sequence for N batteries in 24h, i.e., $\mathfrak{U}_k = [u_k, u_{k+1}, \dots, u_{k+\lambda-1}]^\top \in \mathbb{R}^{\lambda \times N}$. Similar to (12.3.4), define a new utility function as

$$Y(x_k, \mathfrak{U}_k) = \sum_{\theta=0}^{\lambda-1} \gamma^\theta U(x_{k+\theta}, u_{k+\theta}, \theta).$$

Then, for all $k \in \{0, \lambda, 2\lambda, \dots\}$, the Bellman equation (12.4.3) can be expressed as

$$J^*(x_k) = \min_{\mathfrak{U}_k} \{ \Upsilon(x_k, \mathfrak{U}_k) + \tilde{\gamma} J^*(x_{k+\lambda}) \}, \quad (12.4.4)$$

and the optimal control law sequence can be expressed by

$$\mathfrak{U}^*(x_k) = \arg \min_{\mathfrak{U}_k} \{ \Upsilon(x_k, \mathfrak{U}_k) + \tilde{\gamma} J^*(x_{k+\lambda}) \},$$

where $\tilde{\gamma} = \gamma^{24}$. Note that $J^*(x_t, t)$ in (12.4.3) is time-varying. The function $J^*(x_k)$ in (12.4.4) is the same function, but it is time-invariant when the time step is λ , due to the periodic nature of the system parameters in (12.4.2). Therefore, the time dependency of $J^*(x_k)$ will be dropped when the context is clear.

Next, system transformations are developed to derive a system with lower dimensions. Let $\bar{E}_b = \min_{\varsigma \in N_\varsigma} \{E_{b\varsigma}^{\max} - E_{b\varsigma}^{\min}\}$ and $\underline{E}_b = 0$. Let $\bar{P}_{\text{rate}} = \min_{\varsigma \in N} \{P_{\varsigma}^{\text{rate}}\}$ and $\eta_\varsigma = \min_{\varsigma \in N_\varsigma} \bar{\eta}_\varsigma$. Let $\bar{P}_b^{\min} = \max_{\varsigma \in N} \{P_{b\varsigma}^{\min}\}$ and $\bar{P}_b^{\max} = \min_{\varsigma \in N} \{P_{b\varsigma}^{\max}\}$. Let

$$\eta(P_{b,t}) = \eta_\varsigma - 0.173 \frac{|P_{b,t}|}{\bar{P}_{\text{rate}}},$$

and $P_{b,t}$ be the charging/discharging power such that $\bar{P}_b^{\min} \leq P_{b,t} \leq \bar{P}_b^{\max}$. Here, we assume that $\bar{P}_b^{\min} < \bar{P}_b^{\max}$ and $\eta(P_{b,t}) > 0$ to facilitate our analysis. According to the above notations and assumptions, we can construct a new battery, called Battery \aleph , which is expressed as

$$\Delta E_{b,t+1} = \Delta E_{b,t} - P_{b,t} \times \eta(P_{b,t}), \quad (12.4.5)$$

where $\Delta E_{b,t} = E_{b,t} - E_b^o$, $E_b^o = \frac{1}{2}(\bar{E}_b - \underline{E}_b)$, and $E_{b,t}$ is the energy of the new battery at time t . From (12.4.5), we can see that Battery \aleph has the worst performance of all the N batteries (e.g., it is the “smallest” battery). If we replace all batteries ς by Battery \aleph , i.e., $x_{2\varsigma,t} = \Delta E_{b,t}$ and $u_{\varsigma,t} = P_{b,t}$, $\forall \varsigma \in N_\varsigma$, then the home energy management system (12.4.2) can be simplified as

$$z_{t+1} = F'(z_t, v_t, t) = \begin{pmatrix} P_{L,t} - Nv_t \\ x_{2,t} - \eta(v_t)v_t \end{pmatrix}, \quad (12.4.6)$$

where $z_t = [x_{1,t}, x_{2,t}]^\top$, $x_{2,t} = \Delta E_{b,t}$, and $v_t = P_{b,t}$. The cost function can be rewritten as

$$\sum_{t=0}^{\infty} \gamma^t \left(z_t^\top \bar{M}_t z_t + v_t^2 \sum_{\varsigma=1}^N r_\varsigma \right), \quad (12.4.7)$$

where $\bar{M}_t = \text{diag} \left\{ \alpha C_t^2, \sum_{\varsigma=1}^N \beta_{\varsigma} \right\}$. Next, we aim to design an optimal control law for Battery \aleph .

Define $\mathcal{U}_k = [v_k, v_{k+1}, \dots, v_{k+\lambda-1}]^T$, which is the control sequence for the Battery \aleph in 24 h. According to (12.4.6), there exists a function \tilde{F} such that $z_{k+\lambda} = \tilde{F}(z_k, \mathcal{U}_k)$. Let $\bar{U}(z_t, v_t, t) = z_t^T \bar{M}_t z_t + \bar{r} v_t^2$, where $\bar{r} = \sum_{\varsigma=1}^N r_{\varsigma}$. We can define another new utility function as $\Gamma(z_k, \mathcal{U}_k) = \sum_{\theta=0}^{\lambda-1} \gamma^{\theta} \bar{U}(z_{k+\theta}, v_{k+\theta}, \theta)$, $\forall k \in \{0, \lambda, 2\lambda, \dots\}$. Then, the Bellman equation (12.4.4) can be expressed as

$$\mathcal{J}^o(z_k) = \min_{\mathcal{U}_k} \{\Gamma(z_k, \mathcal{U}_k) + \tilde{\gamma} \mathcal{J}^o(z_{k+\lambda})\}, \quad (12.4.8)$$

and the optimal control law sequence can be expressed by

$$\mathcal{U}^o(z_k) = \arg \min_{\mathcal{U}_k} \{\Gamma(z_k, \mathcal{U}_k) + \tilde{\gamma} \mathcal{J}^o(z_{k+\lambda})\}.$$

In this section, an iterative ADP algorithm will be developed to obtain the optimal control laws for Battery \aleph . Let $i = 0, 1, \dots$ be the iteration index. Let $\Psi(z_k)$ be an arbitrary positive-semidefinite function and choose the initial value function $V^0(z_k) = \Psi(z_k)$. We obtain

$$\mathfrak{A}^0(z_k) = \arg \min_{\mathcal{U}_k} \{\Gamma(z_k, \mathcal{U}_k) + \tilde{\gamma} V^0(z_{k+\lambda})\}. \quad (12.4.9)$$

For $i = 1, 2, \dots$, the iterative value function and the iterative control law sequence will be obtained by

$$\begin{aligned} V^i(z_k) &= \min_{\mathcal{U}_k} \{\Gamma(z_k, \mathcal{U}_k) + \tilde{\gamma} V^{i-1}(z_{k+\lambda})\} \\ &= \Gamma(z_k, \mathfrak{A}^{i-1}(z_k)) + \tilde{\gamma} V^{i-1}(\tilde{F}(z_k, \mathfrak{A}^{i-1}(z_k))), \end{aligned} \quad (12.4.10)$$

and

$$\mathfrak{A}^i(z_k) = \arg \min_{\mathcal{U}_k} \{\Gamma(z_k, \mathcal{U}_k) + \tilde{\gamma} V^i(z_{k+\lambda})\}. \quad (12.4.11)$$

For a given $V^i(z_k)$, as λ is finite, according to the principle of dynamic programming [7], the iterative value function $V^i(z_k)$ can be updated in (12.4.10) and the iterative control law sequence

$$\mathfrak{A}^i(z_k) = [v^i(z_k), \dots, v^i(z_{k+\lambda-1})]^T$$

can be obtained in (12.4.11).

Theorem 12.4.1 If for $i = 0, 1, \dots$, $V^i(z_k)$ and $\mathcal{A}^i(z_k)$ are obtained by (12.4.9)–(12.4.11), then $V^i(z_k)$ will converge to the solution of Bellman equation (12.4.8), i.e.,

$$\lim_{i \rightarrow \infty} V^i(z_k) = \mathcal{J}^o(z_k).$$

Proof For functions $\mathcal{J}^o(z_k)$, $\Gamma(z_k, \mathcal{U}_k)$, and $V^0(z_k)$, inspired by [26], there exist constants ζ , $\underline{\delta}$, and $\bar{\delta}$ such that $\tilde{\gamma} \mathcal{J}^o(z_{k+\lambda}) \leq \zeta \Gamma(z_k, \mathcal{U}_k)$, and $\underline{\delta} \mathcal{J}^o(z_k) \leq V^0(z_k) \leq \bar{\delta} \mathcal{J}^o(z_k)$, respectively, where $0 < \zeta < \infty$ and $0 \leq \underline{\delta} \leq 1 \leq \bar{\delta} < \infty$. We will show that for $i = 0, 1, \dots$, the iterative value function $V^i(z_k)$ satisfies

$$\left(1 + \frac{\underline{\delta} - 1}{(1 + \zeta^{-1})^i}\right) \mathcal{J}^o(z_k) \leq V^i(z_k) \leq \left(1 + \frac{\bar{\delta} - 1}{(1 + \zeta^{-1})^i}\right) \mathcal{J}^o(z_k). \quad (12.4.12)$$

Inequality (12.4.12) can be proven by mathematical induction. Let $i = 1$. We have

$$\begin{aligned} V^1(z_k) &= \min_{\mathcal{U}_k} \left\{ \Gamma(z_k, \mathcal{U}_k) + \tilde{\gamma} V^0(z_{k+\lambda}) \right\} \\ &\geq \min_{\mathcal{U}_k} \left\{ \left(1 + \zeta \frac{\underline{\delta} - 1}{1 + \zeta}\right) \Gamma(z_k, \mathcal{U}_k) + \tilde{\gamma} \left(\underline{\delta} - \frac{\underline{\delta} - 1}{1 + \zeta}\right) \mathcal{J}^o(z_{k+\lambda}) \right\} \\ &= \left(1 + \frac{\underline{\delta} - 1}{(1 + \zeta^{-1})}\right) \mathcal{J}^o(z_k). \end{aligned} \quad (12.4.13)$$

Thus, (12.4.12) holds for $i = 0$. Assume that (12.4.12) holds for $i = l - 1$, $l = 1, 2, \dots$. Then, for $i = l$, we have

$$\begin{aligned} V^l(z_k) &= \min_{\mathcal{U}_k} \left\{ \Gamma(x_k, \mathcal{U}_k) + \tilde{\gamma} V^{l-1}(z_{k+\lambda}) \right\} \\ &\geq \min_{\mathcal{U}_k} \left\{ \Gamma(x_k, \mathcal{U}_k) + \tilde{\gamma} \left(1 + \frac{\underline{\delta} - 1}{(1 + \zeta^{-1})^{l-1}}\right) \mathcal{J}^o(z_{k+\lambda}) \right. \\ &\quad \left. + \frac{\underline{\delta} - 1}{(1 + \zeta)(1 + \zeta^{-1})^{l-1}} (\zeta \Gamma(x_k, \mathcal{U}_k) - \tilde{\gamma} \mathcal{J}^o(z_{k+\lambda})) \right\} \\ &= \left(1 + \frac{\underline{\delta} - 1}{(1 + \zeta^{-1})^l}\right) \min_{\mathcal{U}_k} \left\{ \Gamma(x_k, \mathcal{U}_k) + \tilde{\gamma} \mathcal{J}^o(z_{k+\lambda}) \right\} \\ &= \left(1 + \frac{\underline{\delta} - 1}{(1 + \zeta^{-1})^l}\right) \mathcal{J}^o(z_k). \end{aligned} \quad (12.4.14)$$

Based on the mathematical induction (12.4.13) and (12.4.14), we can obtain the upper bound of (12.4.12). On the other hand, if $\underline{\delta} \geq 1$ and $\bar{\delta} \leq 1$, we can let $\underline{\delta} = 1$ and $\bar{\delta} = 1$, where we can see that (12.4.12) can also be verified. Hence, according to (12.4.12), we can obtain

$$\lim_{i \rightarrow \infty} V^i(z_k) = \mathcal{J}^o(z_k).$$

The proof is complete.

Considering the worst-performance optimization for Battery \mathfrak{N} as the initial condition, a new distributed iterative ADP algorithm will be developed. Convergence properties will be established to guarantee that the iterative value function converges to the optimum. Let $\mu_{\varsigma,k} = [u_{\varsigma,k}, \dots, u_{\varsigma,k+\lambda-1}]^T$. Define $\bar{\mathfrak{U}}_\varsigma \in \mathbb{R}^{\lambda \times (N-1)}$ as $\bar{\mathfrak{U}}_\varsigma = [(\mu_{\tilde{\varsigma},k} : \tilde{\varsigma} \in N_\varsigma, \tilde{\varsigma} \neq \varsigma)] = [(\mu_{1,k}, \mu_{2,k}, \dots, \mu_{N,k})$ with $\mu_{\varsigma,k}$ removed], $\forall \varsigma \in N_\varsigma$. Let $\ell = 0, 1, \dots$ increase from 0 to ∞ . Let $\{\vartheta_\ell\}$ be a sequence such that $\vartheta_\ell \in N_\varsigma$, $\forall \ell = 0, 1, \dots$. According to (12.4.2), define a function \bar{F} that satisfies $x_{k+\lambda} = \bar{F}(x_k, \mathfrak{U}_{\vartheta_\ell})$. According to (12.4.7) and (12.4.8), for $k = 0, \lambda, 2\lambda, \dots$, we construct a cost function $J^o(x_k)$ such that $J^o(x_k) = \mathcal{J}^o(z_k)$. Let $\mu_\varsigma^o(x_k) = \mathcal{U}^o(z_k)$, $\forall \varsigma \in N_\varsigma$, and let $\mathfrak{U}_{\vartheta_0}(x_k) = [\mu_1^o(x_k), \dots, \mu_N^o(x_k)]$.

Now for $\ell = 0$ and $\tau = 0$, let the initial value function be given by

$$\mathcal{V}_{\vartheta_0}^0(x_k) = J^o(x_k). \quad (12.4.15)$$

We calculate

$$\mu_{\vartheta_0}^0(x_k) = \arg \min_{\mu_{\vartheta_0,k}} \{ \mathcal{V}(x_k, [\mu_{\vartheta_0,k}, \bar{\mathfrak{U}}_{\vartheta_0}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_0}^0(\bar{F}(x_k, [\mu_{\vartheta_0,k}, \bar{\mathfrak{U}}_{\vartheta_0}(x_k)])) \}. \quad (12.4.16)$$

Note that $\bar{\mathfrak{U}}_{\vartheta_0}(x_k)$ is $\mathfrak{U}_{\vartheta_0}(x_k)$ by removing its ϑ_0 th column. For $\ell = 0$, $\vartheta_0 \in N_\varsigma$ and $\tau = 1, 2, \dots$, the distributed iterative ADP algorithm will proceed between

$$\mathcal{V}_{\vartheta_0}^\tau(x_k) = \mathcal{V}(x_k, [\mu_{\vartheta_0}^{\tau-1}(x_k), \bar{\mathfrak{U}}_{\vartheta_0}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_0}^{\tau-1}(\bar{F}(x_k, [\mu_{\vartheta_0}^{\tau-1}(x_k), \bar{\mathfrak{U}}_{\vartheta_0}(x_k)])), \quad (12.4.17)$$

and

$$\mu_{\vartheta_0}^\tau(x_k) = \arg \min_{\mu_{\vartheta_0,k}} \{ \mathcal{V}(x_k, [\mu_{\vartheta_0,k}, \bar{\mathfrak{U}}_{\vartheta_0}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_0}^\tau(\bar{F}(x_k, [\mu_{\vartheta_0,k}, \bar{\mathfrak{U}}_{\vartheta_0}(x_k)])) \}. \quad (12.4.18)$$

For $\tau \rightarrow \infty$, let $\mathcal{V}_{\vartheta_0}^\infty(x_k) = \lim_{\tau \rightarrow \infty} \mathcal{V}_{\vartheta_0}^\tau(x_k)$ and $\mu_{\vartheta_0}^\infty(x_k) = \lim_{\tau \rightarrow \infty} \mu_{\vartheta_0}^\tau(x_k)$. Let $\mu_\varsigma^1(x_k) = \mu_\varsigma^0(x_k)$ for $\varsigma \neq \vartheta_0$ and $\mu_\varsigma^1(x_k) = \mu_\varsigma^\infty(x_k)$ for $\varsigma = \vartheta_0$. In (12.4.17) and (12.4.18), we update only one of the 24h control sequences for the battery indicated by ϑ_0 , while the control sequences for all other batteries remain unchanged.

Then, for all ϑ_ℓ , $\ell = 1, 2, \dots$, let $\mathcal{V}_{\vartheta_\ell}^0(x_k) = \mathcal{V}_{\vartheta_{\ell-1}}^\infty(x_k)$ and $\mathfrak{U}_{\vartheta_\ell}(x_k) = [\mu_1^\ell(x_k), \dots, \mu_N^\ell(x_k)]$. We calculate

$$\mu_{\vartheta_\ell}^0(x_k) = \arg \min_{\mu_{\vartheta_\ell,k}} \{ \mathcal{V}(x_k, [\mu_{\vartheta_\ell,k}, \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_\ell}^0(\bar{F}(x_k, [\mu_{\vartheta_\ell,k}, \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)])) \}. \quad (12.4.19)$$

For $\vartheta_\ell \in N_S$ and $\tau = 1, 2, \dots$, the distributed iterative ADP algorithm will proceed between

$$\mathcal{V}_{\vartheta_\ell}^\tau(x_k) = \Upsilon(x_k, [\mu_{\vartheta_\ell}^{\tau-1}(x_k), \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_\ell}^{\tau-1}(\bar{F}(x_k, [\mu_{\vartheta_\ell}^{\tau-1}(x_k), \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)])), \quad (12.4.20)$$

and

$$\mu_{\vartheta_\ell}^\tau(x_k) = \arg \min_{\mu_{\vartheta_\ell,k}} \{ \Upsilon(x_k, [\mu_{\vartheta_\ell,k}, \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_\ell}^\tau(\bar{F}(x_k, [\mu_{\vartheta_\ell,k}, \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)])) \}. \quad (12.4.21)$$

For $\tau \rightarrow \infty$, let $\mathcal{V}_{\vartheta_\ell}^\infty(x_k) = \lim_{\tau \rightarrow \infty} \mathcal{V}_{\vartheta_\ell}^\tau(x_k)$ and $\mu_{\vartheta_\ell}^\infty(x_k) = \lim_{\tau \rightarrow \infty} \mu_{\vartheta_\ell}^\tau(x_k)$. Let $\mu_\varsigma^{\ell+1}(x_k) = \mu_\varsigma^\ell(x_k)$ for $\varsigma \neq \vartheta_\ell$ and $\mu_\varsigma^{\ell+1}(x_k) = \mu_\varsigma^\infty(x_k)$ for $\varsigma = \vartheta_\ell$.

As all the N batteries are operated by the same initial control law, the initial control viable is $\mathfrak{U}_{\vartheta_0}(x_k)$, composing of 24 identical columns. In (12.4.15)–(12.4.21), however, it is emphasized that as the control law of battery ϑ_ℓ , $\ell = 0, 1, \dots$, is updated, the control laws of other batteries are kept unchanged. Then, the control variable becomes $[\mu_{\vartheta_\ell}^{\tau-1}(x_k), \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)]$ and \bar{F} in (12.4.15)–(12.4.21) varies according to ϑ_ℓ . To avoid cumbersome notation, we did not write this explicitly, i.e., \bar{F}^{ϑ_ℓ} would be more appropriate than \bar{F} . When the iteration index τ increases, it is desired that the iterative value function is convergent by updating the control law of battery ϑ_ℓ , $\ell = 0, 1, \dots$.

Theorem 12.4.2 (Local convergence property) *For $\ell = 0, 1, \dots$ and $\tau = 0, 1, \dots$, let $\mathcal{V}_{\vartheta_\ell}^\tau(x_k)$ and $\mu_{\vartheta_\ell}^\tau(x_k)$ be obtained by (12.4.15)–(12.4.21). Then, for $\ell = 0, 1, \dots$, the iterative value function $\mathcal{V}_{\vartheta_\ell}^\tau(x_k)$ is convergent as $\tau \rightarrow \infty$, i.e.,*

$$\mathcal{V}_{\vartheta_\ell}^\infty(x_k) = \lim_{\tau \rightarrow \infty} \mathcal{V}_{\vartheta_\ell}^\tau(x_k),$$

where $\mathcal{V}_{\vartheta_\ell}^\infty(x_k)$ satisfies the following Bellman equation

$$\begin{aligned} \mathcal{V}_{\vartheta_\ell}^\infty(x_k) &= \min_{\mu_{\vartheta_\ell,k}} \{ \Upsilon(x_k, [\mu_{\vartheta_\ell,k}, \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_\ell}^\infty(\bar{F}(x_k, [\mu_{\vartheta_\ell,k}, \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)])) \} \\ &= \Upsilon(x_k, [\mu_{\vartheta_\ell}^\infty(x_k), \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_\ell}^\infty(\bar{F}(x_k, [\mu_{\vartheta_\ell}^\infty(x_k), \bar{\mathfrak{U}}_{\vartheta_\ell}(x_k)])). \end{aligned} \quad (12.4.22)$$

Proof The statement can be proven by mathematical induction. Considering the situation of $\ell = 0$, we will prove that for $\tau = 0, 1, \dots$, the inequality

$$\mathcal{V}_{\vartheta_0}^{\tau+1}(x_k) \leq \mathcal{V}_{\vartheta_0}^\tau(x_k) \quad (12.4.23)$$

holds. First, for $\tau = 0$, according to (12.4.15) and (12.4.16), we can get

$$\begin{aligned}
 \mathcal{V}_{\vartheta_0}^1(x_k) &= \Upsilon(x_k, [\mu_{\vartheta_0}^0(x_k), \bar{\mathfrak{U}}_{\vartheta_0}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_0}^0(\bar{F}(x_k, [\mu_{\vartheta_0}^0(x_k), \bar{\mathfrak{U}}_{\vartheta_0}(x_k)])) \\
 &= \min_{\mu_{\vartheta_0,k}} \{ \Upsilon(x_k, [\mu_{\vartheta_0,k}, \bar{\mathfrak{U}}_{\vartheta_0}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_0}^0(\bar{F}(x_k, [\mu_{\vartheta_0,k}, \bar{\mathfrak{U}}_{\vartheta_0}(x_k)])) \} \\
 &\leq \Upsilon(x_k, [\mu_{\vartheta_0}^0(x_k), \bar{\mathfrak{U}}_{\vartheta_0}(x_k)]) + \tilde{\gamma} J^o(\bar{F}(x_k, [\mu_{\vartheta_0}^0(x_k), \bar{\mathfrak{U}}_{\vartheta_0}(x_k)])) \\
 &= \Gamma(z_k, \mathcal{U}^o(z_k)) + \tilde{\gamma} \mathcal{J}^o(\bar{F}(z_k, \mathcal{U}^o(z_k))) \\
 &= \mathcal{V}_{\vartheta_0}^0(x_k).
 \end{aligned} \tag{12.4.24}$$

Assume that (12.4.23) holds for $\tau = \bar{\tau} - 1, \bar{\tau} = 1, 2, \dots$, i.e., $\mathcal{V}_{\vartheta_0}^{\bar{\tau}}(x_k) \leq \mathcal{V}_{\vartheta_0}^{\bar{\tau}-1}(x_k)$. Then, for $\tau = \bar{\tau}$, we can obtain

$$\begin{aligned}
 \mathcal{V}_{\vartheta_0}^{\bar{\tau}+1}(x_k) &= \min_{\mu_{\vartheta_0,k}} \{ \Upsilon(x_k, [\mu_{\vartheta_0,k}, \bar{\mathfrak{U}}_{\vartheta_0}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_0}^{\bar{\tau}}(\bar{F}(x_k, [\mu_{\vartheta_0,k}, \bar{\mathfrak{U}}_{\vartheta_0}(x_k)])) \} \\
 &\leq \min_{\mu_{\vartheta_0,k}} \{ \Upsilon(x_k, [\mu_{\vartheta_0,k}, \bar{\mathfrak{U}}_{\vartheta_0}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\vartheta_0}^{\bar{\tau}-1}(\bar{F}(x_k, [\mu_{\vartheta_0,k}, \bar{\mathfrak{U}}_{\vartheta_0}(x_k)])) \} \\
 &= \mathcal{V}_{\vartheta_0}^{\bar{\tau}}(x_k).
 \end{aligned} \tag{12.4.25}$$

Hence, for $\ell = 0$, inequality (12.4.23) holds for $\tau = 0, 1, \dots$. As the utility function $\Upsilon(x_k, \mathfrak{U}_k) \geq 0$ and $J^o(x_k) \geq 0$, $\mathcal{V}_{\vartheta_0}^{\tau}(x_k) \geq 0$, which is lower bounded. Thus, the iterative value function is convergent as $\tau \rightarrow \infty$. According to (12.4.20) and (12.4.21), letting $\tau \rightarrow \infty$, we can obtain (12.4.22) for $\ell = 0$. For $\ell = \bar{\ell} - 1, \bar{\ell} = 1, 2, \dots$, we can obtain

$$\begin{aligned}
 \mathcal{V}_{\vartheta_{\bar{\ell}-1}}^{\infty}(x_k) &= \min_{\mu_{\vartheta_{\bar{\ell}-1},k}} \{ \Upsilon(x_k, [\mu_{\vartheta_{\bar{\ell}-1},k}, \bar{\mathfrak{U}}_{\bar{\ell}-1}(x_k)]) \\
 &\quad + \tilde{\gamma} \mathcal{V}_{\vartheta_{\bar{\ell}-1}}^{\infty}(\bar{F}(x_k, [\mu_{\vartheta_{\bar{\ell}-1},k}, \bar{\mathfrak{U}}_{\bar{\ell}-1}(x_k)])) \}.
 \end{aligned} \tag{12.4.26}$$

Then, for $\ell = \bar{\ell}$, we have

$$\mathcal{V}_{\vartheta_{\bar{\ell}}}^0(x_k) = \mathcal{V}_{\vartheta_{\bar{\ell}-1}}^{\infty}(x_k),$$

where $\mathcal{V}_{\vartheta_{\bar{\ell}-1}}^{\infty}(x_k)$ satisfies (12.4.26). According to (12.4.24) and (12.4.25), we can obtain

$$\mathcal{V}_{\vartheta_{\bar{\ell}}}^{\tau+1}(x_k) \leq \mathcal{V}_{\vartheta_{\bar{\ell}}}^{\tau}(x_k),$$

which shows that for $\ell = \bar{\ell}$, the iterative value function $\mathcal{V}_{\vartheta_{\bar{\ell}}}^{\tau}(x_k)$ is convergent as $\tau \rightarrow \infty$. According to (12.4.17) and (12.4.18), letting $\tau \rightarrow \infty$, we can obtain (12.4.22). This completes the proof of the theorem.

Theorem 12.4.2 shows that for $\ell = 0, 1, \dots$, the limit

$$\lim_{\tau \rightarrow \infty} \mathcal{V}_{\vartheta_0}^{\tau}(x_k) = \mathcal{V}_{\vartheta_0}^{\infty}(x_k)$$

can be defined and $\mathcal{V}_{\vartheta_\ell}^\tau(x_k)$ converges to the solution of the Bellman equation (12.4.22) as $\tau \rightarrow \infty$. However, the optimization by a single battery cannot guarantee the iterative value function to converge to the solution of the Bellman equation (12.4.4). Hence, a global convergence analysis will be needed.

Before the next theorem, we define some notations. Let $\mathcal{T}_\varsigma = \{\ell : \vartheta_\ell = \varsigma\}$ and let π_ς be the number of elements in \mathcal{T}_ς .

Theorem 12.4.3 (Global convergence property) *For $\ell = 0, 1, \dots$ and $\tau = 0, 1, \dots$, let $\mathcal{V}_{\vartheta_\ell}^\tau(x_k)$ and $\mu_{\vartheta_\ell}^\tau(x_k)$ be obtained by (12.4.15)–(12.4.21). If the sequence $\{\vartheta_\ell\}$ satisfies*

- (1) *for all $\ell = 0, 1, \dots$, $\vartheta_\ell \in N_\varsigma$,*
- (2) *for all $\varsigma \in N_\varsigma$, $\pi_\varsigma \rightarrow \infty$,*

then for all $\tau = 0, 1, \dots$, the iterative value function $\mathcal{V}_{\vartheta_\ell}^\tau(x_k)$ converges to the optimum, as $\ell \rightarrow \infty$, i.e.,

$$\lim_{\ell \rightarrow \infty} \mathcal{V}_{\vartheta_\ell}^\tau(x_k) = J^*(x_k).$$

Proof The statement can be proven in four steps.

- (1) *Show that for $\tau = 0, 1, \dots$, $\mathcal{V}_{\vartheta_\ell}^\tau(x_k)$ is convergent as $\ell \rightarrow \infty$.*

For $\ell = 0, 1, \dots$ and $\tau = 0, 1, \dots$, define a sequence of iterative value functions as

$$\{\mathcal{V}_{\vartheta_\ell}^\tau(x_k)\} \triangleq \{\mathcal{V}_{\vartheta_0}^0(x_k), \mathcal{V}_{\vartheta_0}^1(x_k), \dots, \mathcal{V}_{\vartheta_0}^\infty(x_k), \mathcal{V}_{\vartheta_1}^0(x_k), \mathcal{V}_{\vartheta_1}^1(x_k), \dots, \mathcal{V}_{\vartheta_1}^\infty(x_k), \dots\}.$$

Let $\sigma_1, \sigma_2, \dots, \sigma_{\pi_\varsigma}$ be the elements in \mathcal{T}_ς such that $\sigma_1 < \sigma_2 < \dots < \sigma_{\pi_\varsigma}$, i.e., $\mathcal{T}_\varsigma = \{\sigma_h : h = 1, 2, \dots, \pi_\varsigma\}$, where $\pi_\varsigma \rightarrow \infty$. For $\tilde{\tau} = 0, 1, \dots$, we define

$$\{\mathcal{V}_{\sigma_h}^{\tilde{\tau}}(x_k)\} \triangleq \{\mathcal{V}_{\sigma_1}^{\tilde{\tau}}(x_k), \mathcal{V}_{\sigma_2}^{\tilde{\tau}}(x_k), \dots, \mathcal{V}_{\sigma_h}^{\tilde{\tau}}(x_k), \dots\}, \quad (12.4.27)$$

then $\{\mathcal{V}_{\sigma_h}^{\tilde{\tau}}(x_k)\}$ is a subsequence of $\{\mathcal{V}_{\vartheta_\ell}^\tau(x_k)\}$. According to (12.4.22) and (12.4.23),

$$\mathcal{V}_{\vartheta_0}^0(x_k) \geq \mathcal{V}_{\vartheta_0}^1(x_k) \geq \dots \geq \mathcal{V}_{\vartheta_0}^\infty(x_k) \geq \mathcal{V}_{\vartheta_1}^0(x_k) \geq \mathcal{V}_{\vartheta_1}^1(x_k) \geq \dots \geq \mathcal{V}_{\vartheta_1}^\infty(x_k) \geq \dots,$$

which means $\{\mathcal{V}_{\vartheta_\ell}^\tau(x_k)\}$ is a monotonically nonincreasing and convergent sequence. For $\tilde{\tau} = 0, 1, \dots$, we can also get $\mathcal{V}_{\sigma_1}^{\tilde{\tau}}(x_k) \geq \mathcal{V}_{\sigma_2}^{\tilde{\tau}}(x_k) \geq \dots$, which shows that (12.4.27) is a monotonically nonincreasing and convergent sequence. Since a sequence can only converge to at most one point [5], the sequence $\{\mathcal{V}_{\vartheta_\ell}^\tau(x_k)\}$ and its subsequence $\{\mathcal{V}_{\sigma_h}^{\tilde{\tau}}(x_k)\}$ possess the same limit, i.e.,

$$\lim_{\ell \rightarrow \infty} \mathcal{V}_{\vartheta_\ell}^\tau(x_k) = \lim_{h \rightarrow \infty} \mathcal{V}_{\sigma_h}^{\tilde{\tau}}(x_k), \quad \forall x_k.$$

- (2) *Show that the limit of the iterative value function $\mathcal{V}_{\vartheta_\ell}^\tau(x_k)$ satisfies the Bellman equation, as $\ell \rightarrow \infty$.*

For all $\tau = 0, 1, \dots$, we define

$$\mathcal{V}_\infty(x_k) = \lim_{\hbar \rightarrow \infty} \mathcal{V}_{\sigma_\hbar}^\tau(x_k). \quad (12.4.28)$$

According to the definition of \mathcal{T}_ς , for $\tau = 0, 1, \dots$, the iterative value function $\mathcal{V}_{\sigma_\hbar}^\tau(x_k)$ is updated by battery ς . As $\hbar \rightarrow \infty$, the control law of battery ς is updated infinite times. Thus, for $\hbar \rightarrow \infty$, the control law of battery ς can be defined as

$$\mu_\varsigma^\tau(x_k) = \arg \min_{\mu_{\varsigma,k}} \{ \Upsilon(x_k, [\mu_{\varsigma,k}, \bar{\mathfrak{U}}_\varsigma(x_k)]) + \tilde{\gamma} \mathcal{V}_\infty(\bar{F}(x_k, [\mu_{\varsigma,k}, \bar{\mathfrak{U}}_\varsigma(x_k)])) \}, \quad (12.4.29)$$

From (12.4.22) and (12.4.23), for $\ell = 0, 1, \dots$, and for $\tau = 1, 2, \dots$, we have

$$\begin{aligned} & \min_{\mu_{\sigma_{\hbar-1},k}} \{ \Upsilon(x_k, [\mu_{\sigma_{\hbar-1},k}, \bar{\mathfrak{U}}_{\sigma_{\hbar-1}}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\sigma_{\hbar-1}}^\infty(\bar{F}(x_k, [\mu_{\sigma_{\hbar-1},k}, \bar{\mathfrak{U}}_{\sigma_{\hbar-1}}(x_k)])) \} \\ &= \mathcal{V}_{\sigma_{\hbar-1}}^\infty(x_k) \geq \mathcal{V}_{\sigma_\hbar}^{\tau+1}(x_k) \\ &= \Upsilon(x_k, [\mu_{\sigma_\hbar}^\tau(x_k), \bar{\mathfrak{U}}_{\sigma_\hbar}(x_k)]) + \tilde{\gamma} \mathcal{V}_{\sigma_\hbar}^\tau(\bar{F}(x_k, [\mu_{\sigma_\hbar}^\tau(x_k), \bar{\mathfrak{U}}_{\sigma_\hbar}(x_k)])) \\ &\geq \mathcal{V}_{\sigma_{\hbar+1}}^\infty(x_k) \\ &= \min_{\mu_{\sigma_{\hbar+1},k}} \{ \Upsilon(x_k, [\mu_{\sigma_{\hbar+1},k}, \bar{\mathfrak{U}}_{\sigma_{\hbar+1}}(x_k)]) \\ &\quad + \tilde{\gamma} \mathcal{V}_{\sigma_{\hbar+1}}^\infty(\bar{F}(x_k, [\mu_{\sigma_{\hbar+1},k}, \bar{\mathfrak{U}}_{\sigma_{\hbar+1}}(x_k)])) \}. \end{aligned}$$

Let $\hbar \rightarrow \infty$, according to (12.4.28) and (12.4.29), we can obtain

$$\mathcal{V}_\infty(x_k) = \min_{\mu_{\varsigma,k}} \{ \Upsilon(x_k, [\mu_{\varsigma,k}, \bar{\mathfrak{U}}_\varsigma(x_k)]) + \tilde{\gamma} \mathcal{V}_\infty(\bar{F}(x_k, [\mu_{\varsigma,k}, \bar{\mathfrak{U}}_\varsigma(x_k)])) \}.$$

For every $\varsigma \in N_\varsigma$, we have $\pi_\varsigma \rightarrow \infty$ and

$$\begin{aligned} \mathcal{V}_\infty(x_k) &= \min_{\mu_{1,k}} \{ \Upsilon(x_k, [\mu_{1,k}, \bar{\mathfrak{U}}_1(x_k)]) + \tilde{\gamma} \mathcal{V}_\infty(\bar{F}(x_k, [\mu_{1,k}, \bar{\mathfrak{U}}_1(x_k)])) \}, \\ &\quad \vdots \\ \mathcal{V}_\infty(x_k) &= \min_{\mu_{N,k}} \{ \Upsilon(x_k, [\mu_{N,k}, \bar{\mathfrak{U}}_N(x_k)]) + \tilde{\gamma} \mathcal{V}_\infty(\bar{F}(x_k, [\mu_{N,k}, \bar{\mathfrak{U}}_N(x_k)])) \}, \end{aligned}$$

which means

$$\mathcal{V}_\infty(x_k) = \min_{\mathfrak{U}_k} \{ \Upsilon(x_k, \mathfrak{U}_k) + \tilde{\gamma} \mathcal{V}_\infty(\bar{F}(x_k, \mathfrak{U}_k)) \}.$$

Next, let $\tilde{\mathfrak{U}}(x_k)$ be an arbitrary admissible control law sequence [2, 27] for the N batteries. Define a new value function $P(x_k)$, which satisfies

$$P(x_k) = \Upsilon(x_k, \tilde{\mathcal{U}}(x_k)) + \tilde{\gamma} P(\bar{F}(x_k, \tilde{\mathcal{U}}(x_k))). \quad (12.4.30)$$

Then, we can proceed to the third step of the proof.

(3) *Show that for an arbitrary admissible control law $\tilde{\mathcal{U}}(x_k)$, the value function $P(x_k) \geq \mathcal{V}_\infty(x_k)$.*

Following similar steps as in the part (2) of the proof of Theorem 4.2.2, we can show that for all $x_k, k = 0, 1, \dots$, the inequality

$$P(x_k) \geq \mathcal{V}_\infty(x_k). \quad (12.4.31)$$

holds.

(4) *Show that the value function $\mathcal{V}_\infty(x_k)$ equals the optimal cost function $J^*(x_k)$.*

According to the definition of $J^*(x_k)$ in (12.4.4), for $\ell = 0, 1, \dots$, we have

$$\mathcal{V}_{\theta_\ell}^\tau(x_k) \geq J^*(x_k).$$

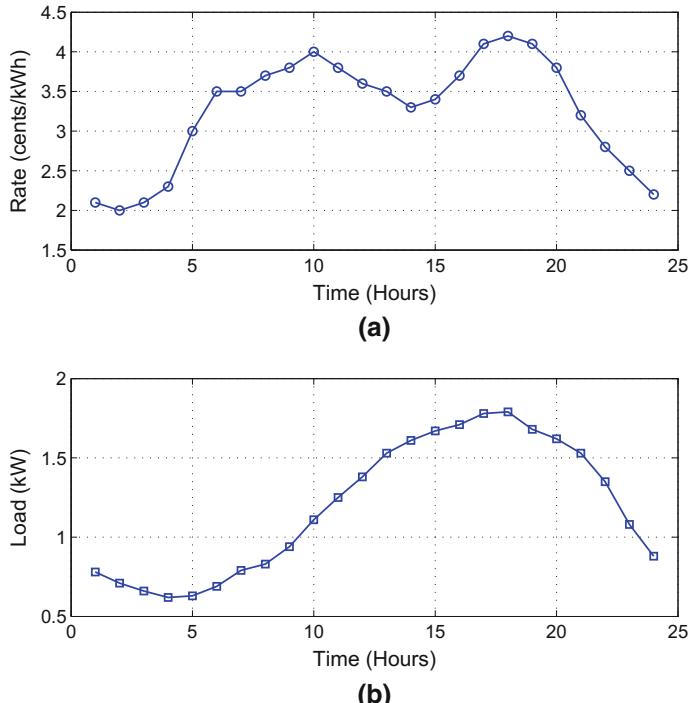


Fig. 12.21 Electricity rate and home load demand. **a** Typical electricity rate for nonsummer seasons. **b** Typical home load demand

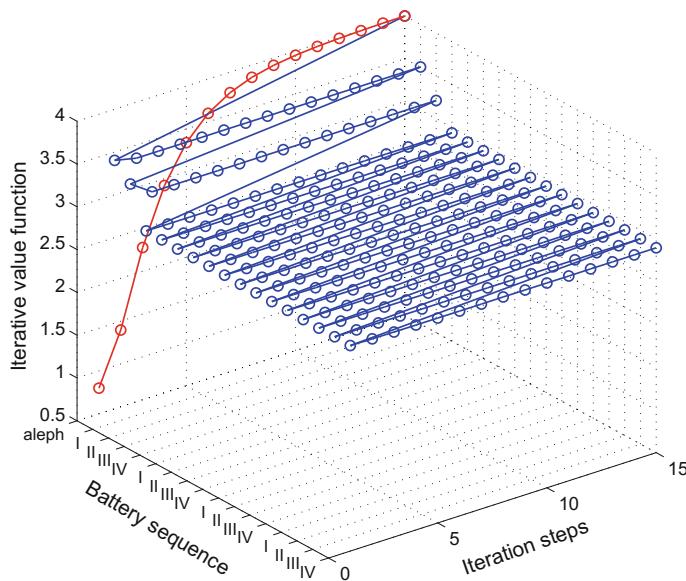


Fig. 12.22 The trajectory of the iterative value function

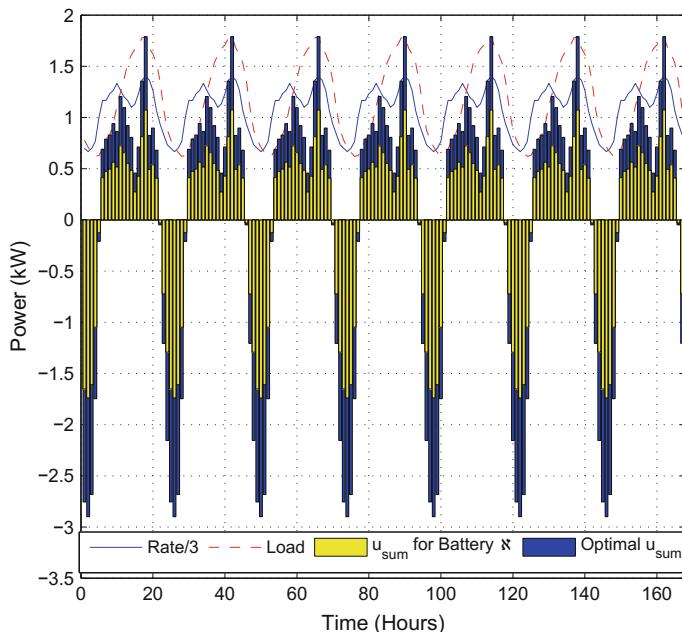


Fig. 12.23 Optimal u_{sum}^* of the batteries in one week

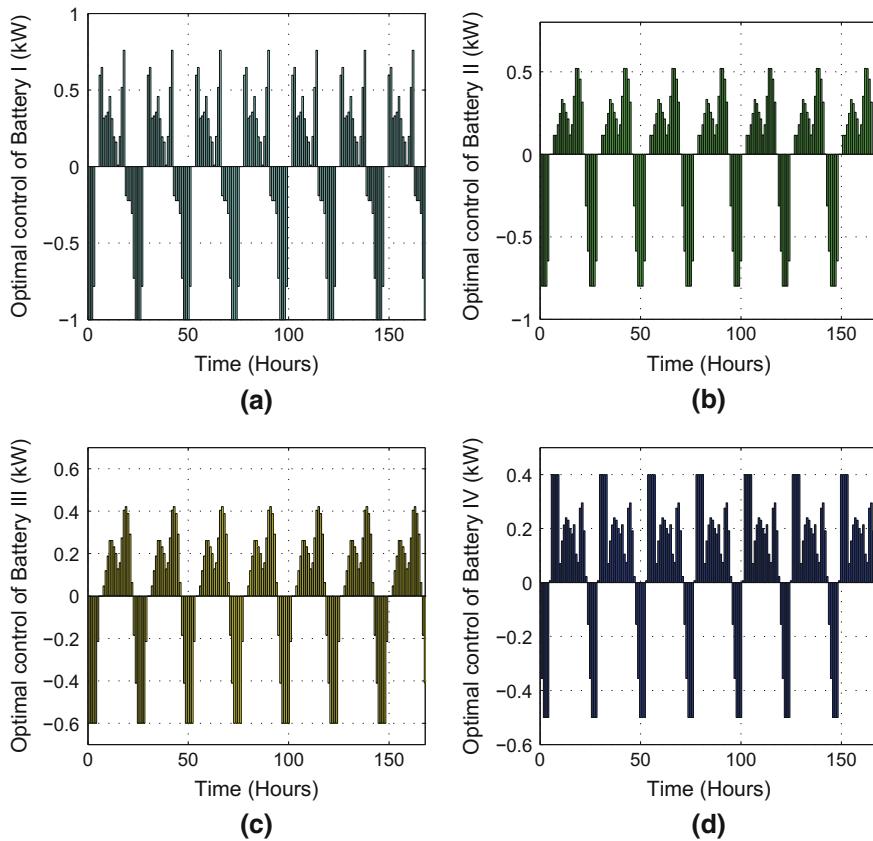


Fig. 12.24 Optimal control of the batteries in one week. **a** Battery I. **b** Battery II. **c** Battery III. **d** Battery IV

Let $\ell \rightarrow \infty$. We can obtain $\mathcal{V}_\infty(x_k) \geq J^*(x_k)$. On the other hand, for an arbitrary admissible control law $\tilde{\mathcal{U}}(x_k)$, (12.4.31) holds. Let $\tilde{\mathcal{U}}(x_k) = \mathcal{U}^*(x_k)$, where $\mathcal{U}^*(x_k)$ is an optimal control law. Then, we can get $\mathcal{V}_\infty(x_k) \leq J^*(x_k)$. Hence, we can obtain $\lim_{\ell \rightarrow \infty} \mathcal{V}_{\theta_\ell}(x_k) = J^*(x_k)$. The proof is complete.

Based on the above preparations, we now summarize the distributed iterative ADP algorithm for the multi-battery home energy management systems in Algorithm 12.4.1. We can see that there are two iteration blocks in Algorithm 12.4.1. In Block 1, iterations are implemented to obtain the optimal control for Battery \aleph . In Block 2, the optimal multi-battery coordination control is obtained.

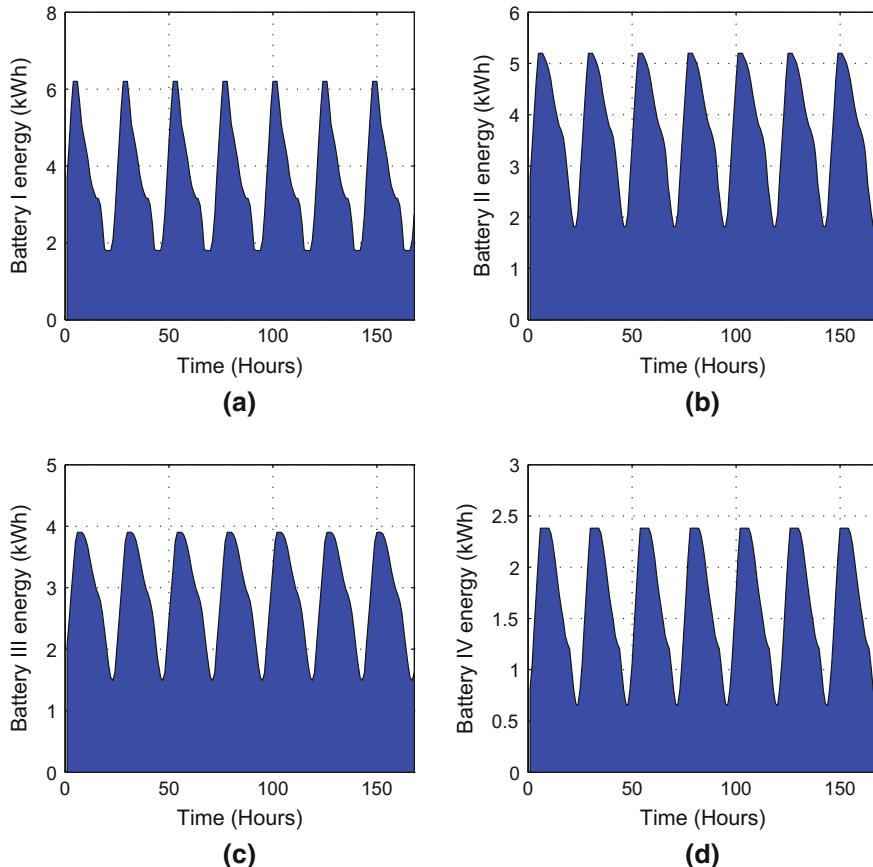


Fig. 12.25 The energy of batteries in one week. **a** Battery I. **b** Battery II. **c** Battery III. **d** Battery IV

Table 12.2 Cost comparison

	Original	Battery \aleph	DIADP	TBQL	PSO
Total cost (cents)	683.10	590.35	499.86	548.33	556.71
Saving (%)		13.57	26.82	19.73	18.50

12.4.2 Numerical Analysis

In this section, the performance of the present distributed iterative ADP algorithm will be examined by numerical experiments. Choose the profiles of real-time electricity rate in nonsummer seasons from ComEd Company in [14], and choose the home load demand from NAHB Research Report in [35], where trajectories of the electricity rate and the home load demand are shown in Fig. 12.21a and b, respectively.

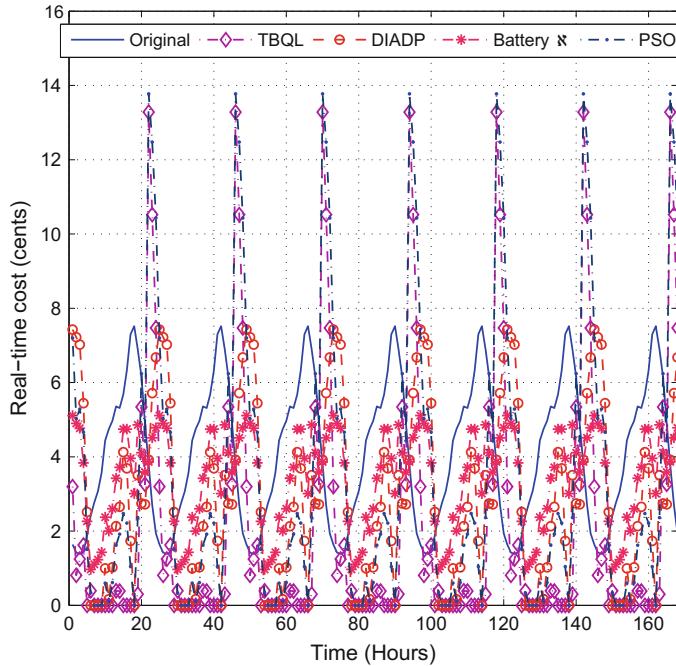


Fig. 12.26 Real-time cost comparison

We assume to have four batteries in the home energy management system, which are denoted by [Battery I, ..., Battery IV], respectively. Define the capacities of the battery as [8, 6, 5, 3] kWh. Let the upper and lower storage limits of the batteries, i.e., $E_{b\varsigma}^{\max}$ and $E_{b\varsigma}^{\min}$, $\varsigma = I, \dots, IV$, be given by [6.2, 5.2, 3.9, 2.4] kWh and [1.8, 1.6, 1.4, 0.6] kWh, respectively. Let the charging and discharging power limits, i.e., $P_{b\varsigma}^{\min}$ and $P_{b\varsigma}^{\max}$, be given by [-1, -0.8, -0.6, -0.4] kW and [1, 0.8, 0.6, 0.4] kW. Let the rated power output of the batteries be the same 0.8 kW and the max efficiency constants be [0.898, 0.798, 0.698, 0.598]. Let the cost function be expressed as in (12.4.1), where for $\varsigma = I, \dots, IV$, we set $\alpha = 1$, $\beta_\varsigma = 0.2$, $r_\varsigma = 0.1$, and $\gamma = 0.995$. Let the initial function $\Psi(z_k, u_k) = [z_k^\top, u_k^\top]P[z_k^\top, u_k^\top]^\top$, where $P = I$ is the identity matrix with a suitable dimension. Choose the computation precision $\varepsilon = 10^{-3}$. First, transform all the four batteries into Battery \aleph , with the parameters $\bar{E}_b = 1.8$ kWh, $\bar{E}_b = 0$ kWh, $\bar{P}^{\text{rate}} = 0.8$ kW, $\bar{P}_b^{\min} = -0.4$ kW, and $\bar{P}_b^{\max} = 0.4$ kW. After the data normalization, implementing the iterative ADP algorithm (12.4.9)–(12.4.11) for 15 iterations, the iterative value function is shown in Fig. 12.22, where we can see that the iterative value function will converge to its optimum. Let $u_{\text{sum}} = \sum_{i=1}^4 u_i$ be the sum of controls, and the optimal control u_{sum}^* based on Battery \aleph for 168 h (one week) is shown in Fig. 12.23.

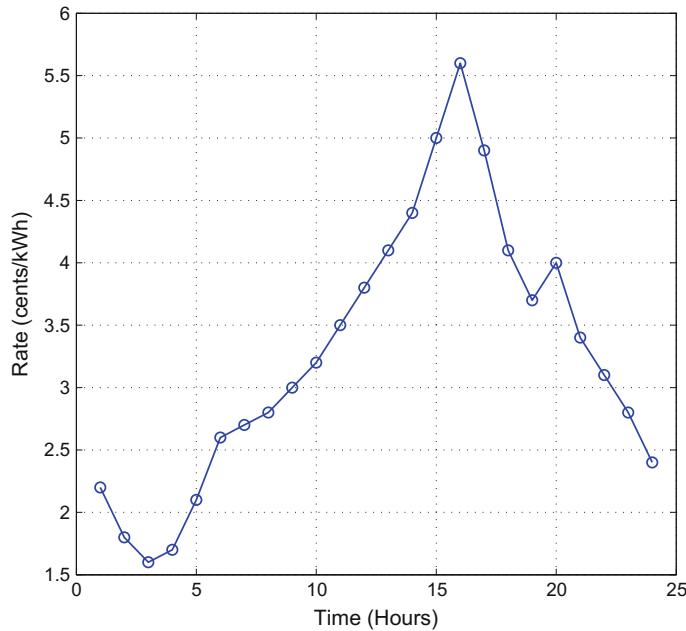


Fig. 12.27 Typical electricity rate in summer

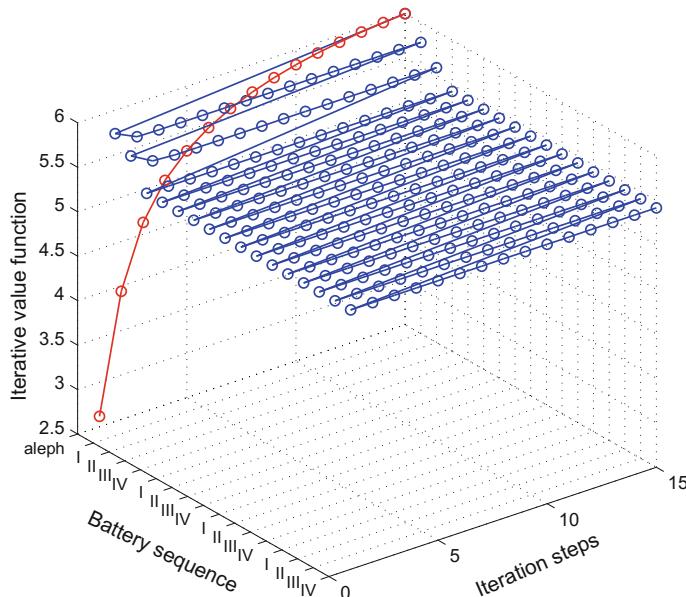


Fig. 12.28 The trajectory of the iterative value function in summer

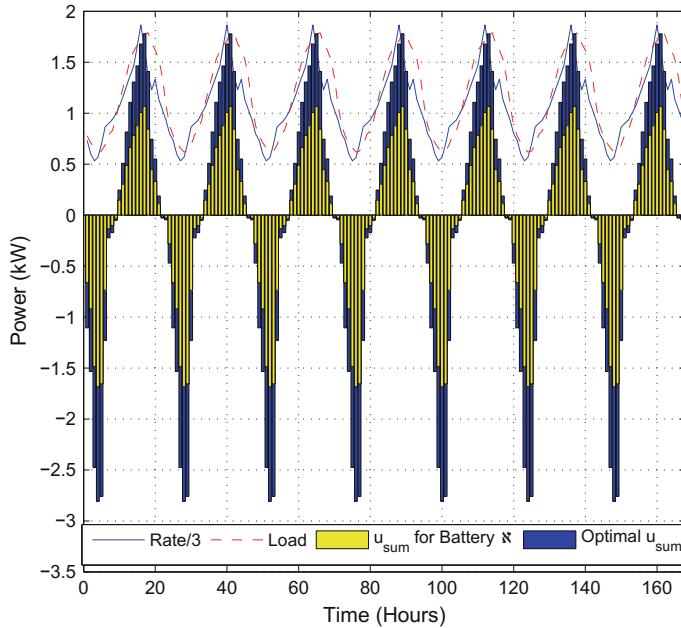


Fig. 12.29 Optimal u_{sum}^* of the batteries for one week in summer

Based on the optimal control law of Battery 8, the distributed iterative ADP algorithm is implemented. Let the optimization sequence of the batteries be chosen as $\{I, II, \dots, IV, I, II, \dots, IV, \dots\}$. The trajectory of the iterative value function is shown in Fig. 12.22, where we can see that the iterative value function is monotonically nonincreasing and converges to the optimum. The optimal sum control u_{sum} for the four batteries is shown in Fig. 12.23, where the charging/discharging power is obviously larger than the one of Battery 8. The optimal control laws for Batteries I, ..., IV are shown in Fig. 12.24, and the energies of the batteries are displayed in Fig. 12.25.

Next, time-based Q-learning (TBQL) algorithm [22] and particle swarm optimization (PSO) algorithm [17] will be compared to illustrate the superiority of our ADP algorithm. For $t = 0, 1, \dots$, the goal of TBQL algorithm [22] is to design an iterative control that satisfies the following optimality equation $Q(x_{t-1}, u_{t-1}) = U(x_t, u_t) + \gamma Q(x_t, u_t)$.

Three-layer backpropagation (BP) neural networks are implemented to approximate the Q-function. The detailed neural network implementation of TBQL can be seen in [8, 22, 38], which is omitted here. For PSO algorithm [17], let $\mathcal{G} = 100$ be the swarm size. The position of each particle at time t is represented by $x_{\ell t}$, $\ell = 1, 2, \dots, \mathcal{G}$, and its movement by the velocity vector $v_{\ell t}$. Then, the updating rule of PSO can be expressed as

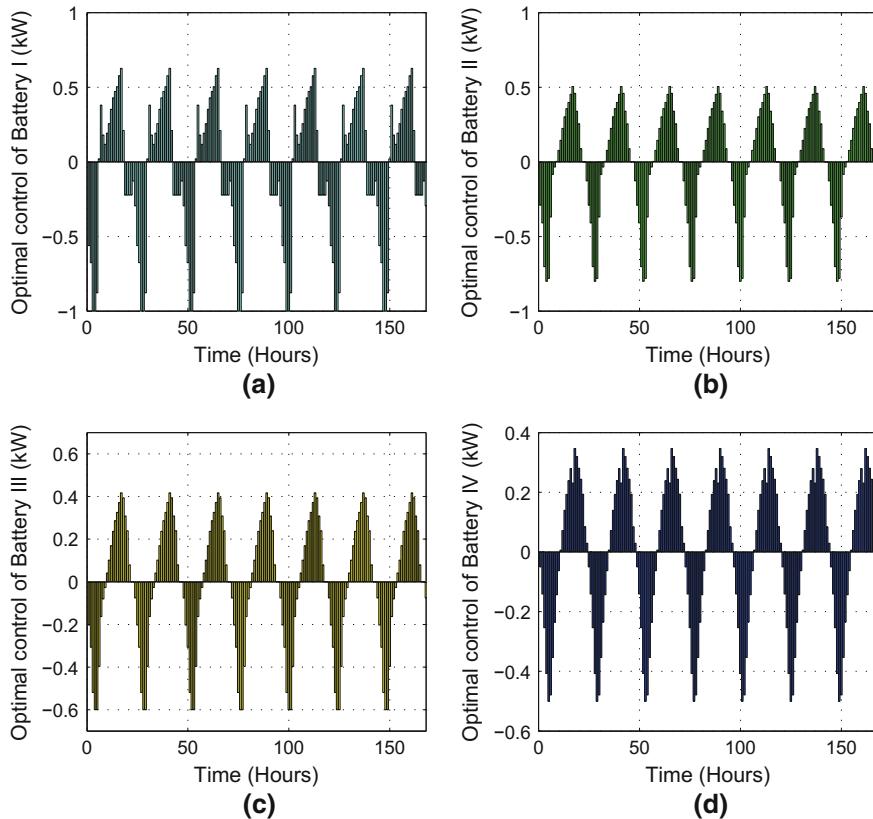


Fig. 12.30 Optimal control of the batteries for one week in summer. **a** Battery I. **b** Battery II. **c** Battery III. **d** Battery IV

$$x_{\ell t} = x_{\ell(t-1)} + v_{\ell t},$$

$$v_{\ell t} = \omega v_{\ell(t-1)} + \varphi_1 \rho_1^\top (p_\ell - x_{\ell(t-1)}) + \varphi_2 \rho_2^\top (p_g - x_{\ell(t-1)}).$$

Let the inertia factor be $\omega = 0.7$. Let the correction factors $\rho_1 = \rho_2 = 1$. Let φ_1 and φ_2 be random numbers in $[0, 1]$. Let p_ℓ be the best position of particles, and let p_g be the global best position. Implement the TBQL for 500 time steps, and implement the PSO algorithm for 500 iterations. The comparison of the total cost for 168 h is displayed in Table 12.2, where we can see that the minimum cost is obtained by our distributed iterative ADP algorithm. Let the real-time cost function be $R_{ct} = C_t P_{gt}$, and the corresponding real-time cost functions are shown in Fig. 12.26, where the term “original” denotes “no battery system” and “DIADP” denotes “distributed iterative ADP algorithm.” According to the numerical comparisons, the superiority of our distributed iterative ADP algorithm can be verified.

Algorithm 12.4.1 The distributed iterative ADP algorithm**Initialization:**

Collect an array of system data for the residential energy system (12.4.2).

Give a positive semidefinite function $\Psi(x_k)$.

Give the computation precision $\varepsilon > 0$.

Give a sequence $\{\vartheta_0, \vartheta_1, \dots\}$, where $\vartheta_\ell \in N_\varsigma$, $\ell = 0, 1, \dots$

Iteration:

Step 1. Construct Battery \aleph by (12.4.5). Establish the home energy management system by (12.4.6) and transform the cost function by (12.4.7).

Block 1.....

Step 2. For $i = 0$, define the initial value function $V^0(z_k) = \Psi(z_k)$. Calculate $\mathcal{A}^0(z_k)$ by (12.4.9).

Step 3. Let $i = i + 1$. The iterative value function $V^i(z_k)$ is obtained by (12.4.10).

Step 4. Obtain the iterative control sequence $\mathcal{A}^i(z_k)$ by (12.4.11).

Step 5. If $|V^i(z_k) - V^{i-1}(z_k)| \leq \varepsilon$, then goto next step; else, goto the Step 3.

Step 6. Let $\mathcal{J}^o(z_k) = V^i(z_k)$.

Block 2.....

Step 8. For $\ell = 0$, let $\mathcal{V}_{\vartheta_0}^0(x_k) = \mathcal{J}^o(z_k)$. Calculate $\mu_{\vartheta_0}^0(x_k)$ by (12.4.16). Let $\tau = 1$.

Step 9. Let the iterative value function $\mathcal{V}_{\vartheta_0}^\tau(x_k)$ and the iterative control law $\mu_{\vartheta_0}^\tau(x_k)$ be obtained by (12.4.17) and (12.4.18), respectively.

Step 10. If $|\mathcal{V}_{\vartheta_0}^\tau(x_k) - \mathcal{V}_{\vartheta_0}^{\tau-1}(x_k)| \leq \varepsilon$, then let $\mathcal{V}_{\vartheta_1}^0(x_k) = \mathcal{V}_{\vartheta_0}^\tau(x_k)$, let $\ell = 1$, let $\tau = 1$, and goto Step 11; else, let $\tau = \tau + 1$ and goto Step 9.

Step 11. Obtain the iterative value function $\mathcal{V}_{\vartheta_\ell}^\tau(x_k)$ and the iterative control law $\mu_{\vartheta_\ell}^\tau(x_k)$ by (12.4.20) and (12.4.21), respectively.

Step 12. If $|\mathcal{V}_{\vartheta_\ell}^\tau(x_k) - \mathcal{V}_{\vartheta_\ell}^{\tau-1}(x_k)| \leq \varepsilon$, then let $\mathcal{V}_{\vartheta_{\ell+1}}^0(x_k) = \mathcal{V}_{\vartheta_\ell}^\tau(x_k)$ and goto the next step; else, let $\tau = \tau + 1$ and goto Step 11.

Step 13. If $|\mathcal{V}_{\vartheta_{\ell+1}}^0(x_k) - \mathcal{V}_{\vartheta_\ell}^0(x_k)| \leq \varepsilon$, then goto the next step; else, let $\tau = 1$, $\ell = \ell + 1$, and goto Step 11.

Step 14. Construct the control law sequence as $\mathfrak{U}_{\vartheta_\ell}(x_k) = [\mu_1^\ell(x_k), \dots, \mu_N^\ell(x_k)]$. Determine the optimal control law sequence $\mathfrak{U}^*(x_k) = \mathfrak{U}_{\vartheta_\ell}(x_k)$ and the optimal value function $J^*(x_k) = \mathcal{V}_{\vartheta_{\ell+1}}^0(x_k)$.

Step 15. Return $\mathfrak{U}^*(x_k)$ and $J^*(x_k)$.

It should be pointed out that if the residential environment, such as the electricity rate, is changed, the optimal multi-battery control will change correspondingly. It is pointed out by ComEd Company [14] that in summer, the electricity rate is different from other seasons, which is shown in Fig. 12.27. Let all the other parameters remain unchanged, and implement the iterative ADP algorithm. The iterative value function is shown in Fig. 12.28, where the nonincreasing monotonicity and optimality are verified. The optimal control of u_{sum} for 168 h in summer is shown in Fig. 12.29. The corresponding optimal multi-battery coordination controls are shown in Fig. 12.30a–d, respectively. Next, implementing TBQL and PSO algorithms for the summer data, the total costs by TBQL and PSO, and the present iterative ADP algorithms for one week are shown in Table 12.3, where the superiority of our distributed iterative ADP algorithm can be verified.

Table 12.3 Cost comparison

	Original	Battery &	DIADP	TBQL	PSO
Total cost (cents)	722.62	607.17	530.21	577.86	590.18
Saving (%)		15.98	26.63	20.03	18.33

12.5 Conclusions

Given the residential load and the real-time electricity rate, the objective of the optimal control in this chapter 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. First, the ADP scheme based on ADHDP that is suitable for applications to residential energy system control and management problem is introduced. Then, a new iterative ADP algorithm, called dual iterative Q-learning algorithm, is developed to solve the optimal battery management and control problem in smart residential environments. The main idea of the present dual iterative Q-learning algorithm is to update the iterative value function and iterative control laws by ADP technique according to the i -iteration and the j -iteration, respectively. The convergence and optimality of the present algorithm are established. Next, optimal multi-battery management problems in smart home energy management systems are solved by iterative ADP algorithms. To obtain the optimal coordination control law for multi-batteries, a new distributed iterative ADP algorithm is developed, which avoids the increasing dimension of controls. Convergence properties are developed to guarantee the optimality of the algorithm. Neural networks are introduced to implement these ADP algorithms. Effectiveness of the algorithms is verified by numerical results.

References

1. Aksoy S, Haralick RM (2001) Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognit Lett* 22(5):563–582
2. Al-Tamimi A, Lewis FL, Abu-Khalaf M (2008) Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Trans Syst Man Cybern Part B: Cybern* 38(4):943–949
3. Amjadi Z, Williamson SS (2010) Power-electronics-based solutions for plug-in hybrid electric vehicle energy storage and management systems. *IEEE Trans Ind Electron* 57(2):608–616
4. Angelis FD, Boaro M, Fuselli D, Squartini S, Piazza F, Wei Q (2013) Optimal home energy management under dynamic electrical and thermal constraints. *IEEE Trans Ind Inf* 9(3):1518–1527
5. Apostol TM (1974) Mathematical analysis, 2nd edn. Addison-Wesley, Boston
6. Bakirtzis AG, Dokopoulos PS (1988) Short term generation scheduling in a small autonomous system with unconventional energy system. *IEEE Trans Power Syst* 3(3):1230–1236
7. Bellman RE (1957) Dynamic programming. Princeton University Press, Princeton

8. Boaro M, Fuselli D, Angelis FD, Liu D, Wei Q, Piazza F (2013) Adaptive dynamic programming algorithm for renewable energy scheduling and battery management. *Cogn Comput* 5(2):264–277
9. Chacra FA, Bastard P, Fleury G, Clavreul R (2005) Impact of energy storage costs on economical performance in a distribution substation. *IEEE Trans Power Syst* 20:684–691
10. Chaouachi A, Kamel RM, Andoulsi R, Nagasaka K (2013) Multiobjective intelligent energy management for a microgrid. *IEEE Trans Ind Electron* 60(4):1688–1699
11. Chen C, Duan S, Cai T, Liu B, Yin J (2009) Energy trading model for optimal microgrid scheduling based on genetic algorithm. In: Proceedings of the IEEE International Power Electronics and Motion Control Conference, pp 2136–2139
12. ComEd, Real time pricing in USA. [Online Available] <http://www.thewattspot.com>
13. Corrigan PM, Heydt GT (2007) Optimized dispatch of a residential solar energy system. In: Proceedings of the North American power symposium. pp 4183–4188
14. Data of electricity rate from ComEd Company, USA. [Online Available] <https://rrtp.comed.com/live-prices>
15. Dierks T, Jagannathan S (2012) Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update. *IEEE Trans Neural Netw Learn Syst* 23(7):1118–1129
16. Fung CC, Ho SCY, Nayar CV (1993) Optimisation of a hybrid energy system using simulated annealing technique. In: Proceedings of the ieee region 10 conference on computer, communication, control and power engineering, pp 235–238
17. Fuselli D, Angelis FD, Boaro M, Liu D, Wei Q, Squartini S, Piazza F (2013) Action dependent heuristic dynamic programming for home energy resource scheduling. *Int J Electr Power Syst* 48:148–160
18. Gudi N, Wang L, Devabhaktuni V, Depuru SSSR (2011) A demand-side management simulation platform incorporating optimal management of distributed renewable resources. In: Proceedings of the ieee/pes power systems conference and exposition, pp 1–7
19. Guerrero JM, Loh PC, Lee TL, Chandorkar M (2013) Advanced control architectures for intelligent microgrids-part II: power quality, energy storage, and AC/DC microgrids. *IEEE Trans Ind Electron* 60(4):1263–1270
20. Hagan MT, Menhaj MB (1994) Training feedforward networks with the Marquardt algorithm. *IEEE Trans Neural Netw* 5(6):989–993
21. Huang T, Liu D (2011) Residential energy system control and management using adaptive dynamic programming. In: Proceedings of the ieee international joint conference on neural networks, pp 119–124
22. Huang T, Liu D (2013) A self-learning scheme for residential energy system control and management. *Neural Comput Appl* 22(2):259–269
23. Jian L, Xue H, Xu G, Zhu X, Zhao D, Shao ZY (2013) Regulated charging of plug-in hybrid electric vehicles for minimizing load variance in household smart microgrid. *IEEE Trans Ind Electron* 60(8):3218–3226
24. Lee TY (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 Trans Energy Convers* 22(3):774–782
25. Lendaris GG, Paintz C (1997) Training strategies for critic and action neural networks in dual heuristic programming method. In: Proceedings of the ieee international conference neural networks, pp 712–717
26. Lincoln B, Rantzer A (2006) Relaxing dynamic programming. *IEEE Trans Autom Control* 51(8):1249–1260
27. Liu D, Wei Q (2014) Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Trans Neural Netw Learn Syst* 25(3):621–634
28. Liu D, Xiong X, Zhang Y (2001) Action-dependent adaptive critic designs. In: Proceedings of the international joint conference on neural networks, pp 990–995
29. Liu D, Zhang Y, Zhang H (2005) A self-learning call admission control scheme for CDMA cellular networks. *IEEE Trans Neural Netw* 16(5):1219–1228

30. Lu B, Shahidehpour M (2005) Short-term scheduling of battery in a grid-connected PV/battery system. *IEEE Trans Power Syst* 20(2):1053–1061
31. Lu X, Sun K, Guerrero JM, Vasquez JC, Huang L (2014) State-of-charge balance using adaptive droop control for distributed energy storage systems in DC microgrid applications. *IEEE Trans Ind Electron* 61(6):2804–2815
32. Maly DK, Kwan KS (1995) Optimal battery energy storage system (BESS) charge scheduling with dynamic programming. *IEE Proc Sci Meas Technol* 142(6):453–458
33. Moraes H, Kádár P, Faria P, Vale ZA, Khodr HM (2010) Optimal scheduling of a renewable micro-grid in an isolated load area using mixed-integer linear programming. *Renew Energy* 35(1):151–156
34. Motapon SN, Dessaint LA, Al-Haddad K (2014) A comparative study of energy management schemes for a fuel-cell hybrid emergency power system of more-electric aircraft. *IEEE Trans Ind Electron* 61(3):1320–1334
35. NAHB Research Center (2014) Review of residential electrical energy use data. 400 Prince George's Boulevard, Upper Marlboro, MD, USA, July 16, 2001. [Online Available] <http://www.toolbase.org/PDF/CaseStudies/Res-Electrical-EnergyUseData.pdf>
36. Rahimi-Eichi H, Baronti F, Chow MY (2014) Online adaptive parameter identification and state-of-charge coestimation for lithium-polymer battery cells. *IEEE Trans Ind Electron* 61(4):2053–2061
37. Riffonneau Y, Bacha S, Barruel F, Ploix S (2011) Optimal power flow management for grid connected PV systems with batteries. *IEEE Trans Sustain Energy* 2(3):309–320
38. Si J, Wang YT (2001) On-line learning control by association and reinforcement. *IEEE Trans Neural Netw* 12(2):264–276
39. Smolenski R, Bojarski J, Kempski A, Lezynski P (2014) Time-domain-based assessment of data transmission error probability in smart grids with electromagnetic interference. *IEEE Trans Ind Electron* 61(4):1882–1890
40. Watkins C (1989) Learning from delayed rewards. Ph.D. Thesis, Cambridge University, Cambridge, England
41. Watkins C, Dayan P (1992) Q-learning. *Mach Learn* 8(3–4):279–292
42. Wei Q, Liu D, Shi G (2015) A novel dual iterative Q-learning method for optimal battery management in smart residential environments. *IEEE Trans Ind Electron* 62(4):2509–2518
43. Wei Q, Liu D, Shi G, Liu Y (2015) Multibattery optimal coordination control for home energy management systems via distributed iterative adaptive dynamic programming. *IEEE Trans Ind Electron* 62(7):4203–4214
44. Yau T, Walker LN, Graham HL, Raithel R (1981) Effects of battery storage devices on power system dispatch. *IEEE Trans Power Appar Syst* PAS-100(1):375–383

Chapter 13

Adaptive Dynamic Programming for Optimal Control of Coal Gasification Process

13.1 Introduction

Coal is the world's most abundant energy resource and the cheapest fossil fuel. The development of coal gasification technologies, which is a primary component of the carbon-based process industries, is of primary importance to deal with the limited petroleum reserves [12]. Hence, optimal control for the coal gasification process is a key problem of carbon-based process industries. To describe the process of coal gasification, many discussions focus on coal gasification modeling approaches [1, 13, 14, 19]. The established models are usually very complex with high nonlinearities. To simplify the controller design, the traditional control method for the coal gasification process adopts feedback linearization control method [6, 10, 18]. However, the controller designed by feedback linearization technique is only effective in the neighborhood of the equilibrium point. When the required operating range is large, the nonlinearities in the system cannot be properly compensated by using a linear model. Therefore, it is necessary to study an optimal control approach for the original nonlinear system [4, 7–9, 17]. But to the best of our knowledge, there are no discussions on the optimal controller design for the nonlinear coal gasification processes. One of the difficulties is the complexity of the coal gasification processes, which leads to very complex expressions for the optimal control law. Generally speaking, the optimal control law cannot be expressed analytically. Another difficulty to obtain the optimal control law lies in solving the time-varying Bellman equation which is usually too difficult to solve analytically. On the other hand, in the real-world control systems of coal gasification processes, the coal quality is also unknown for control systems. This makes it more difficult to obtain the optimal control law of the coal gasification systems. To overcome these difficulties, iterative ADP algorithm will be employed.

For the coal gasification process, the accurate system model is complex and cannot be obtained in general. In each iteration of the iterative ADP algorithms, the accurate iterative control laws and the cost function cannot be accurately obtained either. In this situation, approximation structures, such as neural networks, can be

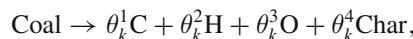
used to approximate the system model, the iterative control law, and the iterative value function, respectively. So, there must exist approximation errors between the approximated functions and the expected ones, no matter what the approximation precisions are obtained. When the accurate system model, iterative control laws, and the iterative value function cannot be obtained, the convergence properties of the accurate iterative ADP algorithms may be invalid. Till now, only in [11], approximation errors for the iterative control law and iterative value function in the iterative ADP algorithm were considered, but the accurate system model is required. To the best of our knowledge, there are no discussions on the optimal control scheme of the iterative ADP algorithms, where modeling errors of unknown systems and iteration errors are both considered. In this chapter, an integrated self-learning optimal control method of the coal gasification process using iterative ADP is developed, where modeling errors and iteration errors are considered [16].

13.2 Data-Based Modeling and Properties

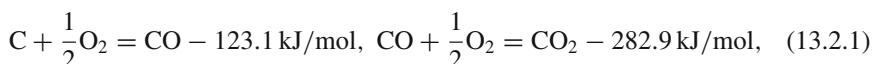
13.2.1 Description of Coal Gasification Process and Control Systems

The coal gasification inputs the coal water slurry (including coal and water) and combines with oxygen into the gasifier. The coal gasification process in the gasifier operates at a high temperature and the output of coal gasification process include synthesis gas and char. The diagram of coal gasification process is given in Fig. 13.1.

The composition of coal contains carbon (C), hydrogen (H), oxygen (O), and char (Char), which is expressed by



where $\sum_{i=1}^4 \theta_k^i = 1$ and $k = 0, 1, \dots$, is the discrete-time. Let $\Theta_k = [\theta_k^1, \theta_k^2, \theta_k^3, \theta_k^4]^T$ denote the coal quality function. The coal gasification reaction can be classified into two phases [13]. The first phase is coal combustion reaction, and the chemical equations are expressed by



where CO is carbon monoxide and CO₂ is carbon dioxide. The other phase is water gas shift reaction which is reversible and mildly exothermic



where H₂O is water.

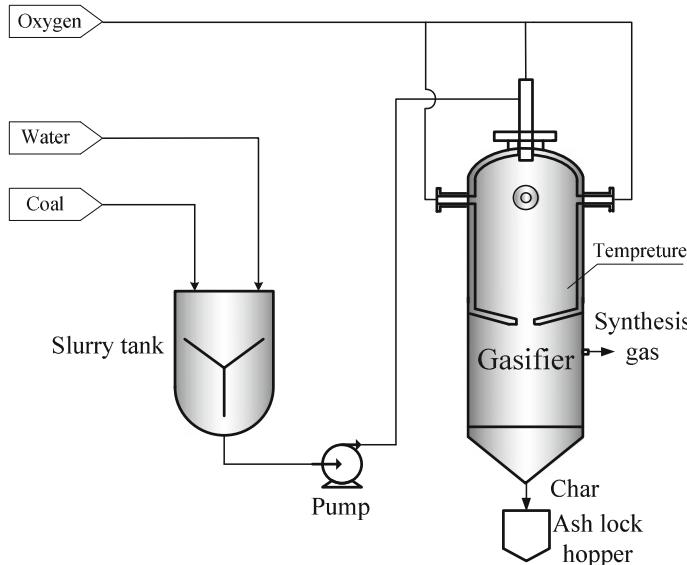


Fig. 13.1 Flow diagram of coal gasification process

The coal combustion reaction is instantaneous and nonreversible. The water gas shift reaction is reversible and the reaction is strongly dependent on the reaction temperature. Let x_k be the reaction temperature and let T_k denote the reaction equilibrium coefficient. Then, we have the following empirical formula [13]

$$T_k = \frac{n_{CO_2} \cdot n_{H_2}}{n_{CO} \cdot n_{H_2O}} = \left(\frac{202.362}{x_k - 635.52} \right)^{0.921914}, \quad (13.2.3)$$

where $n_{(.)}$ denotes the molar quantity in the synthesis gas.

For the coal gasification process, it is pointed out that the reaction temperature is the key parameter [5, 13]. Hence, in this chapter, an optimal control scheme will be established to guarantee the reaction temperature to track effectively a desired one.

Let $P_{(.)}^k$ denote the flow of the control input (kg/h) and $R_{(.)}^k$ denote the flow of the output (kg/h). Then, the control input vector is defined as

$$u_k = [u_{1k}, u_{2k}, u_{3k}]^T = [P_{coal}^k, P_{H_2O}^k, P_{O_2}^k]^T. \quad (13.2.4)$$

The system output vector is defined as

$$y_k = [y_{1k}, y_{2k}, y_{3k}, y_{4k}, y_{5k}]^T = [R_{CO}^k, R_{CO_2}^k, R_{H_2}^k, R_{H_2O}^k, R_{Char}^k]^T. \quad (13.2.5)$$

According to (13.2.1)–(13.2.5), the coal gasification control system can now be expressed as

$$\begin{aligned} x_{k+1} &= F(x_k, u_k, \Theta_k) \\ y_k &= G(x_k, u_k, \Theta_k), \end{aligned} \quad (13.2.6)$$

where F and G are unknown system functions.

Let the desired state trajectory be η . Then, our goal is to design an optimal state-feedback tracking control law $u_k^* = u^*(x_k)$, such that the system state tracks the desired state trajectory. However, it is nearly impossible to obtain a direct optimal tracking controller for system (13.2.6). First, the system functions F and G are unknown nonlinear functions. Second, for desired trajectory η , the corresponding reference control is also difficult to obtain for the unknown system. Furthermore, the coal quality Θ_k is also an unknown and uncontrollable parameter. Thus, novel methods must be established to solve these problems.

13.2.2 Data-Based Process Modeling and Properties

In this section, three-layer backpropagation (BP) NNs are employed to approximate the system (13.2.6). We also use NNs to solve the reference control and obtain the coal quality. Let the number of hidden layer neurons be denoted by L . Let the weight matrix between the input layer and hidden layer be denoted by Y_f . Let the weight matrix between the hidden layer and output layer be denoted by W_f . Let the input vector of the NN be denoted as X . Then, the output of three-layer NN is represented by:

$$\hat{F}_N(X, Y_f, W_f) = W_f \bar{\sigma}(Y_f X), \quad (13.2.7)$$

where $\bar{\sigma}(Y_f X) \in \mathbb{R}^L$, $[\bar{\sigma}(\zeta)]_i = \frac{\exp(\zeta_i) - \exp(-\zeta_i)}{\exp(\zeta_i) + \exp(-\zeta_i)}$, $i = 1, \dots, L$, are the activation function.

The NN estimation error can be expressed by

$$F_N(X) = \hat{F}_N(X, Y_f^*, W_f^*) + \varepsilon(X),$$

where Y_f^* and W_f^* are the ideal weight parameters, $\varepsilon(X)$ is the reconstruction error. For the convenience of analysis, only the output weights W_f are updated during the training, while the hidden weights Y_f are kept fixed [3, 20]. Hence, the NN function (13.2.7) will be simplified by the expression $\hat{F}_N(X, W_f) = W_f \sigma_f(X)$, where $\sigma_f(X) = \bar{\sigma}(Y_f X)$.

Next, using input-state-output data, two BP NNs are used to reconstruct the system (13.2.6). Let the number of hidden layer neurons be denoted by L_{m1} and L_{m2} . Let the ideal weights be denoted by W_{m1}^* and W_{m2}^* , respectively. According to the universal

approximation property of NNs, the NN representation of the system (13.2.6) can be written as

$$x_{k+1} = W_{m1}^* \sigma_1(z_k) + \varepsilon_{m1k},$$

$$y_k = W_{m2}^* \sigma_2(z_k) + \varepsilon_{m2k},$$

where $z_k = [x_k, u_k^\top, \Theta_k^\top]^\top$ is the NN input.

Let $\sigma_1(z_k) = \bar{\sigma}(Y_1 z_k)$ and $\sigma_2(z_k) = \bar{\sigma}(Y_2 z_k)$, where Y_1 and Y_2 are arbitrary matrices with suitable dimensions. Let $\max\{\|\sigma_1(\cdot)\|, \|\sigma_2(\cdot)\|\} \leq \sigma_M$ for a constant σ_M . Let ε_{mik} be the bounded NN reconstruction errors such that $\|\varepsilon_{mik}\| \leq \varepsilon_M$, for $i = 1, 2$, for a positive constant ε_M . The NN model for the system is constructed as

$$\hat{x}_{k+1} = \hat{W}_{m1k}^\top \sigma_1(z_k),$$

$$\hat{y}_k = \hat{W}_{m2k}^\top \sigma_2(z_k), \quad (13.2.8)$$

where \hat{x}_k is the estimated system state vector and \hat{y}_k is the estimated system output vector. Let \hat{W}_{m1k} be the estimation of the ideal weight matrix W_{m1}^* and let \hat{W}_{m2k} be the estimation of the ideal weight matrix W_{m2}^* . Then, we define the system identification errors as

$$\tilde{x}_{k+1} = \hat{x}_{k+1} - x_{k+1} = \tilde{W}_{m1k}^\top \sigma_1(z_k) - \varepsilon_{m1k},$$

$$\tilde{y}_k = \hat{y}_k - y_k = \tilde{W}_{m2k}^\top \sigma_2(z_k) - \varepsilon_{m2k}, \quad (13.2.9)$$

where $\tilde{W}_{m1k} = \hat{W}_{m1k} - W_{m1}^*$ and $\tilde{W}_{m2k} = \hat{W}_{m2k} - W_{m2}^*$.

Let $\phi_{m1k} = \tilde{W}_{m1k}^\top \sigma_1(z_k)$ and $\phi_{m2k} = \tilde{W}_{m2k}^\top \sigma_2(z_k)$. Then, we can get

$$\tilde{x}_{k+1} = \phi_{m1k} - \varepsilon_{m1k},$$

$$\tilde{y}_k = \phi_{m2k} - \varepsilon_{m2k}.$$

The weights are adjusted to minimize the following error

$$E_{mk} = \frac{1}{2} \tilde{x}_{k+1}^2 + \frac{1}{2} \tilde{y}_k^\top \tilde{y}_k.$$

By a gradient-based adaptation rule, the weights are updated as

$$\hat{W}_{m1,k+1} = \hat{W}_{m1k} - l_{m1} \sigma_1(z_k) \tilde{x}_{k+1},$$

$$\hat{W}_{m2,k+1} = \hat{W}_{m2k} - l_{m2} \sigma_2(z_k) \tilde{y}_k^\top, \quad (13.2.10)$$

where $l_{m1} > 0$ and $l_{m2} > 0$ are learning rates.

Before proceeding, the following assumption is necessary.

Assumption 13.2.1 The NN approximation errors ε_{m1k} and ε_{m2k} are assumed to be upper bounded by a function of estimation error such that

$$\begin{aligned}\varepsilon_{m1k}^2 &\leq \lambda_{m1}\tilde{x}^2, \\ \phi_{m2k}^\top \varepsilon_{m2k} &\leq \lambda_{m2}\phi_{m2k}^\top \phi_{m2k},\end{aligned}$$

where $0 < \lambda_{m1} < 1$ and $0 < \lambda_{m2} < 1$ are bounded constant values.

Then, we have the following theorem.

Theorem 13.2.1 *Let the identification scheme (13.2.8) be used to identify the nonlinear system (13.2.6), and let the NN weights be updated by (13.2.10). If Assumption 13.2.1 holds, then the system identification error \tilde{x}_k approaches zero asymptotically and the error matrices \tilde{W}_{m1k} and \tilde{W}_{m2k} both converge to zero, as $k \rightarrow \infty$.*

Proof Consider the following Lyapunov function candidate defined as

$$L(\tilde{x}_k, \tilde{W}_{m1k}, \tilde{W}_{m2k}) = \tilde{x}_k^2 + \frac{1}{l_{m1}} \text{tr}\{\tilde{W}_{m1k}^\top \tilde{W}_{m1k}\} + \frac{1}{l_{m2}} \text{tr}\{\tilde{W}_{m2k}^\top \tilde{W}_{m2k}\}.$$

The difference of the Lyapunov function candidate is given by

$$\begin{aligned}\Delta L(\tilde{x}_k, \tilde{W}_{m1k}, \tilde{W}_{m2k}) &= \tilde{x}_{k+1}^2 - \tilde{x}_k^2 \\ &\quad + \frac{1}{l_{m1}} \text{tr}\{\tilde{W}_{m1,k+1}^\top \tilde{W}_{m1,k+1} - \tilde{W}_{m1k}^\top \tilde{W}_{m1k}\} \\ &\quad + \frac{1}{l_{m2}} \text{tr}\{\tilde{W}_{m2,k+1}^\top \tilde{W}_{m2,k+1} - \tilde{W}_{m2k}^\top \tilde{W}_{m2k}\}.\end{aligned}$$

With the identification error dynamics (13.2.9) and the weight tuning rules of $\hat{W}_{m1,k+1}$ and $\hat{W}_{m2,k+1}$ in (13.2.10), we can obtain

$$\begin{aligned}\Delta L(\tilde{x}_k, \tilde{W}_{m1k}, \tilde{W}_{m2k}) &= \phi_{m1k}^2 - 2\phi_{m1k}\varepsilon_{m1k} + \varepsilon_{m1k}^2 + l_{m1}\sigma_1^\top(z_k)\sigma_1(z_k)\tilde{x}_{k+1}^2 \\ &\quad - \tilde{x}_k^2 + l_{m2}\sigma_2^\top(z_k)\sigma_2(z_k)\tilde{y}_k^\top \tilde{y}_k - 2\tilde{W}_{m2k}\sigma_2^\top(z_k)\tilde{y}_k - 2\phi_{m1k}\tilde{x}_{k+1}.\end{aligned}$$

Applying the Cauchy–Schwarz inequality, we get

$$\begin{aligned}\Delta L(\tilde{x}_k, \tilde{W}_{m1k}, \tilde{W}_{m2k}) &\leq \phi_{m1k}^2 + \varepsilon_{m1k}^2 - \tilde{x}_k^2 + 2l_{m1}\sigma_1^\top(z_k)\sigma_1(z_k) \\ &\quad \times (\phi_{m1k}^2 + \varepsilon_{m1k}^2) - 2(\phi_{m2k}^\top \phi_{m2k} - \phi_{m2k}^\top \varepsilon_{m2k}) \\ &\quad + l_{m2}\sigma_2^\top(z_k)\sigma_2(z_k)(\phi_{m2k} - \varepsilon_{m2k})^\top(\phi_{m2k} - \varepsilon_{m2k}).\end{aligned}$$

Considering $\max\{\|\sigma_1(z_k)\|, \|\sigma_2(z_k)\|\} \leq \sigma_M$, we can get

$$\begin{aligned}\Delta L(\tilde{x}_k, \tilde{W}_{m1k}, \tilde{W}_{m2k}) &\leq -(1 - 2l_{m1}\sigma_M^2)\phi_{m1k}^2 - (1 - \lambda_{m1} - 2l_{m1}\lambda_{m1}\sigma_M^2)\tilde{x}_k^2 \\ &\quad - 2(\|\phi_{m2k}\|^2 - \phi_{m2k}^\top \varepsilon_{m2k}) + l_{m2}\sigma_M^2\|\phi_{m2k} - \varepsilon_{m2k}\|^2.\end{aligned}\quad (13.2.11)$$

As ε_{m2k} is finite, there exists a $\chi_{m2} > 0$ such that

$$\varepsilon_{m2k}^\top \varepsilon_{m2k} \leq \chi_{m2}\phi_{m2k}^\top \phi_{m2k}.$$

Then, (13.2.11) can be written as

$$\begin{aligned}\Delta L(\tilde{x}_k, \tilde{W}_{m1k}, \tilde{W}_{m2k}) &\leq -(1 - 2l_{m1}\sigma_M^2)\phi_{m1k}^2 - (1 - \lambda_{m1} - 2l_{m1}\lambda_{m1}\sigma_M^2)\tilde{x}_k^2 \\ &\quad - 2(1 - \lambda_{m2})\|\phi_{m2k}\|^2 + 2l_{m2}\sigma_M^2(1 + \chi_{m2})\|\phi_{m2k}\|^2.\end{aligned}$$

Let l_{m1} be selected as

$$l_{m1} < \min \left\{ \frac{1}{2\sigma_M^2}, \frac{1 - \lambda_{m1}}{2\lambda_{m1}\sigma_M^2} \right\}$$

and l_{m2} be selected as

$$l_{m2} < \frac{1 - \lambda_{m2}}{\sigma_M^2(1 + \chi_{m2})}.$$

Then, $\Delta L < 0$. The proof is complete.

Next, NN will be used to identify the coal quality function Θ_k and solve the reference control law u_{fk} using the system data. Different from the system modeling, the coal quality data cannot generally be detected and identified in real-time coal gasification process. This means that the coal quality data can only be obtained offline. Noticing this feature, an iterative training method of the neural networks can be adopted.

According to (13.2.6), we can solve Θ_k , which is expressed as

$$\Theta_k = F_\Theta(x_k, x_{k+1}, y_k, u_k). \quad (13.2.12)$$

Usually, $F_\Theta(\cdot)$ is a highly nonlinear function and the analytical expression of $F_\Theta(\cdot)$ is nearly impossible to obtain. Thus, a BP NN (Θ network for brief) is established to identify the coal quality function Θ_k .

Let the number of hidden layer neurons be denoted by L_Θ . Let the ideal weights be denoted by W_Θ^* . The NN representation of (13.2.12) can be written as

$$\Theta_k = W_\Theta^{*\top} \sigma_\Theta(z_{\Theta k}) + \varepsilon_{\Theta k}, \quad (13.2.13)$$

where $z_{\Theta k} = [x_k, x_{k+1}, y_k^T, u_k^T]^T$ and $\varepsilon_{\Theta k}$ is the reconstruction error. Let $\sigma_{\Theta}(z_{\Theta k}) = \bar{\sigma}(Y_{\Theta} z_{\Theta k})$ where Y_{Θ} is an arbitrary matrix. The NN coal quality function is constructed as

$$\hat{\Theta}_k = \hat{W}_{\Theta k}^T \sigma_{\Theta}(z_{\Theta k}), \quad (13.2.14)$$

where $\hat{\Theta}_k$ is the estimated coal quality function, and $\hat{W}_{\Theta k}$ is the estimated weight matrix. According to (13.2.12), we notice that solving Θ_k needs the data x_{k+1} . As we adopt offline data to train the NN, the corresponding data can be obtained. Define the identification error as

$$\tilde{\Theta}_k^j = \Theta_k - \hat{\Theta}_k^j = \phi_{\Theta k}^j - \varepsilon_{\Theta k},$$

where $\phi_{\Theta k}^j = \tilde{W}_{\Theta k}^{jT} \sigma_{\Theta}(z_{\Theta k})$ and $\tilde{W}_{\Theta k}^j = \hat{W}_{\Theta k}^j - W_{\Theta}^*$. Similarly, the weights are updated as

$$\hat{W}_{\Theta k}^{j+1} = \hat{W}_{\Theta k}^j - l_{\Theta} \sigma_{\Theta}(z_{\Theta k}) \tilde{\Theta}_k^j, \quad (13.2.15)$$

where $l_{\Theta} > 0$ is the learning rate.

Next, we will solve the reference control using NN (u_f network for brief). In this chapter, as we aim to design a state-feedback controller to guarantee the system state to track the desired one, according to the state equation in (13.2.6), we utilize x_k, x_{k+1}, Θ_k to approximate the reference control function u_{fk} , which is expressed as

$$u_{fk} = F_u(x_k, x_{k+1}, \Theta_k). \quad (13.2.16)$$

Let the number of hidden layer neurons be denoted by L_u . Let the ideal weights be denoted by W_u^* . The NN representation of (13.2.16) can be written as

$$u_{fk} = W_u^{*T} \sigma_u(z_{uk}) + \varepsilon_{uk}, \quad (13.2.17)$$

where $z_{uk} = [x_k, x_{k+1}, \Theta_k^T]^T$ and ε_{uk} is the reconstruction error. Let $\sigma_u(z_k) = \bar{\sigma}(Y_u z_k)$ where Y_u is an arbitrary matrix. The NN reference control is constructed as

$$\hat{u}_{fk} = \hat{F}_u(x_k, x_{k+1}, \Theta_k) = \hat{W}_{uk}^T \sigma_u(z_{uk}), \quad (13.2.18)$$

where \hat{u}_{fk} is the estimated reference control, and \hat{W}_{uk}^T is the estimated weight matrix.

Define the identification error as

$$\tilde{u}_{fk}^j = u_{fk} - \hat{u}_{fk}^j = \phi_{uk}^j - \varepsilon_{uk}, \quad (13.2.19)$$

where $\phi_{uk}^j = \tilde{W}_{uk}^{jT} \sigma_u(z_{uk})$ and $\tilde{W}_{uk}^j = \hat{W}_{uk}^j - W_u^*$.

Similarly, the weights are updated as

$$\hat{W}_{uk}^{j+1} = \hat{W}_{uk}^j - l_u \sigma_u(z_{uk}) \tilde{u}_k^j, \quad (13.2.20)$$

where $l_u > 0$ is the learning rate.

Next, we give the convergence properties of Θ network and u_f network.

Theorem 13.2.2 *Let the identification schemes (13.2.14) and (13.2.18) be used to identify Θ_k and u_{fk} in (13.2.12) and (13.2.17), respectively. Let the NN weights be updated by (13.2.15) and (13.2.20), respectively. If for $j = 1, 2, \dots$, the inequalities*

$$\begin{aligned} \phi_{\Theta k}^{j\top} \varepsilon_{\Theta k} &\leq \lambda_{\Theta} \phi_{\Theta k}^{j\top} \phi_{\Theta k}^j, \\ \phi_{uk}^{j\top} \varepsilon_{uk} &\leq \lambda_u \phi_{uk}^{j\top} \phi_{uk}^j \end{aligned} \quad (13.2.21)$$

hold, where $0 < \lambda_{\Theta} < 1$ and $0 < \lambda_u < 1$, then the error matrices $\tilde{W}_{\Theta k}$ and \tilde{W}_{uk} both converge to zero, as $j \rightarrow \infty$.

Proof Consider the following Lyapunov function candidate

$$L(\tilde{W}_{\Theta k}^j, \tilde{W}_{uk}^j) = \frac{1}{l_{\Theta}} \text{tr}\{\tilde{W}_{\Theta k}^{j\top} \tilde{W}_{\Theta k}^j\} + \frac{1}{l_u} \text{tr}\{\tilde{W}_{uk}^{j\top} \tilde{W}_{uk}^j\}.$$

As the activation functions $\sigma_{\Theta}(z_{\Theta k})$ and $\sigma_u(z_{uk})$ are both bounded. We can let $\|\sigma_{\Theta}(z_{\Theta k})\| \leq \sigma_{\bar{\Theta}}$ and $\|\sigma_u(z_{uk})\| \leq \sigma_{\bar{u}}$, respectively. The difference of the Lyapunov function candidate is given by

$$\begin{aligned} \Delta L(\tilde{W}_{\Theta k}^j, \tilde{W}_{uk}^j) &= \frac{1}{l_{\Theta}} \text{tr}\{\tilde{W}_{\Theta k}^{(j+1)\top} \tilde{W}_{\Theta k}^{j+1} - \tilde{W}_{\Theta k}^{j\top} \tilde{W}_{\Theta k}^j\} \\ &\quad + \frac{1}{l_u} \text{tr}\{\tilde{W}_{uk}^{(j+1)\top} \tilde{W}_{uk}^{j+1} - \tilde{W}_{uk}^{j\top} \tilde{W}_{uk}^j\} \\ &= -2W_{\Theta k}^{j\top} \sigma_{\Theta k} \tilde{\Theta}_k^j + l_{\Theta} \tilde{\Theta}_k^{j\top} \sigma_{\Theta k}^{\top} \sigma_{\Theta k} \tilde{\Theta}_k^j \\ &\quad - 2W_{uk}^{j\top} \sigma_{uk} \tilde{u}_k^j + l_u \tilde{u}_k^{j\top} \sigma_{uk}^{\top} \sigma_{uk} \tilde{u}_k^j \\ &\leq -2(\phi_{\Theta k}^{j\top} \phi_{\Theta u}^j - \phi_{\Theta k}^{j\top} \varepsilon_{\Theta k}) + l_{\Theta} \sigma_{\bar{\Theta}}^2 \|\phi_{\Theta k}^j - \varepsilon_{\Theta k}\|^2 \\ &\quad - 2(\phi_{uk}^{j\top} \phi_{uk}^j - \phi_{uk}^{j\top} \varepsilon_{uk}) + l_u \sigma_{\bar{u}}^2 \|\phi_{uk}^j - \varepsilon_{uk}\|^2, \end{aligned} \quad (13.2.22)$$

where $\sigma_{\Theta k} = \sigma_{\Theta}(z_{\Theta k})$ and $\sigma_{uk} = \sigma_u(z_{uk})$. As $\varepsilon_{\Theta k}$ and ε_{uk} are bounded, there exist $\chi_{\Theta} > 0$ and $\chi_u > 0$ such that

$$\begin{aligned} \varepsilon_{\Theta k}^{\top} \varepsilon_{\Theta k} &\leq \chi_{\Theta} \phi_{\Theta k}^{j\top} \phi_{\Theta k}^j, \\ \varepsilon_{uk}^{\top} \varepsilon_{uk} &\leq \chi_u \phi_{uk}^{j\top} \phi_{uk}^j. \end{aligned}$$

Then, (13.2.22) can be written as

$$\begin{aligned}\Delta L(\tilde{W}_{\Theta k}^j, \tilde{W}_{u k}^j) \leq & -2(1-\lambda_{\Theta})\phi_{\Theta k}^{j\top}\phi_{\Theta k}^j + 2l_{\Theta}\sigma_{\Theta}^2(1+\chi_{\Theta})\|\phi_{\Theta k}^j\|^2 \\ & -2(1-\lambda_u)\phi_{u k}^{j\top}\phi_{u k}^j + 2l_u\sigma_u^2(1+\chi_u)\|\phi_{u k}^j\|^2.\end{aligned}$$

According to (13.2.21), we can select the learning rates l_{Θ} and l_u as

$$\begin{aligned}l_{\Theta} &< \frac{1-\lambda_{\Theta}}{\sigma_{\Theta}^2(1+\chi_{\Theta})}, \\ l_u &< \frac{1-\lambda_u}{\sigma_u^2(1+\chi_u)}.\end{aligned}$$

Hence, we can obtain $\Delta L(\tilde{W}_{\Theta k}^j, \tilde{W}_{u k}^j) \leq 0$. The proof is complete.

Remark 13.2.1 From Theorem 13.2.2, the coal quality function Θ_k and the reference control law u_{fk} can be approximated by neural networks. It should be pointed out that, in real-world applications, the coal quality is generally a slow time-varying function. It implies that when the coal quality function Θ_k is identified, it can be considered as a constant vector. Hence, from the current coal gasification system, we can obtain the current state, input and output data. Then, we can first use neural network to identify Θ_k according to (13.2.12)–(13.2.15). Then, taking the estimated $\hat{\Theta}_k$ and the state x_k into the u_f network, we can obtain the reference control law \hat{u}_{fk} immediately, according to (13.2.16)–(13.2.20).

13.3 Design and Implementation of Optimal Tracking Control

In the previous section, we have shown how to use the system data to approximate the dynamics of system (13.2.6). NNs are also adopted to solve the reference control and obtain the coal quality function, respectively. In this section, we will present the iterative ADP algorithm to obtain the optimal tracking control law under system and iteration errors.

13.3.1 Optimal Tracking Controller Design by Iterative ADP Algorithm Under System and Iteration Errors

Although the control system, the reference control, and the coal quality function are approximated by NNs, the system errors are still unknown. It is difficult to design the optimal tracking control system with unknown system errors. Thus, an effective system transformation is performed in this section.

In order to transform the system, for the desired system state η , a desired reference control (desired control for brief) can be obtained. Substituting the desired state trajectory η into (13.2.16), we can obtain the reference control trajectory

$$u_{dk} = F_u(\eta, \eta, \Theta_k),$$

where u_{dk} is defined as the desired control or reference control. Let $\hat{F}_u(\eta, \eta, \Theta_k) = \hat{W}_{uk}^\top \sigma_u(\eta, \eta, \Theta_k)$ be the neural network function which approximates the reference control u_{dk} . If the weights \hat{W}_{uk} converge to W_u^* sufficiently, then

$$u_{dk} = \hat{F}_u(\eta, \eta, \Theta_k) + \varepsilon_{uk}.$$

As Θ_k cannot be obtained directly, Θ network is used to approximate Θ_k . According to (13.2.13) and (13.2.14), letting the weights \hat{W}_{Θ_k} be convergent to W_Θ^* sufficiently, we have

$$\Theta_k = \hat{\Theta}_k + \varepsilon_{\Theta k}.$$

Let $z_{duk} = [\eta, \eta, \Theta_k^\top]^\top$ and $\hat{z}_{duk} = [\eta, \eta, \hat{\Theta}_k^\top]^\top$. According to the mean value theorem, we have

$$\hat{F}_u(\hat{z}_{duk}) = \hat{F}_u(z_{duk}) - \nabla(\xi_\Theta) \varepsilon_{\Theta k},$$

where

$$\nabla(\xi_\Theta) = \frac{\partial \hat{F}_u(\eta, \eta, \xi_\Theta)}{\partial \xi_\Theta},$$

$\xi_\Theta = c_\Theta \hat{\Theta}_k + (1 - c_\Theta)(\hat{\Theta}_k - \varepsilon_{\Theta k})$ and $0 \leq c_\Theta \leq 1$ is some constant. As $\varepsilon_{\Theta k}$ is bounded and $\sigma_u(\cdot)$ is smooth, $\|\nabla(\xi_\Theta) \varepsilon_{\Theta k}\|$ is bounded. If we let the neural network-generated reference control be expressed by

$$\hat{u}_{dk} = \hat{F}_u(\eta, \eta, \hat{\Theta}_k), \quad (13.3.1)$$

then we can get

$$u_{dk} = \hat{u}_{dk} + \hat{\varepsilon}_{uk}, \quad (13.3.2)$$

where $\hat{\varepsilon}_{uk} = \nabla(\xi_\Theta) \varepsilon_{\Theta k} + \varepsilon_{uk}$. Let $u_{\delta k}$ be the error between the control u_k and the reference control u_{dk} , then we can obtain

$$u_k = \hat{u}_{dk} + u_{\delta k} + \hat{\varepsilon}_{uk} = \hat{u}_k + \hat{\varepsilon}_{uk}, \quad (13.3.3)$$

where $\hat{u}_k = \hat{u}_{dk} + u_{\delta k}$ is the estimated control.

Remark 13.3.1 From (13.3.3), we can see that if we have obtained the estimated control \hat{u}_k and the control error $\hat{\varepsilon}_{uk}$, then the control input u_k can be determined.

In the real-world industrial processes, however, the control performance will be influenced by lower level controllers. For coal gasification, the fluctuation of flow of the control inputs is important and cannot be ignored. Let $\Delta\varepsilon_{uk}$ be the bounded disturbance on the control signal. The control input can be written as

$$\bar{u}_k = \hat{u}_k + \Delta\varepsilon_{uk} = u_k + \bar{\varepsilon}_{uk}, \quad (13.3.4)$$

where $\bar{\varepsilon}_{uk} = \Delta\varepsilon_{uk} - \hat{\varepsilon}_{uk}$. Next, the disturbance of the control is considered.

According to (13.2.8), let the NN weights be convergent to W_{m1}^* and W_{m2}^* sufficiently. If we let $\hat{F}(z_k) = \hat{W}_{m1k}^\top \sigma_1(z_k)$, then the system state equation can be written as

$$x_{k+1} = \hat{F}(z_k) + \varepsilon_{m1k}. \quad (13.3.5)$$

As u_k and Θ_k cannot be obtained directly, approximations are adopted. Let $\hat{z}_k = [x_k, \bar{u}_k, \hat{\Theta}_k^\top]^\top$. As the activation function $\sigma_1(\cdot)$ is smooth, according to the mean value theorem, we have

$$\hat{F}(\hat{z}_k) = \hat{F}(z_k) + \nabla(\xi_u) \bar{\varepsilon}_{uk} + \nabla(\xi_\Theta) \varepsilon_{\Theta k},$$

where

$$\nabla(\xi_u) = \frac{\partial \hat{F}(x_k, \xi_{uk}, \hat{\Theta}_k)}{\partial \xi_{uk}},$$

$\xi_u = c_u \bar{u}_k + (1 - c_u)(\bar{u}_k - \bar{\varepsilon}_{uk})$, and $0 \leq c_u \leq 1$ is some constant. Let $\nabla(\xi_\Theta) = \partial \hat{F}(x_k, \bar{u}_k, \xi_\Theta) / \partial \xi_\Theta$, and $\xi'_\Theta = c'_\Theta \hat{\Theta}_k + (1 - c'_\Theta)(\hat{\Theta}_k - \varepsilon_{\Theta k})$, where $0 \leq c'_\Theta \leq 1$ is some constant. As $\bar{\varepsilon}_{uk}$ and $\varepsilon'_{\Theta k}$ are both bounded and $\sigma_u(\cdot)$ and $\sigma_\Theta(\cdot)$ are both smooth, then $\|\nabla(\xi_u) \bar{\varepsilon}_{uk}\|$ and $\|\nabla(\xi'_\Theta) \varepsilon'_{\Theta k}\|$ are both bounded. So, we let $\|\nabla(\xi_u) \bar{\varepsilon}_{uk}\| \leq \|\bar{\varepsilon}_u\|$ and $\|\nabla(\xi'_\Theta) \varepsilon'_{\Theta k}\| \leq \|\bar{\varepsilon}_\Theta\|$. Then, (13.3.5) can be written as

$$x_{k+1} = \hat{F}(x_k, \bar{u}_k, \hat{\Theta}_k) + \nabla(\xi_u) \bar{\varepsilon}_{uk} + \nabla(\xi'_\Theta) \varepsilon'_{\Theta k} + \varepsilon_{m1k}. \quad (13.3.6)$$

Let the tracking error be defined as

$$e_k = x_k - \eta, \quad (13.3.7)$$

where η is the desired state trajectory. Let

$$u_{ek} = u_k - u_{dk}, \quad (13.3.8)$$

where \hat{u}_{dk} is the neural network-generated reference control trajectory expressed by (13.3.1). According to (13.3.2), (13.3.4), and (13.3.8), we can get

$$\bar{u}_k = u_{ek} + \hat{u}_{dk} + \bar{\varepsilon}_{uk}, \quad (13.3.9)$$

where $\tilde{\varepsilon}_{uk} = \bar{\varepsilon}_{uk} + \varepsilon_{uk}$. According to (13.3.7) and (13.3.9), we have

$$\hat{F}(x_k, \bar{u}_k, \hat{\Theta}_k) = \hat{F}((e_k + \eta), (u_{ek} + \hat{u}_{dk}), \hat{\Theta}_k) + \nabla(\tilde{\xi}_u)\tilde{\varepsilon}_{uk},$$

where

$$\nabla(\tilde{\xi}_u) = \partial \hat{F}((e_k + \eta), \tilde{\xi}_u, \hat{\Theta}_k) / \partial \tilde{\xi}_u,$$

$$\tilde{\xi}_u = \tilde{c}_u(u_{ek} + \hat{u}_{dk}) + (1 - \tilde{c}_u)(u_{ek} + \hat{u}_{dk} + (\bar{\varepsilon}_{uk} + \varepsilon_{uk})),$$

and $0 \leq \tilde{c}_u \leq 1$. Thus, (13.3.6) can be written as

$$e_{k+1} = \hat{F}((e_k + \eta), (u_{ek} + \hat{u}_{dk}), \hat{\Theta}_k) - \eta + w_k, \quad (13.3.10)$$

where $w_k = \nabla(\tilde{\xi}_u)\tilde{\varepsilon}_{uk} + \nabla(\xi'_\Theta)\varepsilon'_{\Theta k} + \varepsilon_{m1k} + \nabla(\tilde{\xi}_u)\tilde{\varepsilon}_{uk}$. As $\nabla(\tilde{\xi}_u)\tilde{\varepsilon}_{uk}$, $\nabla(\xi'_\Theta)\varepsilon'_{\Theta k}$, $\nabla(\tilde{\xi}_u)\tilde{\varepsilon}_{uk}$, and ε_{m1k} are all bounded, the system disturbance will also be bounded. Let $|\varepsilon_{m1k}| \leq |\bar{\varepsilon}_{m1}|$ and $\|\nabla(\tilde{\xi}_u)\tilde{\varepsilon}_{uk}\| \leq \|\tilde{\varepsilon}_u\|$, then we can get

$$\|w_k\| \leq \|\bar{\varepsilon}_u\| + \|\bar{\varepsilon}_\Theta\| + |\bar{\varepsilon}_{m1}| + \|\tilde{\varepsilon}_u\|.$$

On the other hand, as mentioned in Remark 13.2.1, $\hat{\Theta}_k$ is a constant vector after it is identified. Hence, according to (13.3.1), \hat{u}_{dk} can also be seen as a constant vector. Then, system (13.3.10) can be transformed into the following regulation system

$$e_{k+1} = \bar{F}(e_k, u_{ek}, \hat{\Theta}_k) + w_k, \quad (13.3.11)$$

where

$$\bar{F}(e_k, u_{ek}, \hat{\Theta}_k) = \hat{F}((e_k + \eta), (u_{ek} + \hat{u}_{dk}), \hat{\Theta}_k) - \eta.$$

From (13.3.11), we can see that the nonlinear tracking control system (13.2.6) is transformed into a regulation system, where the system errors and the control fluctuation are transformed into an unknown-bounded system disturbance.

Our goal is to obtain an optimal control such that the tracking error e_k converges to zero under the system disturbance w_k . As the system disturbance w_k is unknown, the design of the optimal controller becomes very difficult. In [2], the optimal control problem for system (13.3.11) was transformed into a two-person zero-sum optimal control problem, where the system disturbance w_k was defined as a control variable. The optimal control law is obtained under the worst case of the disturbance (the disturbance control maximizes the cost function). Inspired by [2], we define w_k as a disturbance control of the system and the two controls u_{ek} and w_k of system (13.3.11) are designed to optimize the following quadratic cost function

$$J(e_0, \underline{u}_{e0}, \underline{w}_0) = \sum_{k=0}^{\infty} (Ae_k^2 + u_{ek}^\top Bu_{ek} - Cw_k^2), \quad (13.3.12)$$

where we let $\underline{u}_{ek} = (u_{ek}, u_{e,k+1}, \dots)$ and $\underline{w}_k = (w_k, w_{k+1}, \dots)$. Let $A > 0$ and $C > 0$ be constants, and let B be a positive definite matrix. Then, the optimal cost function can be defined as

$$J^*(e_k) = \inf_{\underline{u}_{ek}} \sup_{\underline{w}_k} \{ J(e_k, \underline{u}_{ek}, \underline{w}_k) \}.$$

Let $U(e_k, u_{ek}, w_k) = Ae_k^2 + u_{ek}^\top Bu_{ek} - Cw_k^2$ be the utility function. In this chapter, we assume that the utility function $U(e_k, u_{ek}, w_k) > 0, \forall e_k, u_{ek}, w_k \neq 0$. Generally speaking, the system errors are small. This requires the system disturbance w_k to be small and the utility function to be larger than zero. If w_k are large, we can reduce the value of C or enlarge value of A and the matrix B . Hence, the assumption $U(e_k, u_{ek}, w_k) > 0$ can be guaranteed.

According to Bellman's principle of optimality, $J^*(e_k)$ satisfies the discrete-time Hamilton–Jacobi–Isaacs (HJI) equation

$$J^*(e_k) = \min_{u_{ek}} \max_{w_k} \{ U(e_k, u_{ek}, w_k) + J^*(e_{k+1}) \}. \quad (13.3.13)$$

Define the laws of optimal controls as

$$\begin{aligned} w^*(e_k) &= \arg \max_{w_k} \{ U(e_k, u_{ek}, w_k) + J^*(e_{k+1}) \}, \\ u_e^*(e_k) &= \arg \min_{u_{ek}} \{ U(e_k, u_{ek}, w^*(e_k)) + J^*(e_{k+1}) \}. \end{aligned}$$

Hence, the HJI equation (13.3.13) can be written as

$$J^*(e_k) = U(e_k, u_e^*(e_k), w^*(e_k)) + J^*(e_{k+1}).$$

We can see that if we want to obtain the optimal control laws $u_e^*(e_k)$ and $w^*(e_k)$, we must obtain the optimal cost function $J^*(e_k)$. Generally speaking, $J^*(e_k)$ is unknown before all the controls u_{ek} and w_k are considered, which means that the HJI equation is generally unsolvable. In this chapter, an iterative ADP algorithm with system and approximation errors is employed to overcome these difficulties. In the present iterative ADP algorithm, the cost function and control law are updated by iterations, with the iteration index i increasing from 0 to infinity. Let the initial value function $\hat{V}_0(e_k) \equiv 0, \forall e_k$.

From $\hat{V}_0(\cdot) = 0$, we calculate

$$\omega_0(e_k) = \arg \max_{w_k} \{ U(e_k, u_{ek}, w_k) + \hat{V}_0(e_{k+1}) \}, \quad (13.3.14)$$

$$\hat{v}_0(e_k) = \arg \min_{u_{ek}} \{ U(e_k, u_{ek}, \omega_0(e_k)) + \hat{V}_0(e_{k+1}) \} + \rho_0(e_k), \quad (13.3.15)$$

where $\hat{V}_0(e_{k+1}) = 0$ and $\rho_0(e_k)$ is the finite approximation error function of the iterative control $\hat{v}_0(e_k)$. For $i = 1, 2, \dots$, the iterative algorithm calculates the iterative value function $\hat{V}_i(e_k)$,

$$\begin{aligned}\hat{V}_i(e_k) &= \min_{u_{ek}} \max_{w_k} \left\{ U(e_k, u_{ek}, w_k) + \hat{V}_{i-1}(e_{k+1}) \right\} + \pi_i(e_k) \\ &= U(e_k, \hat{v}_{i-1}(e_k), \omega_{i-1}(e_k) + \hat{V}_{i-1}(e_{k+1}) + \pi_i(e_k)),\end{aligned}\quad (13.3.16)$$

where e_{k+1} is expressed as in (13.3.11) and $\pi_i(e_k)$ is the finite approximation error function of the iterative value function and the control laws $\omega_i(e_k)$ and $\hat{v}_i(e_k)$,

$$\omega_i(e_k) = \arg \max_{w_k} \left\{ U(e_k, u_{ek}, w_k) + \hat{V}_i(e_{k+1}) \right\}, \quad (13.3.17)$$

$$\hat{v}_i(e_k) = \arg \min_{u_{ek}} \left\{ U(e_k, u_{ek}, \omega_i(e_k)) + \hat{V}_i(e_{k+1}) \right\} + \rho_i(e_k), \quad (13.3.18)$$

where $\rho_i(e_k)$ is the finite approximation error function of the iterative control.

Remark 13.3.2 From (13.3.11), the system is affine for the disturbance control w_k (it is actually linear in this case). According to (13.3.14) and (13.3.17), using the necessary condition of optimality, for $i = 0, 1, 2, \dots$, $\omega_i(e_k)$ can be obtained as

$$\omega_i(e_k) = \frac{1}{2} C^{-1} \frac{d\hat{V}_i(e_{k+1})}{de_{k+1}}.$$

Next, we consider the properties of the iterative ADP algorithm with system errors, iteration errors, and control disturbance. For the two-person zero-sum iterative ADP algorithm described in (13.3.14)–(13.3.18), as the iteration errors are unknown, the properties of the iterative value functions $\hat{V}_i(e_k)$ and the iterative control laws $\omega_i(e_k)$ and $\hat{v}_i(e_k)$ are very difficult to analyze, for $i = 0, 1, \dots$. On the other hand, in [11], for nonlinear systems with a single controller, an “error bound” analysis method is proposed to prove the convergence of the iterative value function. In this chapter, we will establish similar “error bound” convergence analysis results for the iterative value functions for nonlinear two-person zero-sum optimal control problems.

For $i = 1, 2, \dots$, define an iterative value function as

$$\begin{aligned}V_i(e_k) &= \min_{u_{ek}} \max_{w_k} \left\{ U(e_k, u_{ek}, w_k) + \hat{V}_i(e_{k+1}) \right\} \\ &= U(e_k, v_i(e_k), \omega_i(e_k)) + \hat{V}_i(e_{k+1}),\end{aligned}\quad (13.3.19)$$

where

$$V_0(e_{k+1}) = \hat{V}_0(e_{k+1}) = 0$$

and

$$v_i(e_k) = \arg \min_{u_{ek}} \left\{ U(e_k, u_{ek}, \omega_i(e_k)) + \hat{V}_i(e_{k+1}) \right\}, \quad (13.3.20)$$

is the accurate iterative control law. According to (13.3.18), there exists a finite constant $\tau \geq 1$ such that

$$\hat{V}_i(e_k) \leq \tau V_i(e_k), \quad \forall i = 0, 1, \dots, \quad (13.3.21)$$

hold uniformly. Hence, we have the following theorems.

Theorem 13.3.1 For all $i = 0, 1, \dots$, let $V_i(e_k)$ be expressed as in (13.3.19) and $\hat{V}_i(e_k)$ be expressed as in (13.3.18). Let e_{k+1} be expressed as in (13.3.11). Let $0 < \gamma < \infty$ be a constant such that

$$J^*(e_{k+1}) \leq \gamma U(e_k, u_{ek}, w_k)$$

holds uniformly. If there exists $1 \leq \tau < \infty$ such that (13.3.21) holds uniformly, then

$$\hat{V}_i(e_k) \leq \tau \left(1 + \sum_{j=1}^i \frac{\gamma^j \tau^{j-1} (\tau - 1)}{(\gamma + 1)^j} \right) J^*(e_k), \quad (13.3.22)$$

where we define $\sum_j^i (\cdot) = 0$, $\forall j > i$ and $i, j = 0, 1, \dots$

Proof The theorem can be proved by mathematical induction. First, let $i = 0$. We have $\hat{V}_0(e_k) = 0$. So, the conclusion holds for $i = 0$. Next, for $i = 1$, according to (13.3.19), we have

$$\begin{aligned} V_1(e_k) &= \min_{u_{ek}} \max_{w_k} \left\{ U(e_k, u_{ek}, w_k) + \hat{V}_0(e_{k+1}) \right\} \\ &\leq \min_{u_{ek}} \max_{w_k} \left\{ U(e_k, u_{ek}, w_k) + \tau J^*(e_{k+1}) \right\} \\ &\leq \left(1 + \frac{\gamma(\tau - 1)}{\gamma + 1} \right) \min_{u_{ek}} \max_{w_k} \{ U(e_k, u_{ek}, w_k) + J^*(e_{k+1}) \} \\ &= \left(1 + \frac{\gamma(\tau - 1)}{\gamma + 1} \right) J^*(e_k). \end{aligned}$$

According to (13.3.21), we can obtain

$$\hat{V}_1(e_k) \leq \tau \left(1 + \frac{\gamma(\tau - 1)}{\gamma + 1} \right) J^*(e_k),$$

which shows that (13.3.22) holds for $i = 1$. Assume that (13.3.22) holds for $i = l-1$, where $l = 1, 2, \dots$. Then, for $i = l$, we have

$$\begin{aligned}
V_l(e_k) &= \min_{u_{ek}} \max_{w_k} \left\{ U(e_k, u_{ek}, w_k) + \hat{V}_{l-1}(e_{k+1}) \right\} \\
&\leq \min_{u_{ek}} \max_{w_k} \left\{ \left(1 + \gamma \sum_{j=1}^{l-1} \frac{\gamma^{j-1} \tau^{j-1} (\tau - 1)}{(\gamma + 1)^j} \right) U(e_k, u_{ek}, w_k) \right. \\
&\quad \left. + \left[\tau \left(1 + \sum_{j=1}^{l-1} \frac{\gamma^j \tau^{j-1} (\tau - 1)}{(\gamma + 1)^j} \right) \right. \right. \\
&\quad \left. \left. - \sum_{j=1}^{l-1} \frac{\gamma^{j-1} \tau^{j-1} (\tau - 1)}{(\gamma + 1)^j} \right] J^*(e_{k+1}) \right\} \\
&= \left(1 + \sum_{j=1}^l \frac{\gamma^j \tau^{j-1} (\tau - 1)}{(\gamma + 1)^j} \right) \\
&\quad \times \min_{u_{ek}} \max_{w_k} \left\{ U(e_k, u_{ek}, w_k) + J^*(e_{k+1}) \right\} \\
&= \left(1 + \sum_{j=1}^l \frac{\gamma^j \tau^{j-1} (\tau - 1)}{(\gamma + 1)^j} \right) J^*(e_k).
\end{aligned}$$

According to (13.3.21), we can obtain (13.3.22) which proves the conclusion for $i = 0, 1, \dots$. This completes the proof of the theorem.

Theorem 13.3.2 *Suppose Theorem 13.3.1 holds. If for $0 < \gamma < \infty$, the inequality*

$$1 \leq \tau < \frac{\gamma + 1}{\gamma} \quad (13.3.23)$$

holds, then as $i \rightarrow \infty$, the iterative value function $\hat{V}_i(e_k)$ in the iterative ADP algorithm (13.3.14)–(13.3.18) is uniformly convergent to a bounded neighborhood of the optimal cost function $J^(e_k)$, i.e.,*

$$\lim_{i \rightarrow \infty} \hat{V}_i(e_k) = \hat{V}_\infty(e_k) \leq \tau \left(1 + \frac{\gamma(\tau - 1)}{1 - \gamma(\tau - 1)} \right) J^*(e_k). \quad (13.3.24)$$

Proof According to (13.3.22) in Theorem 13.3.1, we can see that for $j = 1, 2, \dots$, the sequence $\left\{ \frac{\gamma^j \tau^{j-1} (\tau - 1)}{(\gamma + 1)^j} \right\}$ is a geometrical series. If $1 \leq \tau < (\gamma + 1)/\gamma$, then, for $i \rightarrow \infty$, (13.3.22) becomes

$$\lim_{i \rightarrow \infty} V_i(e_k) = V_\infty(e_k) \leq \left(1 + \frac{\gamma(\tau - 1)}{1 - \gamma(\tau - 1)} \right) J^*(e_k). \quad (13.3.25)$$

According to (13.3.21), let $i \rightarrow \infty$, we have

$$\hat{V}_\infty(e_k) \leq \tau V_\infty(e_k). \quad (13.3.26)$$

According to (13.3.25) and (13.3.26), we can obtain (13.3.24). This completes the proof of the theorem.

Corollary 13.3.1 Suppose Theorem 13.3.1 holds. If for $0 < \gamma < \infty$ and $1 \leq \tau < \infty$, the inequality (13.3.23) holds, then the iterative control laws $\omega_i(e_k)$ and $\hat{v}_i(e_k)$ of the iterative ADP algorithm (13.3.14)–(13.3.18) are convergent, i.e.,

$$\begin{cases} \omega_\infty(e_k) = \lim_{i \rightarrow \infty} \omega_i(e_k), \\ \hat{v}_\infty(e_k) = \lim_{i \rightarrow \infty} \hat{v}_i(e_k). \end{cases}$$

13.3.2 Neural Network Implementation

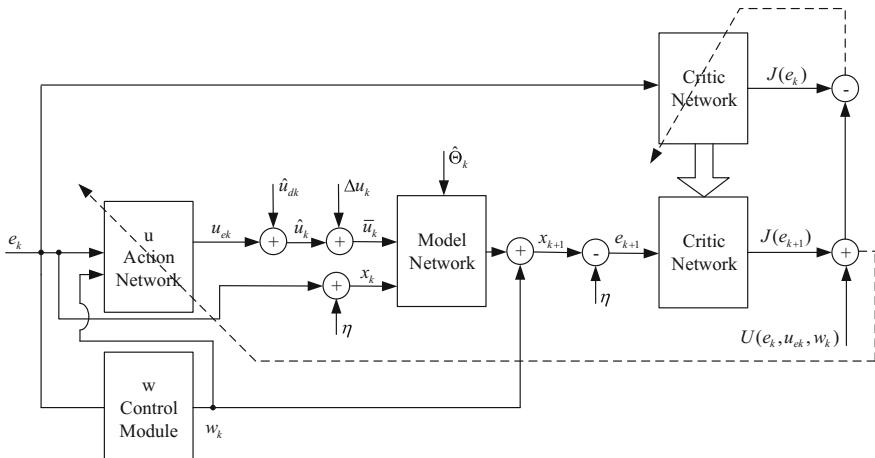
In this section, neural networks, including action network and critic network, are used to implement the present iterative ADP algorithm. Both the neural networks are chosen as three-layer BP networks. The whole structural diagram is shown in Fig. 13.2.

For all $i = 1, 2, \dots$, the critic network is used to approximate the value function $V_i(e_k)$ in (13.3.19). The output of the critic network is denoted by

$$\hat{V}_i^j(e_k) = W_{ci}^{j\top} \sigma_c(e_k)$$

for $j = 0, 1, \dots$. Let W_{ci}^0 be random weight matrices. Let $\sigma_c(e_k) = \bar{\sigma}(Y_c e_k)$ where Y_c is an arbitrary matrix. Then, $\sigma_c(e_k)$ is upper bounded, i.e., $\|\sigma_c(e_k)\| \leq \bar{\sigma}_c$ for a positive constant $\bar{\sigma}_c > 0$. The target function can be written as

$$V_i(e_k) = U(e_k, v_{i-1}(e_k), w_{i-1}(e_k)) + \hat{V}_{i-1}(e_{k+1}).$$



Then, we define the error function of the critic network as

$$e_{cik}^j = V_i(e_k) - \hat{V}_i^j(e_k).$$

The objective function to be minimized in the critic network training is

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

The gradient-based weight update rule [15] can be applied here to train the critic network

$$\begin{aligned} W_{ck}^{i(j+1)} &= W_{ck}^{ij} + \Delta W_{ck}^{ij}, \\ &= W_{ck}^{ij} - l_c \left[\frac{\partial E_{cik}^j}{\partial \hat{V}_i^j(e_k)} \frac{\partial \hat{V}_i^j(e_k)}{\partial W_{ck}^{ij}} \right] \\ &= W_{ck}^{ij} - l_c e_{cik}^j \sigma_c(e_k), \end{aligned} \quad (13.3.27)$$

where $l_c > 0$ is the learning rate of critic network. If the training precision is achieved, then $V_i(e_k)$ can be approximated by the critic network.

The action network is used to approximate the iterative control law $v_i(e_k)$, where $v_i(e_k)$ is defined by (13.3.20). The output can be formulated as

$$\hat{v}_i^j(e_k) = W_a^{ij\top} \sigma_a(e_k).$$

Let $\sigma_a(e_k) = \bar{\sigma}(Y_a e_k)$ where Y_a is an arbitrary matrix. Then, $\sigma_a(e_k)$ is upper bounded, i.e., $\|\sigma_a(e_k)\| \leq \bar{\sigma}_a$ for a positive constant $\bar{\sigma}_a > 0$. So, we can define the output error of the action network as

$$e_{aik}^j = v_i(e_k) - \hat{v}_i^j(e_k).$$

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

$$E_{aik}^j = \frac{1}{2} e_{aik}^{j\top} e_{aik}^j.$$

The weight updating algorithm is similar to the one for the critic network. By the gradient descent rule, we can obtain

$$\begin{aligned} W_{ak}^{i(j+1)} &= W_{ak}^{ij} + \Delta W_{ak}^{ij}, \\ &= W_{ak}^{ij} - l_a \left[\frac{\partial E_{aik}^j}{\partial e_{aik}^j} \frac{\partial e_{aik}^j}{\partial \hat{v}_{ik}^j} \frac{\partial \hat{v}_{ik}^j}{\partial W_{ak}^{ij}} \right] \\ &= W_{ak}^{ij} - l_a e_{aik}^j \sigma_a(e_k), \end{aligned} \quad (13.3.28)$$

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

To guarantee the effectiveness of neural network implementation, the convergence of the neural network weights should be proved which allows the approximation of the iterative value function and iterative control using critic and action networks, respectively. The weight convergence property of the neural networks is shown in the following theorem.

Theorem 13.3.3 *Let the target value function and the target iterative control law be expressed by*

$$V_{i+1}(e_k) = W_c^{*i\top} \sigma_c(e_k) + \varepsilon_{cik},$$

$$v_i(e_k) = W_a^{*i\top} \sigma_a(e_k) + \varepsilon_{aik},$$

respectively, where ε_{cik} and ε_{aik} are reconstruction errors. Let the critic and action networks be trained by (13.3.27) and (13.3.28), respectively. Let $\tilde{W}_c^{ij} = W_c^{ij} - W_c^{*i}$ and $\tilde{W}_a^{ij} = W_a^{ij} - W_a^{*i}$. If for $j = 1, 2, \dots$, there exist $0 < \lambda_c < 1$ and $0 < \lambda_a < 1$ such that

$$\phi_{cik}^j \varepsilon_{cik} \leq \lambda_c (\phi_{cik}^j)^2, \quad \phi_{aik}^{j\top} \varepsilon_{aik} \leq \lambda_a \phi_{aik}^{j\top} \phi_{aik}^j, \quad (13.3.29)$$

where $\phi_{cik}^j = \tilde{W}_{ck}^{ij\top} \sigma_c(e_k)$ and $\phi_{aik}^j = \tilde{W}_{ak}^{ij\top} \sigma_a(e_k)$, then the error matrices \tilde{W}_{ck}^i and \tilde{W}_{ak}^i both converge to zero, as $j \rightarrow \infty$.

Proof From (13.3.27) and (13.3.28), we have

$$\tilde{W}_{ck}^{i(j+1)} = \tilde{W}_c^{ijk} - \alpha_c e_{cik}^j \sigma_c(e_k),$$

$$\tilde{W}_{ak}^{i(j+1)} = \tilde{W}_a^{ijk} - \beta_a e_{aik}^j \sigma_a(e_k).$$

Consider the following Lyapunov function candidate

$$L(\tilde{W}_c^{ij}, \tilde{W}_a^{ij}) = \text{tr}\{\tilde{W}_c^{ij\top} \tilde{W}_c^{ij} + \tilde{W}_a^{ij\top} \tilde{W}_a^{ij}\}. \quad (13.3.30)$$

Let $\tilde{V}_i^j(e_k) = V_i(e_k) - \hat{V}_i^j(e_k)$ and $\tilde{v}_i^j(e_k) = v_i(e_k) - \hat{v}_i^j(e_k)$. Then, the difference of the Lyapunov function candidate (13.3.30) is given by

$$\begin{aligned} \Delta L(\tilde{W}_c^{ij}, \tilde{W}_a^{ij}) &= \frac{1}{l_c} \text{tr}\{\tilde{W}_c^{ij\top} \tilde{W}_c^{ij} + \tilde{W}_a^{ij\top} \tilde{W}_a^{ij}\} - \frac{1}{l_a} \text{tr}\{\tilde{W}_c^{ij\top} \tilde{W}_c^{ij} + \tilde{W}_a^{ij\top} \tilde{W}_a^{ij}\} \\ &= -2W_{ck}^{ij\top} \sigma_c(e_k) \tilde{V}_{i+1}^j(e_k) - 2W_{ak}^{ij\top} \sigma_a(e_k) \tilde{v}_i^j(e_k) + l_c \tilde{V}_{i+1}^j(e_k) \sigma_c^\top(e_k) \\ &\quad \times \sigma_c(e_k) \tilde{V}_{i+1}^j(e_k) + l_\Theta \tilde{v}_i^j(e_k) \sigma_a^\top(e_k) \sigma_a(e_k) \tilde{v}_i^j(e_k) \\ &\leq -2((\phi_{cik}^j)^2 - \phi_{cik}^j \varepsilon_{cik}) - 2(\phi_{aik}^{j\top} \phi_{aik}^j - \phi_{aik}^{j\top} \varepsilon_{aik}) \\ &\quad + l_c \sigma_c^2 (\phi_{cik}^j - \varepsilon_{cik})^2 + l_a \sigma_a^2 \|\phi_{aik}^j - \varepsilon_{aik}\|^2. \end{aligned} \quad (13.3.31)$$

As ε_{cik} and ε_{aik} are both finite, there exist $\chi_c > 0$ and $\chi_a > 0$ such that

$$\varepsilon_{cik}^2 \leq \chi_c (\phi_{cik}^j)^2, \quad \varepsilon_{aik}^T \varepsilon_{aik} \leq \chi_a \phi_{aik}^{jT} \phi_{aik}^j. \quad (13.3.32)$$

Then, (13.3.31) can be written as

$$\begin{aligned} \Delta L(\tilde{W}_c^{ij}, \tilde{W}_a^{ij}) &\leq -2(1 - \lambda_c) (\phi_{cik}^j)^2 + 2l_c \sigma_c^2 (1 + \chi_c) (\phi_{cik}^j)^2 \\ &\quad - 2(1 - \lambda_a) \phi_{aik}^{jT} \phi_{aik}^j + 2l_a \sigma_a^2 (1 + \chi_a) \|\phi_{aik}^j\|^2. \end{aligned}$$

Selecting the learning rates l_c and l_a such that

$$\begin{aligned} l_c &< \frac{1 - \lambda_c}{\sigma_c^2 (1 + \chi_c)}, \\ l_a &< \frac{1 - \lambda_a}{\sigma_a^2 (1 + \chi_a)}, \end{aligned}$$

we can obtain $\Delta L(\tilde{W}_c^{ij}, \tilde{W}_a^{ij}) < 0$. The proof is complete.

13.4 Numerical Analysis

In this section, numerical experiments are studied to show the effectiveness of our iterative ADP algorithm. Let the coal gasification control system be expressed as in (13.2.6). We let the initial reaction temperature in the gasifier be $x_0 = 1000^\circ\text{C}$. Observe the corresponding system input and output data (kg/h) which are

$$u_0 = [60960, 47572, 44752]^T \text{ and } y_0 = [74690, 34381, 4265, 29653, 10295]^T.$$

Let the desired reaction temperature $\eta = 1320^\circ\text{C}$. To model the coal gasification control system (13.2.6), we collect 20,000 temperature data from the real-world coal gasification system. The corresponding 20,000 system input data and 20,000 system output data are also recorded. Then, a three-layer BP NN is established with the structure of 8–20–1 to approximate the state equation in (13.2.6) and the NN is the model network. The control input is expressed by (13.2.4). We also use three-layer BP NN with structure of 8–20–5 to approximate the input–output equation in (13.2.6) and the NN is the input–output network. Let the learning rates of the model network and input–output network be $l_m = 0.002$. Use the gradient-based weight update rule [15] to train the neural networks for 20,000 iteration steps to reach the training precision of 10^{-6} . The converged weights, respectively, are given by

$$\begin{aligned} \hat{W}_{m1} &= [-0.0030, 0.0005, -0.0032, -3.7356, -1.8093, 0.0120, 0.0020, \\ &\quad -0.0030, 4.3394, 0.0001, -0.5405, 0.0200, 0.0080, 0.0310, \\ &\quad -0.0600, -0.014, 0.025, -0.3138, 0.0059, -2.1526] \end{aligned} \quad (13.4.1)$$

and

$$\hat{W}_{m2} = \begin{bmatrix} 0.1062 & 1.0652 & 0.2797 & -0.0631 & -0.1974 \\ -0.0843 & 1.6798 & -0.9755 & 0.0069 & 0.5187 \\ 0.9926 & 0.3194 & -0.3022 & -0.0434 & 0.0509 \\ -0.7054 & 1.4218 & 0.3699 & 0.0479 & 0.0666 \\ 0.0301 & 4.9841 & 0.0175 & 0.1136 & -0.0059 \\ \\ -0.0047 & 0.2946 & 0.0254 & -0.0129 & -0.1453 \\ -0.0232 & -0.9079 & -0.0723 & 0.0366 & 0.0950 \\ -0.0266 & -0.2629 & -0.0252 & 0.0132 & 0.0765 \\ -0.1865 & 0.2426 & 0.0268 & -0.0136 & -0.0804 \\ 0.0179 & -0.0199 & 0.0004 & 0.0001 & -0.0089 \\ \\ 0.1064 & -0.9205 & -0.1178 & -0.3625 & 0.0570 \\ -0.0113 & 2.4362 & 0.5149 & 0.7794 & -0.0446 \\ 0.0751 & -1.2588 & 0.4976 & -0.6136 & 0.0411 \\ -0.0808 & 2.3609 & -0.8929 & 0.3444 & -0.0466 \\ -0.1947 & 9.6033 & 0.0280 & -0.0772 & 0.0162 \\ \\ 0.4106 & 0.0056 & -0.7702 & -2.1900 & 0.0363 \\ -0.7316 & -0.0802 & 2.1927 & 0.9994 & -0.2945 \\ -0.3442 & 0.0189 & 0.1892 & -1.0204 & -0.1601 \\ 0.3453 & -0.0478 & 0.3243 & 1.0497 & -0.4577 \\ 0.0149 & 0.0103 & -0.0156 & 3.6508 & 0.0482 \end{bmatrix}.$$

Next, we adopt three-layer BP NNs to identify the coal quality equation (13.2.13) and the reference control equation (13.2.16). The structure of Θ network and u_f network is chosen as 10–20–4 and 6–20–3, respectively. Using the gradient-based weight update rule, train the two neural networks for 20,000 iteration steps under the learning rate 0.002 to reach the training precision of 10^{-6} . The converged weights are given by

$$\hat{W}_\Theta = \begin{bmatrix} 0.003 & 0.2629 & -0.0030 & 0.0010 & -0.0868 \\ 0.001 & 0.4992 & -0.0001 & 0.0001 & 0.0020 \\ 0.001 & -0.0391 & 0.0010 & -0.0001 & -0.0285 \\ -0.004 & 0.2651 & 0.0020 & -0.0010 & 0.1151 \\ \\ 0.0010 & 0.0001 & -0.001 & 0.0126 & -0.0001 \\ 0.0001 & -0.0001 & 0.001 & -0.0079 & -0.0001 \\ 0.0001 & -0.0001 & -0.002 & -0.0179 & -0.0010 \\ -0.0010 & -0.0001 & 0.003 & 0.0131 & -0.0002 \end{bmatrix}$$

$$\begin{array}{cccccc}
0.0064 & 0.1072 & -0.2253 & 0.0640 & 0.0015 \\
0.0002 & 0.0024 & -0.0040 & -0.0016 & -0.0001 \\
-0.0014 & -0.0184 & 0.0852 & 0.0301 & 0.0004 \\
-0.0055 & -0.0910 & 0.1438 & -0.0923 & -0.0004 \\
\\
0.2770 & -0.004 & -0.002 & 0.017 & -0.0400 \\
0.0026 & 0.002 & -0.001 & -0.001 & 0.4850 \\
-0.0064 & -0.011 & -0.001 & 0.001 & -0.2182 \\
-0.2653 & 0.016 & 0.003 & -0.017 & -0.2320
\end{array}$$

and

$$\hat{W}_u = \begin{bmatrix}
0.0108 & -0.0011 & -0.0013 & 0.0464 & -0.0515 \\
-0.0389 & -0.0038 & -0.0126 & -0.1154 & -0.0345 \\
0.0990 & 0.0010 & -0.0011 & 0.5440 & 0.1863 \\
\\
6.9212 & -0.1401 & -0.0266 & 0.1011 & -0.0022 \\
0.6060 & 5.4144 & 0.0902 & -0.4428 & -0.0039 \\
-0.2832 & 0.2710 & 0.1602 & 0.4762 & 0.0006 \\
\\
0.0276 & 0.0074 & -0.1958 & -0.1729 & 0.1457 \\
-0.0427 & 0.0081 & 0.3125 & 0.1914 & -0.3870 \\
0.1827 & 0.1162 & 0.4931 & 0.4594 & 1.7896 \\
\\
-0.0058 & -0.0077 & -0.0010 & -0.0000 & 0.4769 \\
0.0839 & 0.1626 & -0.0229 & 0.0344 & 0.4023 \\
-0.0482 & -0.3090 & -0.0133 & -0.0392 & 0.8065
\end{bmatrix}.$$

Taking the current system data x_0 , u_0 , and y_0 into Θ network, we can obtain the coal quality as

$$\hat{\Theta}_k = [0.6789, 0.0373, 0.1149, 0.1689]^\top.$$

Taking the desired state $\eta = 1320$ and the coal quality $\hat{\Theta}_k$ into u_f network, we can obtain the desired control input expressed by $\hat{u}_{dk} = [61408.74, 44430.69, 51200]^\top$. According to the weights of model network, Θ network, and u_f network, we can easily obtain the system disturbance $\|w\| \leq 26.44$.

Next, the present iterative ADP algorithm is established to obtain the optimal tracking control law. Let the cost function be defined as in (13.3.12), where $A = 1$ and $C = 0.5$, and B is the identity matrix with suitable dimensions. The critic and action networks are both chosen as three-layer BP neural networks with the structures of 1–8–3 and 1–8–1, respectively. For each iteration, the critic and action networks are trained for 1000 steps using the learning rate of $l_c = l_a = 0.01$ so that the neural network training errors become less than 10^{-6} . Let the iteration index $i = 100$. The converged weights of the critic network and the action network are expressed as

$$W_c = [0.2194, 0.2555, 0.2797, 0.4137, 0.1777, 0.2165, 0.0092, 0.1841]$$

This PDF document was edited with **Icecream PDF Editor**.

Upgrade to PRO to remove watermark.

and

$$W_a = \begin{bmatrix} -0.0625 & 0.0746 & -0.0122 & -0.0717 \\ 0.0497 & -0.2990 & -0.1878 & 0.0750 \\ 0.0265 & 0.5098 & 0.3543 & 0.0404 \\ -0.0263 & -0.0236 & -0.0311 & 0.0031 \\ -0.1385 & 0.0146 & 0.1016 & 0.0823 \\ 0.2876 & 0.1056 & -0.0604 & -0.0862 \end{bmatrix},$$

Fig. 13.3 Convergence trajectory of iterative value function

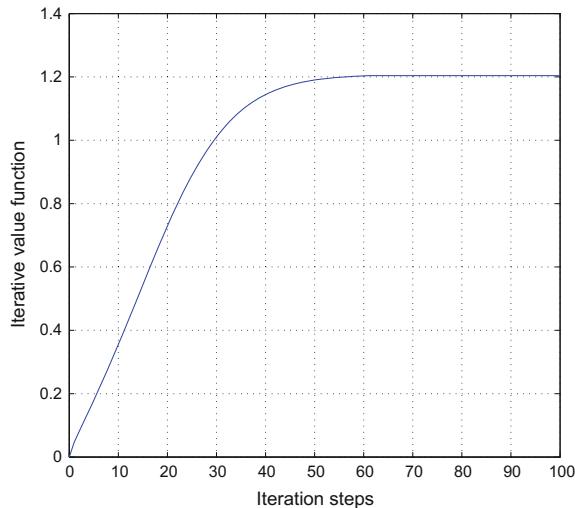
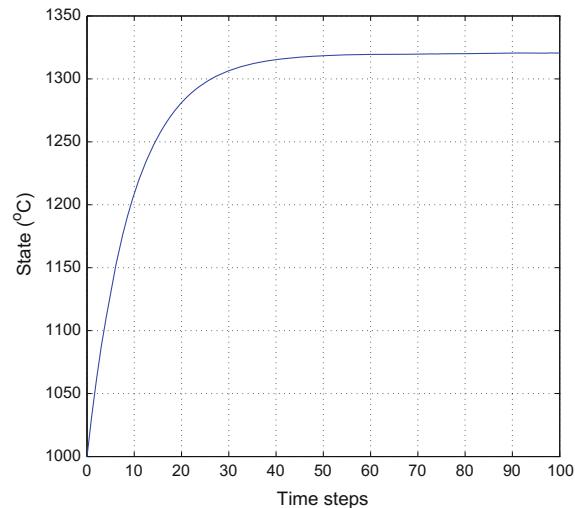


Fig. 13.4 Trajectory of state



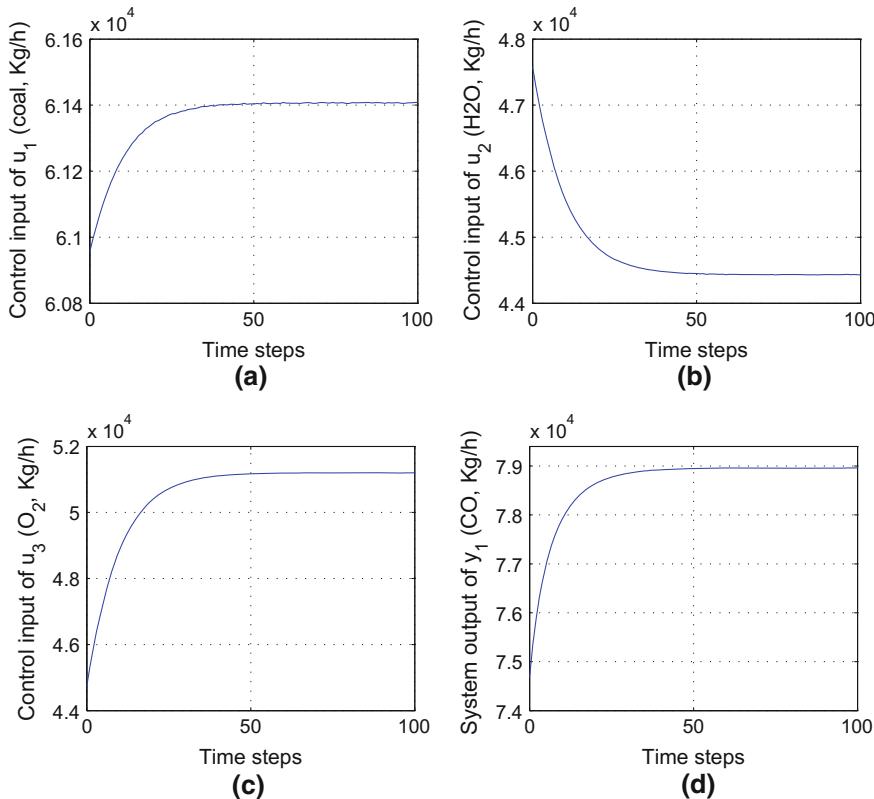


Fig. 13.5 Trajectories of control inputs and system output. **a** Coal input trajectory. **b** H_2O input trajectory. **c** O_2 input trajectory. **d** CO output trajectory

respectively. The convergence trajectory of the iterative value function is shown in Fig. 13.3. We apply the optimal control law to the system for $T_f = 100$ time steps and obtain the following results. The optimal state trajectory is shown in Fig. 13.4. The corresponding control trajectories and system output trajectories are shown in Figs. 13.5 and 13.6, respectively.

From the above numerical results, we can see that under the given NN training precisions, the system state tracks successfully the desired temperature using the optimal control derived by the present iterative ADP algorithm with system and iteration errors, which shows the effectiveness of the present algorithm. In the following, we will change the NN training precisions to show the performance of the present iterative ADP algorithm. First, we change the training precisions of model network, Θ network, and u_f network to 10^{-3} . While the training precisions of critic and action networks are kept at 10^{-6} .

Let the iteration index $i = 200$. The convergence trajectory of the iterative value function is shown in Fig. 13.7. We apply the optimal control law to the system for

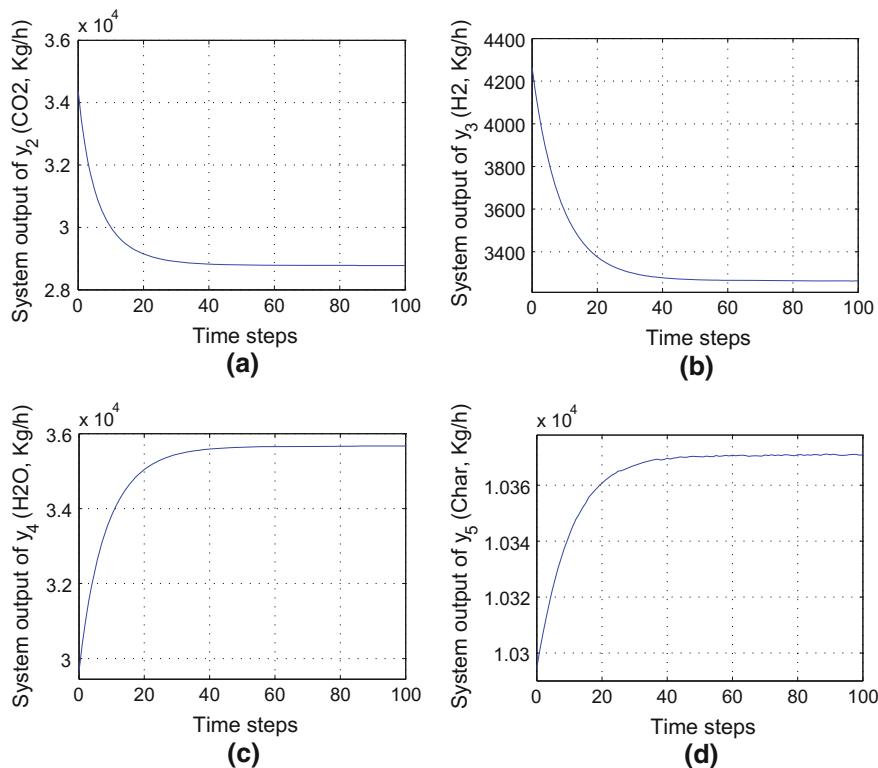
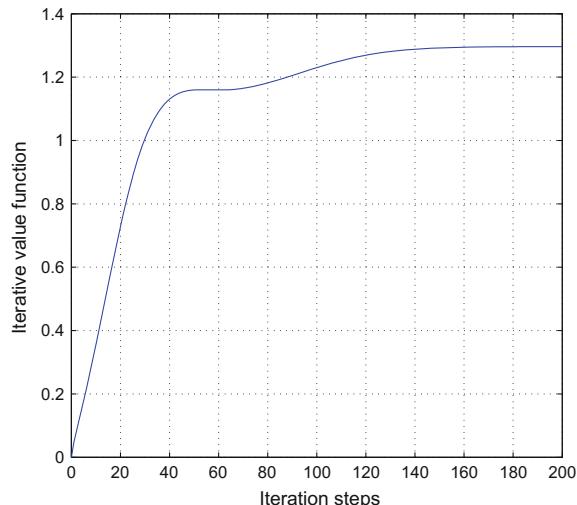


Fig. 13.6 Trajectories of system outputs. **a** CO₂ output trajectory. **b** H₂ output trajectory. **c** H₂O output trajectory. **d** Char output trajectory

Fig. 13.7 Convergence trajectory of iterative value function



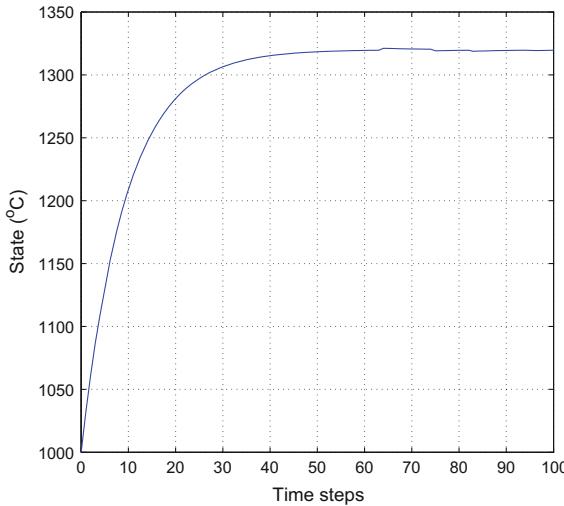


Fig. 13.8 Trajectory of state

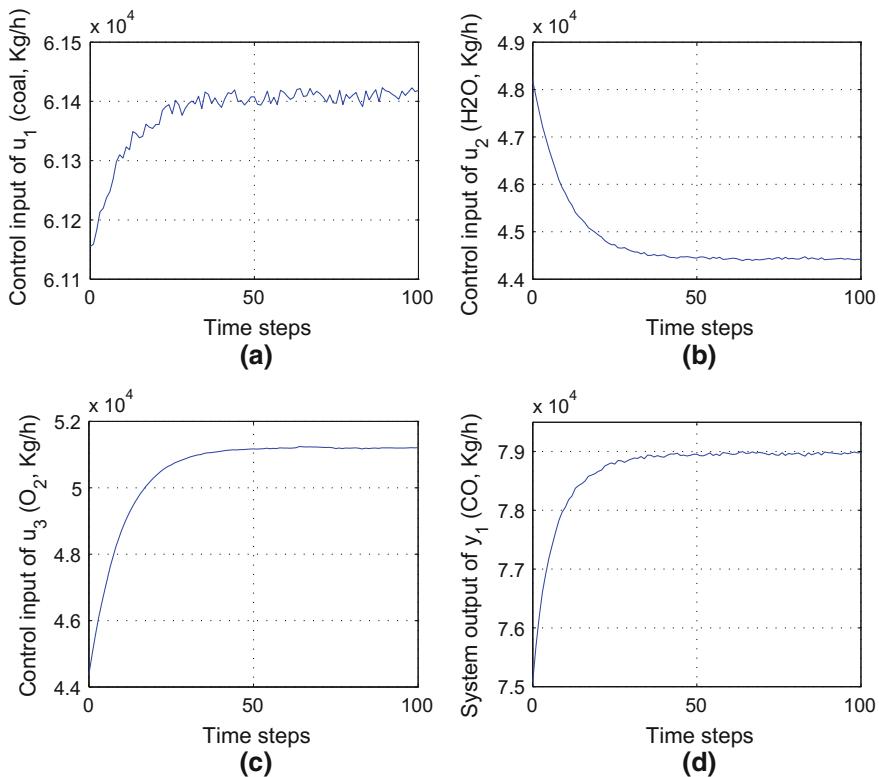


Fig. 13.9 Trajectories of control inputs and system output. **a** Coal input trajectory. **b** H_2O input trajectory. **c** O_2 input trajectory. **d** CO output trajectory

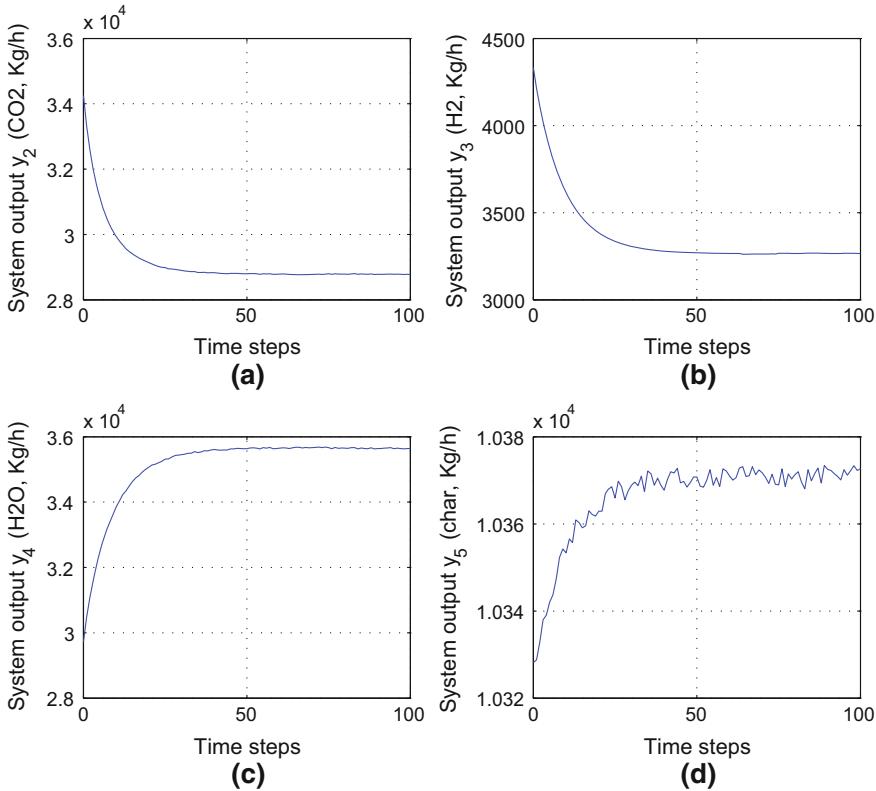


Fig. 13.10 Trajectories of system outputs. **a** CO₂ output trajectory. **b** H₂ output trajectory. **c** H₂O output trajectory. **d** Char output trajectory

$T_f = 100$ time steps and obtain the following results. The optimal state trajectory is shown in Fig. 13.8. The corresponding control trajectories and system output trajectories are shown in Figs. 13.9 and 13.10, respectively.

As is known, for the real-world coal gasification, the flow fluctuation of the control inputs is important and cannot be ignored. We will display the control system performance under the control disturbance. Let Δu_k be a zero-expectation white noise of control input, with $|\Delta u_{1k}| \leq 100$, $|\Delta u_{2k}| \leq 40$, $|\Delta u_{3k}| \leq 40$. The disturbance trajectories of the control input are displayed in Figs. 13.11a, b and c, respectively. Let the training precisions of model network, Θ network, and u_f network be 10^{-3} and the training precisions of critic and action networks are kept at 10^{-6} . The convergence trajectory of the iterative value function is shown in Fig. 13.12. The optimal state trajectory is shown in Fig. 13.13. The corresponding control trajectories and system output trajectories are shown in Figs. 13.14 and 13.15, respectively. From the numerical results, we can see that under the disturbance of the control input, we can also obtain the optimal tracking control of the system which shows the effectiveness

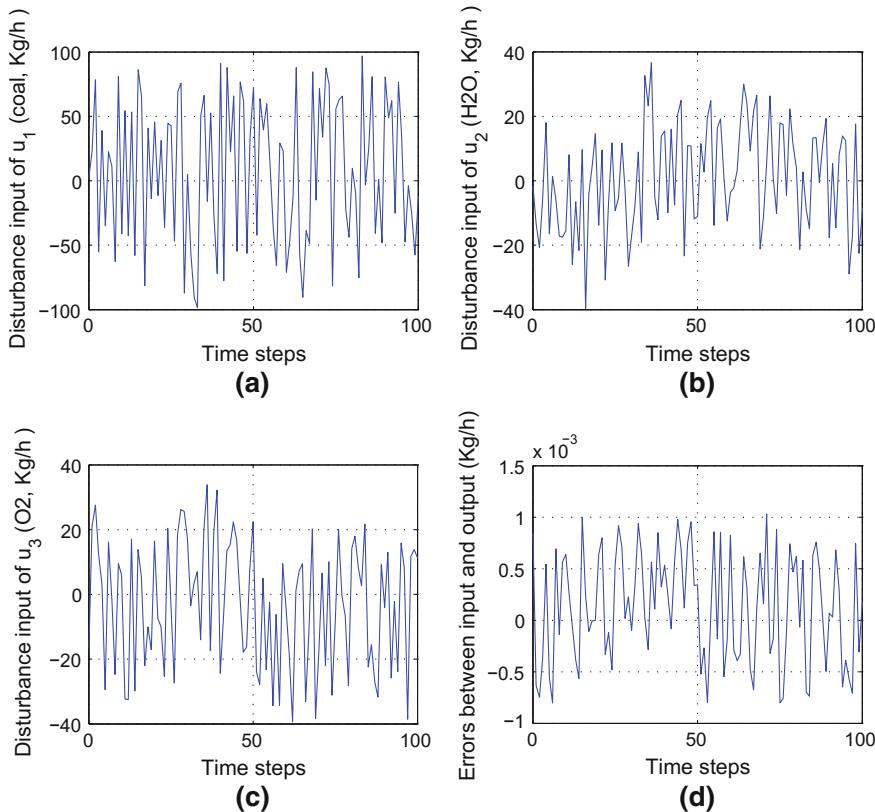
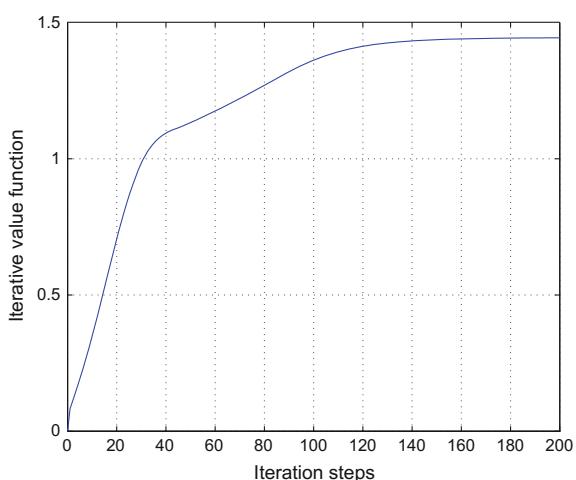


Fig. 13.11 Control disturbance and the input–output mass error. **a** Control disturbance Δu_{1k} . **b** Control disturbance Δu_{2k} . **c** Control disturbance Δu_{3k} . **d** The error between the input and output mass

Fig. 13.12 Convergence trajectory of iterative value function



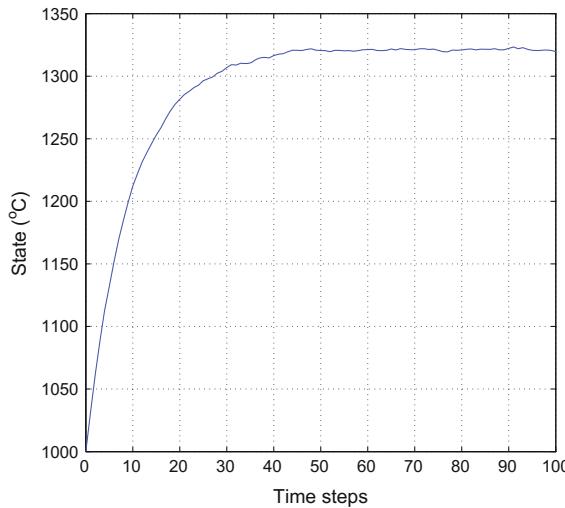


Fig. 13.13 Trajectory of state

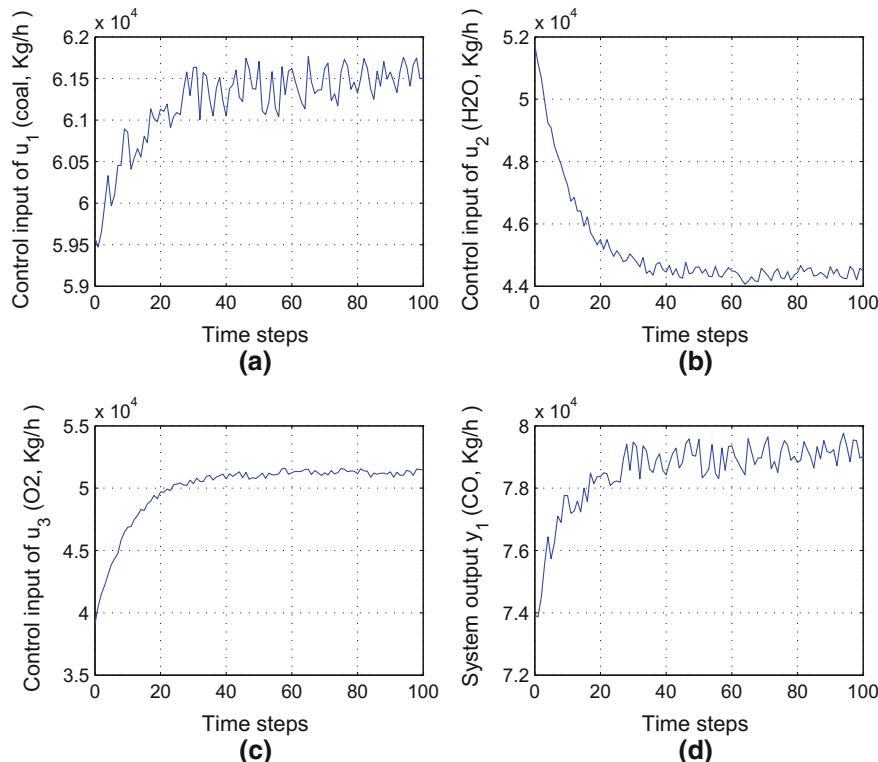


Fig. 13.14 Trajectories of control inputs and system output. **a** Coal input trajectory. **b** H_2O input trajectory. **c** O_2 input trajectory. **d** CO output trajectory

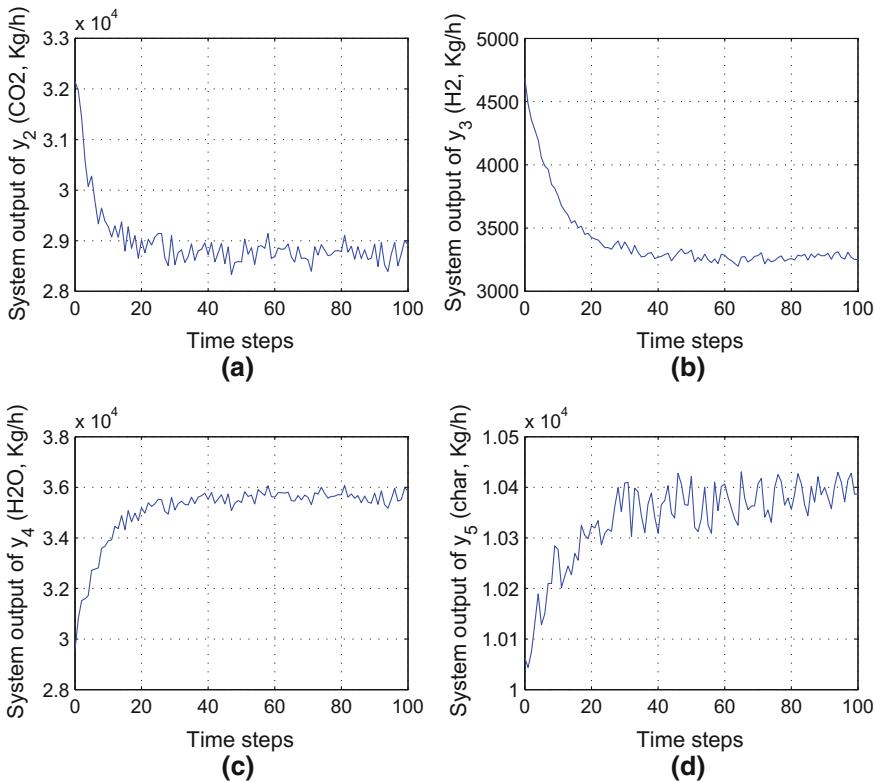


Fig. 13.15 Trajectories of the system outputs. **a** CO₂ output trajectory. **b** H₂ output trajectory. **c** H₂O output trajectory. **d** Char output trajectory

and robustness of the present iterative ADP method. To verify the correctness of the model and the present method, the mass errors between the input and output are given in Fig. 13.11d.

From the numerical results, we can see that when the system errors and the disturbance of the control input are enlarged, the iterative ADP algorithm is still effective to find the optimal tracking control scheme for the system. On the other hand, if we enlarge the iteration errors, the control property is quite different. Let the disturbance of the control $\Delta u_k = 0$. Let the training precisions for model network, Θ network, and u_f network be kept at 10^{-3} . We change the training precisions of critic and action networks to 10^{-3} . Let the iteration index $i = 100$. The convergence trajectory of the iterative value function is shown in Fig. 13.16a, where we can see that the iterative value function is not convergent any more. The corresponding state trajectory is shown in Fig. 13.16b, where we notice that the desired state is not achieved.

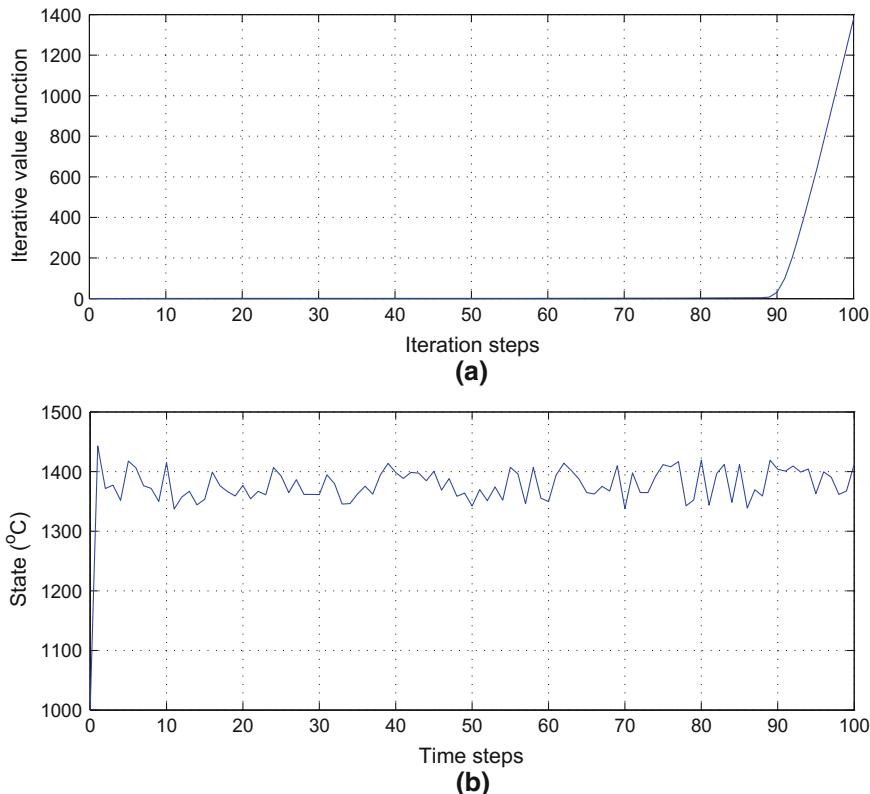


Fig. 13.16 Simulation results. **a** The trajectory of iterative value function. **b** The trajectory of state

13.5 Conclusions

In this chapter, an effective iterative ADP algorithm is established to solve the optimal tracking control problem for coal gasification systems. Using the input-state-output data of the system, NNs are used to approximate the system model, the coal quality, and the reference control, respectively, and the mathematical model of the coal gasification is unnecessary. Considering the system errors of NNs and the control disturbance, the optimal tracking control problem is transformed into a two-person zero-sum optimal regulation control problem. Iterative ADP algorithm is then established to obtain the optimal control law where the approximation errors in each iteration are considered. Convergence analysis is given to guarantee that the iterative value functions are convergent to a finite neighborhood of the optimal cost function.

References

1. Abani N, Ghoniem AF (2013) Large eddy simulations of coal gasification in an entrained flow gasifier. *Fuel* 104:664–680
2. Basar T, Bernard P (1995) H_∞ Optimal control and related minimax design problems. Birkhauser, Boston
3. Bhasin S, Kamalapurkar R, Johnson M, Vamvoudakis KG, Lewis FL, Dixon WE (2013) A novel actor-critic-identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica* 49(1):82–92
4. Chen Y, Li Z, Zhou M (2014) Optimal supervisory control of flexible manufacturing systems by petri nets: a set classification approach. *IEEE Trans Autom Sci Eng* 11(2):549–563
5. Gopalsami N, Raptis AC (1984) Acoustic velocity and attenuation measurements in thin rods with application to temperature profiling in coal gasification systems. *IEEE Trans Sonics Ultrason* 31(1):32–39
6. Guo R, Cheng G, Wang Y (2006) Texaco coal gasification quality prediction by neural estimator based on dynamic PCA. In: Proceedings of the IEEE international conference on mechatronics and automation, pp 2241–2246
7. Jia QS (2011) An adaptive sampling algorithm for simulation-based optimization with descriptive complexity preference. *IEEE Trans Autom Sci Eng* 8(4):720–731
8. Jin X, Hu SJ, Ni J, Xiao G (2013) Assembly strategies for remanufacturing systems with variable quality returns. *IEEE Trans Autom Sci Eng* 10(1):76–85
9. Kang Q, Zhou M, An J, Wu Q (2013) Swarm intelligence approaches to optimal power flow problem with distributed generator failures in power networks. *IEEE Trans Autom Sci Eng* 10(2):343–353
10. Kostur K, Kacur J (2012) Developing of optimal control system for UCG. In: Proceedings of the international carpathian control conference, pp 347–352
11. Liu D, Wei Q (2013) Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems. *IEEE Trans Cybern* 43(2):779–789
12. Matveev IB, Messerle VE, Ustimenko AB (2009) Investigation of plasma-aided bituminous coal gasification. *IEEE Trans Plasma Sci* 37(4):580–585
13. Ruprecht P, Schafer W, Wallace P (1988) A computer model of entrained coal gasification. *Fuel* 67(6):739–742
14. Serbin SI, Matveev IB (2010) Theoretical investigations of the working processes in a plasma coal gasification system. *IEEE Transactions on Plasma Science* 12(38):3300–3305
15. Si J, Wang YT (2001) Online learning control by association and reinforcement. *IEEE Trans Neural Netw* 12(2):264–276
16. Wei Q, Liu D (2014) Adaptive dynamic programming for optimal tracking control of unknown nonlinear systems with application to coal gasification. *IEEE Trans Autom Sci Eng* 11(4):1020–1036
17. Wigstrom O, Lennartson B, Vergnano A, Breitholtz C (2013) High-level scheduling of energy optimal trajectories. *IEEE Trans Autom Sci Eng* 10(1):57–64
18. Wilson JA, Chew M, Jones WE (2006) State estimation-based control of a coal gasifier. *IEEE Proc-Control Theory Appl* 153(3):268–276
19. Xu J, Qiao L, Gore J (2013) Multiphysics well-stirred reactor modeling of coal gasification under intense thermal radiation. *Int J Hydrg Energy* 38(17):7007–7015
20. Yang Q, Jagannathan S (2012) Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators. *IEEE Trans Syst, Man, Cybern-Part B: Cybern* 42(2):377–390

Chapter 14

Data-Based Neuro-Optimal Temperature Control of Water Gas Shift Reaction

14.1 Introduction

Water gas shift (WGS) reactor is an essential component of the coal-based chemical industry [11]. The WGS reactor combines carbon monoxide (CO) and water (H₂O) in the reactant stream to produce carbon dioxide (CO₂) and hydrogen (H₂). Proper regulation of the operating temperature is critical to achieving adequate CO conversion during transients [3]. Hence, optimal control of the reaction temperate is a key problem for WGS reaction process. To describe the dynamics of the WGS reaction process, many discussions focused on WGS modeling approaches [6, 16]. Unfortunately, the established WGS models are generally complex with high nonlinearities. Thus, the traditional linearized control method [28, 31, 32] is only effective in the neighborhood of the equilibrium point. When the required operating range is large, the nonlinearities in the system cannot be properly compensated by using a linear model. Therefore, it is necessary to study optimal control approaches for the original nonlinear system [3, 11]. Although optimal control of nonlinear systems has been the focus of control field in the last several decades [1, 2, 4, 5, 9, 18, 19, 22, 29], the optimal controller design for WGS reaction systems (WGS systems in brief) is still challenging, due to the complexity of the WGS reaction process.

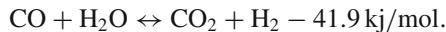
Iterative ADP method has played an important role as an effective way to solve Bellman equation indirectly and received lots of attentions [12, 14, 23, 24, 27, 30, 33]. For most previous ADP algorithms, it requires that the system model, the iterative control, and the cost function can accurately be approximated which guarantees the convergence property of the proposed algorithms. In real-world implementation of ADP, e.g., for WGS systems, the reconstruction errors by approximators and the disturbances of system states and controls inherently exist. Thus, the system models, iterative control laws, and cost functions are impossible to obtain accurately. Although in [8, 21], ADP was explored to design optimal temperature controller of the WGS system, the effects of approximation errors and disturbances were not considered. Furthermore, the convergence and stability properties were not discussed.

In this chapter, a stable iterative ADP algorithm is developed to obtain the optimal control law for the WGS system, such that the temperature of WGS system tracks the desired temperature [26].

14.2 System Description and Data-Based Modeling

14.2.1 Water Gas Shift Reaction

The WGS reaction inputs the water gas, which includes CO, CO₂, H₂, and H₂O, into the WGS reactor. The WGS reaction, which is slightly exothermic, converts CO to CO₂ and H₂ as shown in the following equation



The WGS reaction rate [3] can be described as follows:

$$r_{\text{WGS}} = \rho_{\text{cat}} k_r \exp\left(-\frac{5126 \text{ K}}{T}\right) [\text{CO}]^{0.78} [\text{H}_2\text{O}]^{0.15} \left(1 - \frac{[\text{CO}_2][\text{H}_2]}{[\text{CO}][\text{H}_2\text{O}]K_T}\right), \quad (14.2.1)$$

where the rate is in (kmol/m³/s). The catalyst density is $\rho_{\text{cat}} = 1.8 \times 10^{-4} \text{ kg/m}^3$. T is the current reaction temperature in K (Kelvin). The rate constant is $k_r = 1.32 \times 10^9 \text{ kmol/kg/s}$. The reaction equilibrium coefficient K_T is given as in [17], which is expressed by $K_T = \exp\left(\frac{4577.7 \text{ K}}{T} - 4.33\right)$.

For the WGS reaction (14.2.1), we can see that the reaction temperature is the key parameter [3, 11].

Let u_k denote the control input representing the flow of water gas (m³/s). Let

$$P(u_k) = u_k[\theta_{\text{CO}}, \theta_{\text{CO}_2}, \theta_{\text{H}_2}, \theta_{\text{H}_2\text{O}}]^\top$$

where θ_{CO} , θ_{CO_2} , θ_{H_2} , and $\theta_{\text{H}_2\text{O}}$ denote the given percentage compositions of CO, CO₂, H₂, and H₂O. Generally speaking, the water gas of WGS systems comes from the previous reaction process, such as coal gasification [16]. This means that the composition ratios of the mixed gas are uncontrollable for the WGS systems and the amount of water gas flow is the only one to be controlled. Let x_k denote the temperature of the WGS reactor. The WGS system can be expressed as

$$x_{k+1} = F(x_k, u_k), \quad (14.2.2)$$

where $F(\cdot)$ is an unknown system function. Let $x_k \in \mathbb{R}$ and $u_k \in \mathbb{R}$. Let the desired state be τ . Then, our goal is to design an optimal state feedback tracking control law $u_k^* = u^*(x_k)$, such that the system state tracks the desired state trajectory.

This PDF document was edited with **Icecream PDF Editor**.

Upgrade to PRO to remove watermark.

14.2.2 Data-Based Modeling and Properties

Three-layer backpropagation (BP) NNs are introduced to construct the dynamics of the WGS system and solve the reference control, respectively. Let \mathcal{L} be the number of hidden layer neurons. Let $X \in \mathbb{R}^{\mathcal{N}}$ be the input of NN, and let $Z \in \mathbb{R}^{\mathcal{M}}$ be the output. Then, function of the BP NNs can be expressed by

$$Z = \hat{F}_N(X, Y_f, W_f) = W_f^T \sigma(Y_f X),$$

where $Y_f \in \mathbb{R}^{\mathcal{L} \times \mathcal{N}}$ is the input–hidden layer weight matrix and $W_f \in \mathbb{R}^{\mathcal{L} \times \mathcal{M}}$ is the hidden–output layer weight matrix. Let σ be a sigmoid activation function [20, 25]. For the convenience of analysis, only the hidden–output layer weight W_f is updated during the NN training, while the input–hidden weight is fixed [30]. Hence, the NN function will be simplified by the expression $\hat{F}_N(X, W_f) = W_f^T \sigma_N(X)$, where $\sigma_N(X) = \sigma(Y_f X)$.

The model NN of system (14.2.2) can be written as

$$x_{k+1} = W_m^{*\top} \sigma_m(z_k) + \varepsilon_{m1k},$$

where $z_k = [x_k, u_k]^T$ denotes the NN input and $W_m^* \in \mathbb{R}^{L_m \times 1}$ denotes the ideal weight matrix of model NN, where L_m is the number of hidden layer neurons. Let $\sigma_m(z_k) = \sigma(Y_m z_k)$, where Y_m is an arbitrary weight matrix with a suitable dimension. Let $\|\sigma_m(\cdot)\| \leq \bar{\sigma}_M$ for a constant $\bar{\sigma}_M > 0$, and let ε_{m1k} be the bounded NN reconstruction error such that $|\varepsilon_{m1k}| \leq \bar{\varepsilon}_{m1}$ for a constant $\bar{\varepsilon}_{m1} > 0$. To train the model NN, it requires an array of WGS system and control data, such as the data from a period of time. The NN model for the system is constructed as

$$\hat{x}_{k+1} = \hat{W}_{mk}^T \sigma_m(z_k), \quad (14.2.3)$$

where \hat{x}_k is the estimated system state vector. Let \hat{W}_{mk} be the estimation of the ideal weight matrix W_m^* . Then, we define the system identification error as

$$\tilde{x}_{k+1} = \hat{x}_{k+1} - x_{k+1} = \tilde{W}_{mk}^T \sigma_m(z_k) - \varepsilon_{m1k},$$

where $\tilde{W}_{mk} = \hat{W}_{mk} - W_m^*$. Let $\phi_{mk} = \tilde{W}_{mk}^T \sigma_m(z_k)$. We can get

$$\tilde{x}_{k+1} = \phi_{mk} - \varepsilon_{m1k}.$$

The weights are adjusted to minimize the following error function

$$E_{mk} = \frac{1}{2} \tilde{x}_{k+1}^T \tilde{x}_{k+1} = \frac{1}{2} \tilde{x}_{k+1}^2.$$

By a gradient descent adaptation rule [20, 25], the weights are updated as

$$\hat{W}_{m,k+1} = \hat{W}_{mk} - l_m \sigma_m(z_k) \tilde{x}_{k+1}, \quad (14.2.4)$$

where $l_m > 0$ is the learning rate.

Theorem 14.2.1 *Let the model network (14.2.3) be used to identify WGS system (14.2.2). If there exists a constant $0 < \lambda_m < 1$ such that $\varepsilon_{m1k}^2 \leq \lambda_m \tilde{x}_k^2$, then the system identification error \tilde{x}_k is asymptotically stable and the error matrix \tilde{W}_{mk} converges to zero, as $k \rightarrow \infty$.*

Proof Consider the following Lyapunov function candidate defined as

$$L(\tilde{x}_k, \tilde{W}_{mk}) = \tilde{x}_k^2 + \frac{1}{l_m} \text{tr}\{\tilde{W}_{mk}^\top \tilde{W}_{mk}\}.$$

Taking the difference of the Lyapunov function candidate, we can obtain

$$\begin{aligned} \Delta L(\tilde{x}_k, \tilde{W}_{mk}) &\leq \phi_{mk}^2 + \varepsilon_{m1k}^2 - \tilde{x}_k^2 + 2l_m \sigma_m^\top(z_k) \sigma_m(z_k) (\phi_{mk}^2 + \varepsilon_{mk}^2) \\ &\leq -(1 - 2l_m \sigma_M^2) \phi_{mk}^2 - (1 - \lambda_m (1 + 2l_m \sigma_M^2)) \tilde{x}_k^2. \end{aligned}$$

Selecting the learning rate

$$l_m < \min \left\{ \frac{1}{2\sigma_M^2}, \frac{1 - \lambda_m}{2\lambda_m \sigma_M^2} \right\},$$

we can obtain $\Delta L(\tilde{x}_k, \tilde{W}_{mk}) \leq 0$. The proof is complete.

Next, we will solve the reference control by NN (u_f network in brief). According to the state equation in (14.2.2), we give x_k and x_{k+1} to approximate the reference control function u_{fk} , which is expressed as $u_{fk} = F_u(x_k, x_{k+1})$. We notice that solving u_{fk} needs the data of x_{k+1} . Hence, it requires to adopt off-line or history data to train u_f network. Let the number of hidden layer neurons be L_u . Let W_u^* be the ideal weight matrix. The NN representation of u_f network can be written as

$$u_{fk} = W_u^{*\top} \sigma_u(z_{uk}) + \varepsilon_{u1k},$$

where $z_{uk} = [x_k, x_{k+1}]^\top$ and ε_{u1k} is the NN reconstruction error such that $|\varepsilon_{u1k}| \leq \bar{\varepsilon}_u$ for a constant $\bar{\varepsilon}_u > 0$. Let $\sigma_u(z_{uk}) = \sigma(Y_u z_{uk})$, where Y_u is an arbitrary weight matrix with a suitable dimension.

The NN reference control is constructed as

$$\hat{u}_{fk} = \hat{F}_u(x_k, x_{k+1}) = \hat{W}_{uk}^\top \sigma_u(z_{uk}),$$

where \hat{u}_{fk} is the estimated reference control, and \hat{W}_{uk} is the estimated weight matrix. Define the identification error as

$$\tilde{u}_{fk} = \hat{u}_{fk} - u_{fk} = \phi_{uk} - \varepsilon_{u1k},$$

where $\phi_{uk} = \tilde{W}_{uk}^\top \sigma_u(z_{uk})$ and $\tilde{W}_{uk} = W_u^* - \hat{W}_{uk}$. The weight of u_f network is adjusted to minimize the error function

$$E_{uk} = \frac{1}{2} \tilde{u}_{fk}^2.$$

By gradient-based adaptation rule, the weight is updated as

$$\hat{W}_{u,k+1} = \hat{W}_{uk} - l_u \sigma_u(z_{uk}) \tilde{u}_{fk}, \quad (14.2.5)$$

where $l_u > 0$ is the learning rate.

Theorem 14.2.2 *Let the NN weight of u_f network be updated by (14.2.5). If there exists a constant $0 < \lambda_u < 1$ such that $\phi_{uk} \varepsilon_{u1k} \leq \lambda_u \phi_{uk}^2$, then the error matrix \tilde{W}_{uk} asymptotically converges to zero, as $k \rightarrow \infty$.*

14.3 Design of Neuro-Optimal Temperature Controller

In this section, a stable iterative ADP algorithm will be employed to obtain the optimal control law such that the temperature of WGS system tracks the desired one with convergence and stability analysis.

14.3.1 System Transformation

For WGS system (14.2.2), if we let the desired state be τ , then we can define the tracking error $e_k = x_k - \tau$. Let u_{dk} be the corresponding desired reference control (desired control in brief) for the desired state τ . As the system function is unknown, the desired control u_{dk} cannot directly be obtained by the WGS system (14.2.2). On the other hand, in the real-world WGS systems, the disturbances of the system and control input are both unavoidable. Thus, the system transformation method with accurate system model [34] is difficult to implement. To overcome these difficulties, a system transformation with NN reconstruction errors and disturbances is developed. First, according to the desired state τ , we can obtain $u_{dk} = F_u(\tau, \tau)$.

Let

$$\hat{u}_{dk} = \hat{F}_u(\tau, \tau) = \hat{W}_{uk}^\top \sigma_u(\tau, \tau)$$

be the output of u_f network. Let ε_{u2k} be an unknown bounded control disturbance such that $|\varepsilon_{u2k}| \leq \bar{\varepsilon}_{u2}$ for a constant $\bar{\varepsilon}_{u2} > 0$. Then, we can define the control error u_{ek} as

$$u_{ek} = u_k - \hat{u}_{dk} - \varepsilon_{uk},$$

where $\varepsilon_{uk} = \varepsilon_{u1k} + \varepsilon_{u2k}$. As ε_{u1k} and ε_{u2k} are bounded, there exists a constant $\bar{\varepsilon}_u > 0$ such that $|\varepsilon_{uk}| \leq \bar{\varepsilon}_u$. On the other hand, let $\hat{F}(z_k) = \hat{W}_{mk}^\top \sigma_m(z_k)$ be the model NN function. Let ε_{m2k} be an unknown bounded system disturbance such that $|\varepsilon_{m2k}| \leq \bar{\varepsilon}_{m2}$ for a constant $\bar{\varepsilon}_{m2} > 0$. Then, the tracking error system e_{k+1} can be defined as

$$\begin{aligned} e_{k+1} &= \bar{F}(e_k, u_{ek}) \\ &= \hat{F}((e_k + \tau), (u_{ek} + \hat{u}_{dk})) - \tau + \nabla \hat{F}(\xi_u) \varepsilon_u + \varepsilon_{mk}, \end{aligned} \quad (14.3.1)$$

where

$$\nabla \hat{F}(\xi_u) = \frac{\partial \hat{F}((e_k + \tau), \xi_u)}{\partial \xi_u},$$

$$\xi_u = c_u(u_{ek} + \hat{u}_{dk}) + (1 - c_u)(u_{ek} + \hat{u}_{dk} + \varepsilon_{uk}),$$

and $0 \leq c_u \leq 1$. Let $\varepsilon_{mk} = \varepsilon_{m1k} + \varepsilon_{m2k}$. We have $|\varepsilon_{mk}| \leq \bar{\varepsilon}_m$ for a constant $\bar{\varepsilon}_m > 0$. Let the NN tracking error \hat{e}_{k+1} be expressed as

$$\begin{aligned} \hat{e}_{k+1} &= F_e(e_k, \hat{u}_{ek}) \\ &= \hat{F}((e_k + \tau), (u_{ek} + \hat{u}_{dk})) - \tau. \end{aligned} \quad (14.3.2)$$

We can get $e_{k+1} = \hat{e}_{k+1} + \varepsilon_{ek}$, where we define

$$\varepsilon_{ek} = \nabla \hat{F}(\xi_u) \varepsilon_u + \varepsilon_{mk}$$

as the system error such that $|\varepsilon_{ek}| \leq \bar{\varepsilon}_e$ for a constant $\bar{\varepsilon}_e > 0$.

14.3.2 Derivation of Stable Iterative ADP Algorithm

In this section, our goal is to design an optimal control scheme such that the tracking error e_k converges to zero. Let

$$U(e_k, u_{ek}) = Qe_k^2 + Ru_{ek}^2$$

be the utility function, where Q and R are positive constants. Define the cost function as

$$J(e_0, \underline{u}_{e0}) = \sum_{k=0}^{\infty} U(e_k, u_{ek}),$$

where we let $\underline{u}_{ek} = (u_{ek}, u_{e,k+1}, \dots)$. The optimal cost function can be defined as

$$J^*(e_k) = \inf_{\underline{u}_{ek}} \{J(e_k, \underline{u}_{ek})\}.$$

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

$$J^*(e_k) = \min_{u_{ek}} \{U(e_k, u_{ek}) + J^*(e_{k+1})\}. \quad (14.3.3)$$

Define the laws of optimal controls as

$$u_e^*(e_k) = \arg \min_{u_{ek}} \{U(e_k, u_{ek}) + J^*(e_{k+1})\}.$$

Hence, Bellman equation (14.3.3) can be written as

$$J^*(e_k) = U(e_k, u_e^*(e_k)) + J^*(e_{k+1}). \quad (14.3.4)$$

Generally speaking, $J^*(e_k)$ is a highly nonlinear and nonanalytical function, which is nearly impossible to obtain by solving (14.3.4) directly. To overcome this difficulty, a new ADP algorithm is developed to obtain the optimal control law iteratively.

In the present stable iterative ADP algorithm, the value function and control law are updated by iterations, with the iteration index i increasing from 0 to infinity. First, let $\mu(e_k)$ be an arbitrary admissible control law, and let $P(e_k)$ be the corresponding value function which satisfies

$$P(e_k) = U(e_k, \mu(e_k)) + P(e_{k+1}). \quad (14.3.5)$$

Let the initial value function $\hat{V}_0(e_k) = P(e_k)$. The control law $\hat{v}_0(e_k)$ is obtained by

$$\hat{v}_0(e_k) = \arg \min_{\hat{u}_{ek}} \{U(e_k, \hat{u}_{ek}) + \hat{V}_0(\hat{e}_{k+1})\} + \rho_0(e_k). \quad (14.3.6)$$

Then, for $i = 1, 2, \dots$, the iterative ADP algorithm will iterate between

$$\hat{V}_i(e_k) = U(e_k, \hat{v}_{i-1}(e_k)) + \hat{V}_{i-1}(\hat{e}_{k+1}) + \pi_i(e_k), \quad (14.3.7)$$

and

$$\hat{v}_i(e_k) = \arg \min_{\hat{u}_{ek}} \{U(e_k, \hat{u}_{ek}) + \hat{V}_i(\hat{e}_{k+1})\} + \rho_i(e_k), \quad (14.3.8)$$

where $\pi_i(e_k)$ and $\rho_i(e_k)$ are iteration errors and $\hat{u}_{ek} = u_k - \hat{u}_{dk}$.

From the stable iterative ADP algorithm (14.3.6)–(14.3.8), we can see that the iterative value function $\hat{V}_i(e_k)$ is used to approximate $J^*(e_k)$ and the iterative control

law $\hat{v}_i(e_k)$ is used to approximate $u^*(e_k)$. Therefore, when $i \rightarrow \infty$, the algorithm should be convergent, i.e., $\hat{V}_i(e_k)$ and $\hat{v}_i(e_k)$ converge to the optimal ones. In the next section, we will show the properties of the present iterative ADP algorithm.

14.3.3 Properties of Stable Iterative ADP Algorithm with Approximation Errors and Disturbances

From the iterative ADP algorithm (14.3.6)–(14.3.8), as the existence of system errors, iteration errors, and disturbances, the convergence analysis methods for the accurate ADP algorithms are no longer valid. In this chapter, inspired by [13, 14], an “error bound”-based convergence and stability analysis will be developed. First, we define a new value function

$$\Gamma_i(e_k) = \min_{u_{ek}} \{U(e_k, u_{ek}) + \hat{V}_i(e_{k+1})\}. \quad (14.3.9)$$

Then, we can derive the following theorem.

Theorem 14.3.1 For $i = 0, 1, \dots$, the iterative value function $\hat{V}_i(e_k)$ and the iterative control law $\hat{v}_i(e_k)$ are obtained by (14.3.6)–(14.3.8). Let $\Gamma_i(e_k)$ be expressed as in (14.3.9). Then, there exists a constant $\sigma > 1$ such that

$$\hat{V}_i(e_k) \leq \sigma \Gamma_i(e_k) \quad (14.3.10)$$

holds uniformly.

Proof For all $i = 0, 1, \dots$, if we let

$$\bar{v}_i(e_k) = \arg \min_{u_{ek}} \{U(e_k, \hat{u}_{ek}) + \hat{V}_i(\hat{e}_{k+1})\}, \quad (14.3.11)$$

then

$$\bar{v}_i(e_k) = \hat{v}_i(e_k) - \rho_i(e_k).$$

According to (14.3.7), we have

$$\hat{V}_i(e_k) = U(e_k, (\bar{v}_{i-1}(e_k) + \rho_{i-1}(e_k)) + \pi_i(e_k) + V_{i-1}(F_e(e_k, (\bar{v}_{i-1}(e_k) + \rho_{i-1}(e_k)))).$$

Let

$$\nabla U(\xi) = \frac{\partial U(e_k, \xi)}{\partial \xi} \text{ and } \nabla V_i(\xi) = \frac{\partial \hat{V}_i(F_e(e_k, \xi))}{\partial \xi}.$$

Let $0 \leq c_{Ui} \leq 1$, $0 \leq c'_{Ui} \leq 1$, $0 \leq c_{Vi} \leq 1$ and $0 \leq c'_{Vi} \leq 1$ be constants and let $\xi_{Ui} = c_{Ui} \hat{v}_i(e_k) + (1 - c_{Ui}) \bar{v}_i(e_k)$, $\xi'_{Ui} = c'_{Ui} \hat{u}_{ek} + (1 - c'_{Ui}) u_{ek}$,

$$\xi_{Vi} = c_{Vi} \hat{V}_i(F_e(e_k, \bar{v}_i(e_k))) + (1 - c_{Vi}) \hat{V}_i(F_e(e_k, \hat{v}_i(e_k))),$$

and

$$\xi'_{Vi} = c'_{Vi} \hat{V}_i(F_e(e_k, \hat{u}_e(e_k))) + (1 - c'_{Vi}) \hat{V}_i(F_e(e_k, u_e(e_k))).$$

Then, the iterative value function $\hat{V}_i(e_k)$ can be expressed as

$$\begin{aligned} \hat{V}_i(e_k) &= U(e_k, (\bar{v}_{i-1}(e_k) + \rho_{i-1}(e_k))) + \pi_i(e_k) + V_{i-1}(F_e(e_k, (\bar{v}_{i-1}(e_k) + \rho_{i-1}(e_k)))) \\ &= \min_{u_{ek}} \{U(e_k, u_{ek}) + \nabla U(\xi'_{Ui}) \varepsilon_{uk} + V_{i-1}(e_{k+1}) + \nabla V_i(\xi'_{Vi}) \varepsilon_{ek}\} + \pi_i(e_k) \\ &\quad + \nabla U(\xi_{Ui}) \rho_i(e_k) + \nabla V_i(\xi_{Vi}) \rho_i(e_k). \end{aligned}$$

As $\nabla U(\xi'_{Ui})$, $\nabla V_i(\xi'_{Vi})$, $\nabla U(\xi_{Ui})$, and $\nabla V_i(\xi_{Vi})$ are upper bounded, if we let $|\nabla U(\xi'_{Ui}) \varepsilon_{uk}| \leq \bar{\varepsilon}_{Ui}$, $|\nabla V_i(\xi'_{Vi}) \varepsilon_{ek}| \leq \bar{\varepsilon}_{Vi}$, $|\nabla U(\xi_{Ui}) \rho_i(e_k)| \leq \varepsilon_{Ui}$, $|\nabla V_i(\xi_{Vi}) \rho_i(e_k)| \leq \varepsilon_{Vi}$, and $|\pi_i(e_k)| \leq \varepsilon_{\pi_i}$ for constants $\bar{\varepsilon}_{Ui}$, $\bar{\varepsilon}_{Vi}$, ε_{Ui} , and ε_{Vi} , then

$$\hat{V}_i(e_k) \leq \Gamma_i(e_k) + \varepsilon_i, \quad (14.3.12)$$

where $\varepsilon_i = \bar{\varepsilon}_{Ui} + \bar{\varepsilon}_{Vi} + \varepsilon_{Ui} + \varepsilon_{Vi} + \varepsilon_{\pi_i}$ is finite. Hence, for $i = 0, 1, \dots$, there exists a $\sigma \geq 1$ such that (14.3.10) holds uniformly. The proof is complete.

From Theorem 14.3.1, we can see that, for $i = 0, 1, \dots$, there must exist a finite $\sigma \geq 1$ such that (14.3.10) holds uniformly. Thus, σ can be seen as a uniform approximation error. Then, we can derive the following theorem.

Theorem 14.3.2 For all $i = 0, 1, \dots$, let $\Gamma_i(e_k)$ be expressed as in (14.3.10) where $\sigma \geq 1$ is a constant. Let $0 < \gamma < \infty$ and $1 \leq \delta < \infty$ be both constants such that

$$\begin{aligned} J^*(\bar{F}(e_k, u_{ek})) &\leq \gamma U(e_k, u_{ek}) \\ V_0(e_k) &\leq \delta J^*(e_k) \end{aligned}$$

holds uniformly. If the constant σ in (14.3.10) satisfies

$$\sigma \leq 1 + \frac{\delta - 1}{\gamma \delta}, \quad (14.3.13)$$

then the iterative value function $\hat{V}_i(e_k)$ converges to a finite neighborhood of the optimal cost function $J^*(e_k)$.

Proof The theorem can be proven in two steps. First, using mathematical induction, we will prove that, for $i = 0, 1, \dots$, the iterative value function $\hat{V}_i(e_k)$ satisfies

$$\hat{V}_i(e_k) \leq \sigma \left(1 + \sum_{j=1}^i \frac{\gamma^j \sigma^{j-1} (\sigma - 1)}{(\gamma + 1)^j} + \frac{\gamma^i \sigma^i (\delta - 1)}{(\gamma + 1)^i} \right) J^*(e_k). \quad (14.3.14)$$

Let $i = 0$. Then, (14.3.14) becomes $\hat{V}_0(e_k) \leq \sigma \delta J^*(e_k)$. We have the conclusion holds for $i = 0$. Assume that (14.3.14) holds for $i = l - 1$, $l = 1, 2, \dots$. Then, for $i = l$, we have

$$\begin{aligned} \Gamma_l(e_k) &\leq \min_{u_{ek}} \left\{ \left(1 + \gamma \sum_{j=1}^{l-1} \frac{\gamma^{j-1} \sigma^{j-1} (\sigma - 1)}{(\gamma + 1)^j} \right. \right. \\ &\quad \left. \left. + \frac{\gamma^{l-1} \sigma^{l-1} (\sigma \delta - 1)}{(\gamma + 1)^l} \right) U(e_k, u_{ek}) \right. \\ &\quad \left. + \left[\sigma \left(1 + \sum_{j=1}^l \frac{\gamma^j \sigma^{j-1} (\sigma - 1)}{(\gamma + 1)^j} + \frac{\gamma^l \sigma^l (\delta - 1)}{(\gamma + 1)^l} \right) \right. \right. \\ &\quad \left. \left. - \left(\sum_{j=1}^{l-1} \frac{\gamma^{j-1} \sigma^{j-1} (\sigma - 1) s}{(\gamma + 1)^j} + \frac{\gamma^{l-1} \sigma^{l-1} (\sigma \delta - 1)}{(\gamma + 1)^l} \right) \right] \right. \\ &\quad \left. \times J^*(\bar{F}(e_k, u_{ek})) \right\} \\ &= \left(1 + \sum_{j=1}^l \frac{\gamma^j \sigma^{j-1} (\sigma - 1)}{(\gamma + 1)^j} + \frac{\gamma^l \sigma^l (\delta - 1)}{(\gamma + 1)^l} \right) J^*(e_k), \end{aligned}$$

which proves (14.3.14). The mathematical induction is complete.

Second, according to (14.3.13), we have $\gamma^j \sigma^{j-1} / (\gamma + 1)^j < 1$, and hence, the geometrical series $\{\gamma^j \sigma^{j-1} (\sigma - 1) / (\gamma + 1)^j\}$ is finite as $i \rightarrow \infty$. According to (14.3.14), the iterative value function $\hat{V}_i(e_k)$ is convergent to a finite neighborhood of the optimal cost function $J^*(e_k)$. This completes the proof of the theorem.

Next, we can derive the stability property.

Theorem 14.3.3 For $i = 0, 1, \dots$, let $\hat{V}_i(e_k)$ and $\hat{v}_i(e_k)$ be obtained by (14.3.7) and (14.3.8), respectively. Then, the tracking error system (14.3.1) is UUB under the iterative control law $\hat{v}_i(e_k)$.

Proof According to (14.3.14), for $i = 0, 1, \dots$, let

$$\chi_i = \sigma \left(1 + \sum_{j=1}^i \frac{\gamma^j \sigma^{j-1} (\sigma - 1)}{(\gamma + 1)^j} + \frac{\gamma^i \sigma^i (\delta - 1)}{(\gamma + 1)^i} \right). \quad (14.3.15)$$

Define a new iterative value function as $\bar{V}_i(e_k) = \chi_i J^*(e_k)$, where χ_i is defined as in (14.3.15). According to (14.3.13), we can get $\chi_{i+1} - \chi_i \leq 0$, which means $\bar{V}_{i+1}(e_k) \leq \bar{V}_i(e_k)$. Let

$$\xi_{\bar{V}i} = c_{\bar{V}i} \bar{V}_i(F_e(e_k, \bar{v}_i(e_k))) + (1 - c_{\bar{V}i}) \bar{V}_i(F_e(e_k, \hat{v}_i(e_k))),$$

for $0 \leq c_{\bar{V}i} \leq 1$. Let $|\nabla(\xi_{\bar{V}i})\varepsilon_e| \leq \varepsilon_{\bar{V}i}$ for a constant $\varepsilon_{\bar{V}i}$, and we can get

$$\begin{aligned} \bar{V}_i(e_{k+1}) - \bar{V}_i(e_k) &\leq -U(e_k, \hat{v}_i(e_k)) + \nabla(\xi_{\bar{V}i})\varepsilon_e \\ &\leq -U(e_k, \hat{v}_i(e_k)) + \varepsilon_{\bar{V}i}. \end{aligned}$$

Define a new state error set $\Omega_e = \{e_k : U(e_k, \hat{v}_i(e_k)) \leq \varepsilon_{\bar{V}i}\}$. As $U(e_k, \hat{v}_i(e_k))$ is a positive-definite function, $|e_k|$ is finite for $e_k \in \Omega_e$. We define

$$e_M = \sup_{x \in \Omega_e} \{|x|\}.$$

Define two scalar functions $\alpha(|e_k|)$ and $\beta(|e_k|)$ which satisfy the following two conditions.

(1) If $|e_k| \leq e_M$, then

$$\alpha(|e_k|) = \beta(|e_k|) = \bar{V}_i(e_k). \quad (14.3.16)$$

(2) If $|e_k| > e_M$, then $\alpha(|e_k|)$ and $\beta(|e_k|)$ are both monotonically increasing functions and satisfy

$$0 < \alpha(|e_k|) \leq \bar{V}_i(e_k) \leq \beta(|e_k|). \quad (14.3.17)$$

For an arbitrary constant $\varsigma > e_M$, there exists a $\varrho(\varsigma) > e_M$ such that $\beta(\varrho) \leq \alpha(\varsigma)$. For $T = 1, 2, \dots$, if $|e_k| > e_M$ and $|e_{k+T}| > e_M$, then $\bar{V}_i(e_{k+T}) - \bar{V}_i(e_k) \leq 0$.

Hence, for all $|e_k| > e_M$ satisfying $e_M < |e_k| \leq \beta(\varrho)$, there exists a $T > 0$ such that

$$\alpha(\varsigma) \geq \beta(\varrho) \geq \bar{V}_i(e_k) \geq \bar{V}_i(e_{k+T}) \geq \alpha(|e_{k+T}|),$$

which obtains $\varsigma > |e_{k+T}|$. Therefore, for all $|e_k| > e_M$, there exists a $T = 1, 2, \dots$ such that $|e_{k+T}| \leq \varsigma$. As ς is arbitrary, let $\varsigma \rightarrow e_M$. Then, we can obtain $|e_{k+T}| \leq e_M$. According to the definition in [10], e_k is UUB.

Next, for $\hat{V}_i(e_k) \leq \bar{V}_i(e_k)$, there exists time instants T_0 and T_1 such that

$$\bar{V}_i(e_k) \geq \bar{V}_i(e_{k+T_0}) \geq \hat{V}_i(e_k) \geq \bar{V}_i(e_{k+T_1}) \quad (14.3.18)$$

for all $|e_k| > e_M$, $|e_{k+T_0}| > e_M$, and $|e_{k+T_1}| > e_M$. Choose $\varsigma_1 > 0$ to satisfy $\hat{V}_i(e_k) \geq \alpha(\varsigma_1) \geq \bar{V}_i(e_{k+T_1})$. Then, there exists $\varrho_1(\varsigma_1) > 0$ such that

$$\alpha(\varsigma_1) \geq \beta(\varrho_1) \geq \bar{V}_i(e_{k+T_1}).$$

According to (14.3.18) and the definition of $\alpha(|e_k|)$ and $\beta(|e_k|)$ in (14.3.16) and (14.3.17), we have

$$\alpha(\varsigma) \geq \beta(\varrho) \geq \hat{V}_i(e_k) \geq \alpha(\varsigma_1) \geq \beta(\varrho_1) \geq \bar{V}_i(e_{k+T_1}) \geq \alpha(|e_{k+T_1}|).$$

For an arbitrary constant $\varsigma > e_M$, we can obtain $|e_{k+T_1}| \leq \varsigma$, which shows that $\hat{V}_i(e_k)$ is a UUB control law for the tracking error system (14.3.1). This completes the proof of the theorem.

Corollary 14.3.1 For $i = 0, 1, \dots$, let $\hat{V}_i(e_k)$ and $\hat{v}_i(e_k)$ be obtained by (14.3.7) and (14.3.8), respectively. If

$$U(e_k, \hat{v}_i(e_k)) > \nabla V_i(\xi_{\bar{V}_i}) \varepsilon_{ek},$$

$\forall e_k$, then the iterative control law $\hat{v}_i(e_k)$ is an asymptotically stable control law for system (14.3.1).

14.4 Neural Network Implementation for the Optimal Tracking Control Scheme

In this section, neural networks, including action network and critic network, are used to implement the present stable iterative ADP algorithm. The whole structural diagram is shown in Fig. 14.1.

For all $i = 0, 1, \dots$, the critic network is used to approximate the value function in (14.3.8). Collect an array of tracking errors $\mathcal{E}_k = \{e_k^1, \dots, e_k^p\}$, where p is a large integer. For $j = 0, 1, \dots, p$, let the output of the critic network be $\hat{V}_i^j(e_k) = W_{ci}^{j\top} \sigma_c(e_k)$, where $\sigma_c(e_k) = \sigma(Y_c e_k)$ and Y_c is an arbitrary matrix with a suitable dimension.

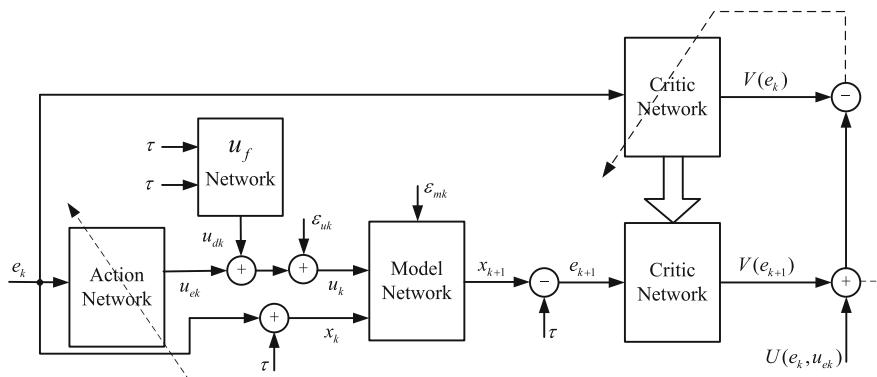


Fig. 14.1 The structural diagram of the stable iterative ADP algorithm

The target function can be written as

$$V_i(e_k) = U(e_k, \hat{v}_{i-1}(e_k)) + \hat{V}_{i-1}(\hat{e}_{k+1}).$$

Define the error function of the critic network as $\vartheta_{ci}^j(e_k^j) = V_i(e_k^j) - \hat{V}_i^j(e_k^j)$. The weights of the critic network are updated as [15, 25]

$$W_c^{i(j+1)} = W_c^{ij} - l_c \vartheta_{ci}^j(e_k^j) \sigma_c(e_k^j), \quad (14.4.1)$$

where $\|\sigma_c(e_k^j)\| \leq \sigma_C$ for a constant σ_C and $l_c > 0$ is the learning rate of critic network.

The action network is used to approximate the iterative control law $\bar{v}_i(e_k)$, where $\bar{v}_i(e_k)$ is defined by (14.3.11). The output can be formulated as $\hat{v}_i^j(e_k) = W_a^{ij\top} \sigma_a(e_k)$, where $\sigma_a(e_k) = \sigma(Y_a e_k)$. Let Y_a be an arbitrary matrix with a suitable dimension. According to \mathcal{E}_k , we can define the output error of the action network as $\vartheta_{ai}^j(e_k^j) = \hat{v}_i^j(e_k^j) - \bar{v}_i(e_k^j)$, $j = 1, 2, \dots, p$. The weight of the action network can be updated as

$$W_a^{i(j+1)} = W_a^{ij} - l_a \sigma_a(e_k^j) \vartheta_{ai}^j(e_k^j), \quad (14.4.2)$$

where $\|\sigma_a(e_k^j)\| \leq \sigma_A$ for a constant σ_A and $l_a > 0$ is the learning rate of action network. The weight convergence property of the neural networks is shown in the following theorem.

Theorem 14.4.1 For $j = 1, 2, \dots, p$, let the ideal critic and action network function be expressed by

$$V_i(e_k^j) = W_c^{*i\top} \sigma_c(e_k^j) + \varepsilon_{ci}(e_k^j),$$

and

$$\bar{v}_i(e_k^j) = W_a^{*i\top} \sigma_a(e_k^j) + \varepsilon_{ai}(e_k^j),$$

respectively. The critic and action networks are trained by (14.4.1) and (14.4.2), respectively. Let $\tilde{W}_{ci}^j = W_{ci}^j - W_{ci}^*$ and $\tilde{W}_{ai}^j = W_{ai}^j - W_{ai}^*$. For all $i = 1, 2, \dots$, if there exist constants $0 < \lambda_c < 1$ and $0 < \lambda_a < 1$ such that

$$\phi_{ci}^j \varepsilon_{ci}(e_k^j) \leq \lambda_c (\phi_{ci}^j)^2 \text{ and } \phi_{ai}^j \varepsilon_{ai}(e_k^j) \leq \lambda_a (\phi_{ai}^j)^2,$$

respectively, where $\phi_{ci}^j = \tilde{W}_{ci}^{ij\top} \sigma_c(e_k^j)$ and $\phi_{ai}^j = \tilde{W}_a^{ij\top} \sigma_a(e_k^j)$, then the error matrices \tilde{W}_c^{ij} and \tilde{W}_a^{ij} converge to zero, as $j \rightarrow \infty$.

Proof Consider the following Lyapunov function candidate

$$L(\tilde{W}_c^{ij}, \tilde{W}_a^{ij}) = \frac{1}{l_c} \text{tr}\{\tilde{W}_c^{ij\top} \tilde{W}_c^{ij}\} + \frac{1}{l_a} \text{tr}\{\tilde{W}_a^{ij\top} \tilde{W}_a^{ij}\}.$$

The difference of the Lyapunov function candidate is given by

$$\begin{aligned}\Delta L(\tilde{W}_c^{ij}, \tilde{W}_a^{ij}) &\leq -2((\phi_{cik}^j)^2 - \phi_{cik}^j \varepsilon_{ci}(e_k^j)) - 2((\phi_{aik}^j)^2 - \phi_{aik}^j \varepsilon_{ai}(e_k^j)) \\ &\quad + l_c \sigma_C^2 (\phi_{cik}^j - \varepsilon_{ci}(e_k^j))^2 + l_a \sigma_A^2 (\phi_{aik}^j - \varepsilon_{ai}(e_k^j))^2 \\ &\leq (-2(1 - \lambda_c) + l_c \sigma_C^2 (1 + \chi_c)) (\phi_{cik}^j)^2 \\ &\quad + (-2(1 - \lambda_a) + l_a \sigma_A^2 (1 + \chi_a)) (\phi_{aik}^j)^2,\end{aligned}$$

where we let $\chi_c > 0$ and $\chi_a > 0$ be constants such that $(\varepsilon_{ci}(e_k^j))^2 \leq \chi_c (\phi_{cik}^j)^2$ and $(\varepsilon_{ai}(e_k^j))^2 \leq \chi_a (\phi_{aik}^j)^2$, respectively. Selecting l_c and l_a such that

$$l_c < \frac{2(1 - \lambda_c)}{\sigma_C^2 (1 + \chi_c)}, \quad l_a < \frac{2(1 - \lambda_a)}{\sigma_A^2 (1 + \chi_a)},$$

we have $\Delta L(\tilde{W}_c^{ij}, \tilde{W}_a^{ij}) \leq 0$, $j = 1, 2, \dots, p$. Let $j \rightarrow \infty$, and we can obtain the conclusion. This completes the proof of the theorem.

Based on the above analysis, the whole data-driven stable iterative ADP algorithm for the WGS system can be summarized in Algorithm 14.4.1.

Algorithm 14.4.1 Data-driven stable iterative ADP algorithm

NN modeling and system transformation:

- Step 1. Collect an array of system data of the WGS system (14.2.2).
- Step 2. Establish model network, where the NN training rule is expressed as in (14.2.4).
- Step 3. Establish u_f network, where the NN training rule is expressed as in (14.2.5).
- Step 4. Transform the WGS tracking system (14.2.2) into an error regulation system (14.3.2).

Stable iterative ADP algorithm:

- Step 5. Let $i = 0$ and $V_0(e_k) = P(e_k)$, where $P(e_k)$ satisfies (14.3.5).
 - Step 6. Update the iterative value function $\hat{V}_i(e_k)$ by (14.3.7). Compute the iterative control law $\hat{v}_i(e_k)$ by (14.3.8).
 - Step 7. If the approximation error σ satisfies (14.3.13), then goto Step 8; else, reduce the approximation error σ and goto Step 6.
 - Step 8. If $|\hat{V}_i(e_k) - \hat{V}_{i-1}(e_k)| \leq \zeta$, then goto next step; else, let $i = i + 1$ and goto Step 6.
 - Step 9. Return $\hat{V}_i(e_k)$ and $\hat{v}_i(e_k)$.
-

Remark 14.4.1 One property should be pointed out. For all $i = 1, 2, \dots$, if we define the approximation error function $\varepsilon_i(e_k)$ as

$$\hat{V}_i(e_k) = \Gamma_i(e_k) + \varepsilon_i(e_k),$$

then according to (14.3.12), we have $\varepsilon_i(e_k) \leq \varepsilon$, where $\varepsilon = \sup\{\varepsilon_i\}$, $i = 0, 1, \dots$. According to (14.3.10) and (14.3.13), we can obtain the following equivalent convergence criterion

$$\varepsilon_i(e_k) \leq \frac{\hat{V}_i(e_k)(\delta - 1)}{\gamma\delta + \delta - 1}. \quad (14.4.3)$$

From (14.4.3), we can see that if $|e_k|$ is large, then the present iterative ADP algorithm permits convergence under large approximation errors, and if $|e_k|$ is small, then small approximation errors are required to ensure the convergence of the iterative ADP algorithm. As the existences of the approximation errors and disturbances, the convergence criterion (14.4.3) cannot generally be satisfied for every e_k . Define a new tracking error set

$$\Theta_e = \left\{ e_k : \varepsilon_i(e_k) > \frac{\hat{V}_i(e_k)(\delta - 1)}{\gamma\delta + \delta - 1} \right\}.$$

As $\varepsilon_i(e_k) \leq \varepsilon$ is finite, if we define $\Upsilon = \sup_{e_k \in \Theta_e} \{|e_k|\}$, then Υ is finite. Thus, for all $e_k \in \overline{\Theta}_e$, $\overline{\Theta}_e \triangleq \mathbb{R}^n \setminus \Theta_e$, we can get that $\hat{V}_i(e_k)$ is convergent, i.e.,

$$\hat{V}_\infty(e_k) = \lim_{i \rightarrow \infty} \hat{V}_i(e_k).$$

14.5 Numerical Analysis

In this section, numerical experiments will be studied to show the effectiveness of the present stable iterative ADP algorithm with approximation errors and disturbances. For WGS system (14.2.2), we let the initial reaction temperature be $x_0 = 273^\circ\text{C}$. Let the desired reaction temperature be $\tau = 375^\circ\text{C}$. Observe the volume percentage compositions in the inlet water gas of the WGS system, and we obtain $[\theta_{\text{CO}}, \theta_{\text{CO}_2}, \theta_{\text{H}_2}, \theta_{\text{H}_2\text{O}}] = [24.39\%, 14.71\%, 22.79\%, 38.11\%]$.

To model WGS system (14.2.2), we collect 20,000 input-state data from the actual WGS operational system. Then, three-layer BP NNs are established with the structures of 2–15–1 and 2–15–1 to approximate the WGS system and reference control, respectively. Give the disturbances of the system and control input in Fig. 14.2a, b, respectively. Let the learning rates of NNs be 0.001 and implement our iterative ADP algorithm for 25 iterations. The curve of the admissible approximation errors for the present ADP algorithm is displayed in Fig. 14.4. From Fig. 14.4, we can see that for different e_k and iteration index i , it requires different approximation error to guarantee the convergence of our iterative ADP algorithm. Let $\bar{\varepsilon} = \max\{\varepsilon_m, \varepsilon_u, \rho_i, \pi_i\}$ be the maximum reconstruction error of NNs for $i = 0, 1, \dots, 25$. We choose two reconstruction errors $\bar{\varepsilon}$'s, which are 10^{-6} and 10^{-4} , respectively, to train the neural networks. The convergence trajectories of the iterative value functions are shown in Fig. 14.2c, d, respectively.

Implement the iterative control law for the WGS system (14.2.2). Let the implementation time $T_f = 100$ and the trajectories of the states and controls are displayed

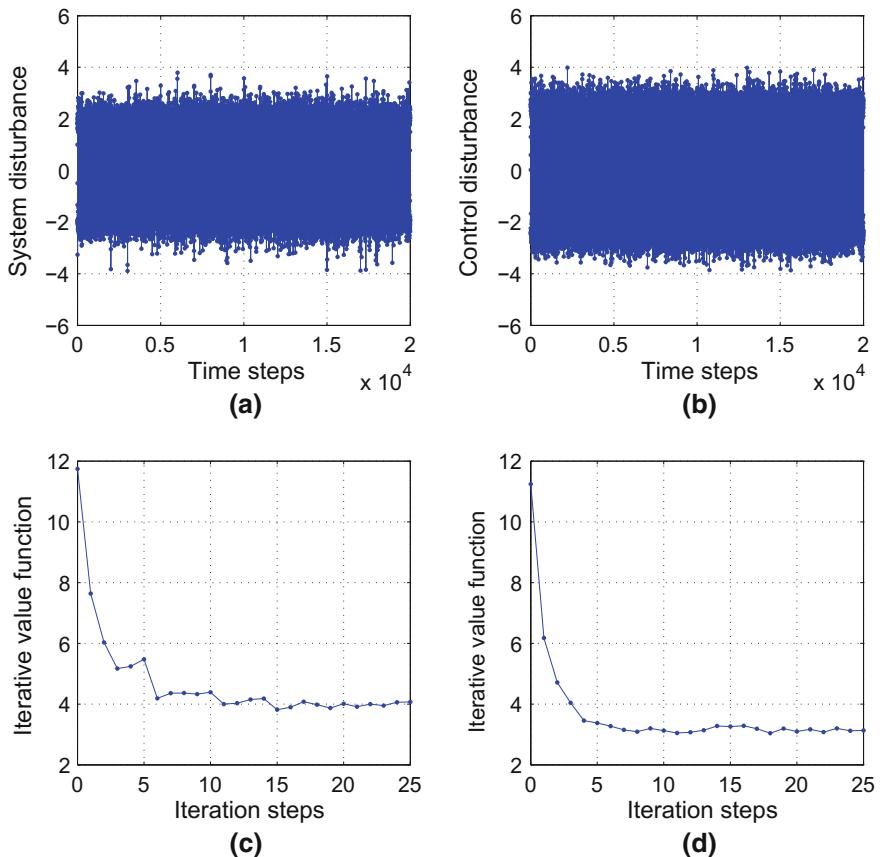


Fig. 14.2 Disturbances and iterative value function. **a** System disturbance. **b** Control disturbance. **c** Iterative value function under $\bar{\varepsilon} = 10^{-4}$. **d** Iterative value function under $\bar{\varepsilon} = 10^{-6}$.

in Fig. 14.3a–d, respectively. From the numerical results, we can see that by the stable iterative ADP algorithm, the iterative control law can guarantee the tracking error system to be UUB, which shows the robustness of the present algorithm. Moreover, we can see that if we enhance the training precisions of the NNs, such as reducing $\bar{\varepsilon}$ from 10^{-4} to 10^{-6} , then the approximation errors can be reduced and the system state will be closer to the desired one. The optimal state and control trajectories for $\bar{\varepsilon} = 10^{-6}$ are shown in Fig. 14.5a, b, respectively. In the real-world neural network training, the training precision of NNs is generally set to a uniform one. Thus, it is recommended that the present iterative ADP algorithm is implemented with a high training precision which allows the iterative value function to converge for most of the state space.

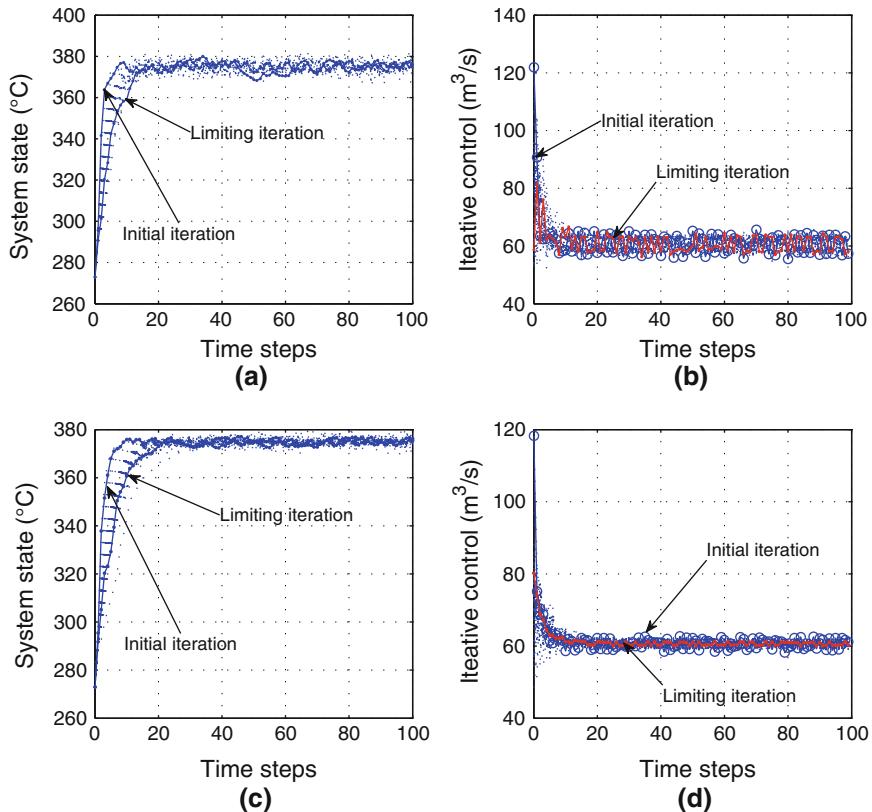
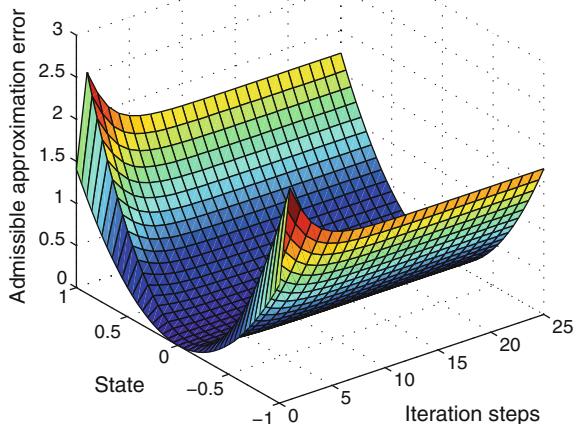


Fig. 14.3 Iterative trajectories of states and controls for different $\bar{\varepsilon}$'s. **a** State for $\bar{\varepsilon} = 10^{-4}$. **b** Control for $\bar{\varepsilon} = 10^{-4}$. **c** State for $\bar{\varepsilon} = 10^{-6}$. **d** Control for $\bar{\varepsilon} = 10^{-6}$.

Fig. 14.4 The curve of the admissible approximation errors



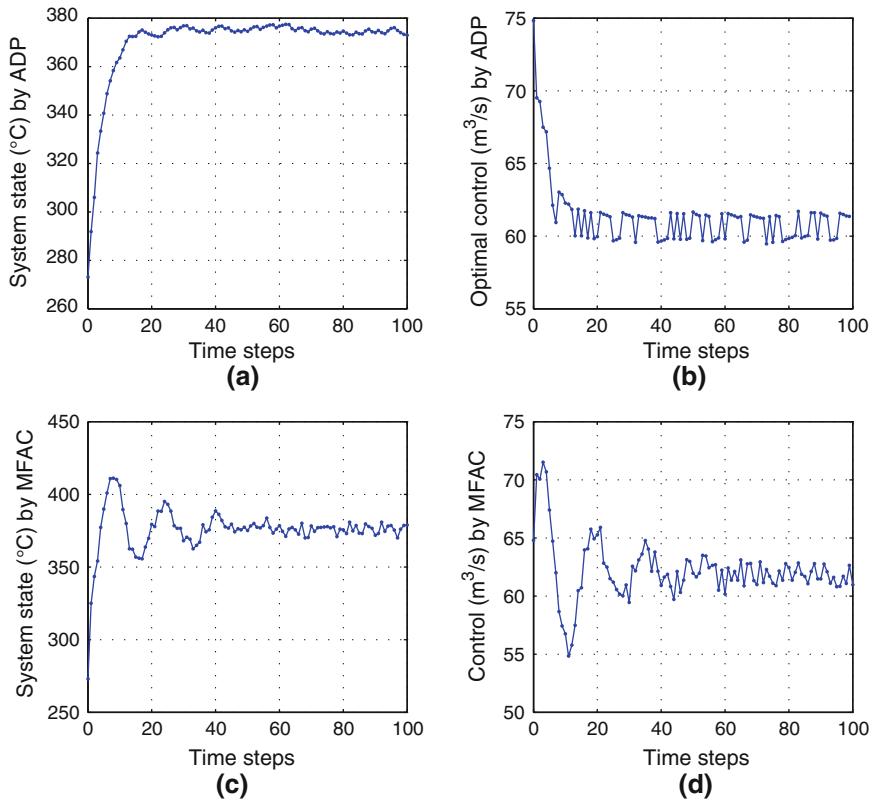


Fig. 14.5 Comparisons by ADP and MFAC. **a** State trajectory by ADP. **b** Control trajectory by ADP. **c** State trajectory by MFAC. **d** Control trajectory by MFAC.

On the other hand, to show the effectiveness of the stable iterative ADP algorithm, numerical results by our algorithm will be compared with the ones by the data-driven model-free adaptive control (MFAC) algorithm [7]. According to [7], the controller is designed by

$$u_k = u_{k-1} + \frac{\rho \Phi_k (\tau - x_k)}{\lambda + \Phi_k^2},$$

and

$$\Phi_k = \Phi_{k-1} + \frac{\eta (\Delta x_k - \Phi_{k-1} \Delta u_{k-1}) \Delta u_{k-1}}{\mu + (\Delta u_{k-1})^2},$$

where $\eta = \rho = \mu = 1$, $\lambda = 0.5$. Let Φ_0 be initialized by an arbitrary positive-definite matrix. The corresponding state and control trajectories are shown in Fig. 14.5c, d, respectively.

From the numerical results, we can see that using the present stable iterative ADP algorithm, it takes 25 time steps for the system state to track the desired one. By MFAC algorithm in [7], it takes 50 iteration steps for the system state to track the desired one. Furthermore, there exist overshoots by the method of [7], while the overshoots are avoided by the present stable iterative ADP algorithm. These illustrate the effectiveness of our algorithm.

14.6 Conclusions

In this chapter, an effective data-driven stable iterative ADP algorithm is established to solve optimal temperature control problems for WGS systems. Using the WGS system data, NNs are used to approximate the system model and the reference control, respectively. The stable iterative ADP algorithm is established to obtain the optimal control law where the approximation errors of NNs and the disturbances are both considered. The convergence and stability properties are analyzed.

References

1. Alonso-Martinez J, Eloy-Garcia J, Santos-Martin D, Arnaltes S (2013) A new variable-frequency optimal direct power control algorithm. *IEEE Trans Ind Electron* 60(4):1442–1451
2. Andrikopoulos G, Nikolakopoulos G, Arvanitakis I, Manesis S (2014) Piecewise affine modeling and constrained optimal control for a pneumatic artificial muscle. *IEEE Trans Ind Electron* 61(2):904–916
3. Baier T, Kolb G (2007) Temperature control of the water gas shift reaction in microstructured reactors. *Chem Eng Sci* 62(17):4602–4611
4. Barros JD, Silva JFA, Jesus EGA (2013) Fast-predictive optimal control of NPC multilevel converters barros. *IEEE Trans Ind Electron* 60(2):619–627
5. Do TD, Choi HH, Jung JW (2012) SDRE-based near optimal control system design for PM synchronous motor. *IEEE Trans Ind Electron* 59(11):4063–4074
6. Falco MD, Piemonte V, Basile A (2012) Performance assessment of water gas shift membrane reactors by a two-dimensional model. *Comput Aided Chem Eng* 31:610–614
7. Hou Z, Jin S (2011) Data-driven model-free adaptive control for a class of MIMO nonlinear discrete-time systems. *IEEE Trans Neural Netw* 22(12):2173–2188
8. Huang Y, Liu D, Wei Q (2012) Temperature control in water-gas shift reaction with adaptive dynamic programming. In: Proceedings of the international symposium on neural networks, pp 478–487
9. Jing X, Cheng L (2013) An optimal PID control algorithm for training feedforward neural networks. *IEEE Trans Ind Electron* 60(6):2273–2283
10. Khalil HK (2002) Nonlinear syst. Prentice-Hall, Upper Saddle River
11. Kim GY, Mayor JR, Ni J (2005) Parametric study of microreactor design for water gas shift reactor using an integrated reaction and heat exchange model. *Chem Eng J* 110(1):1–10
12. Lewis FL, Liu D (2012) Reinforcement learning and approximate dynamic programming for feedback control. Wiley, Hoboken
13. Lincoln B, Rantzer A (2006) Relaxing dynamic programming. *IEEE Trans Automa Control* 51(8):1249–1260

14. Liu D, Wei Q (2013) Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems. *IEEE Trans Cybern* 43(2):779–789
15. Liu D, Wei Q (2014) Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Trans Neural Netw Learn Syst* 25(3):621–634
16. Lu X, Wang T (2013) Watergas shift modeling in coal gasification in an entrained-flow gasifier. Part 1: development of methodology and model calibration. *Fuel* 108:629–638
17. Moe JM (1962) Design of water gas shift reactors. *Chem Eng Prog* 58(3):33–36
18. Olalla C, Queinnec I, Leyva R, Aroudi AE (2012) Optimal state-feedback control of bilinear DC-DC converters with guaranteed regions of stability. *IEEE Trans Ind Electro* 59(10):3868–3880
19. Rathore R, Holtz H, Boller T (2013) Generalized optimal pulselwidth modulation of multilevel inverters for low-switching-frequency control of medium-voltage high-power industrial AC drives. *IEEE Trans Ind Electron* 60(10):4215–4224
20. Si J, Wang YT (2001) Online learning control by association and reinforcement. *IEEE Trans Neural Netw* 12(2):264–276
21. Sudhakar M, Narasimhan S, Kaisare NS (2012) Approximate dynamic programming based control for water gas shift reactor. *Comput Aided Chem Eng* 31:340–344
22. Ueyama Y, Miyashita E (2014) Optimal feedback control for predicting dynamic stiffness during arm movement. *IEEE Trans Ind Electron* 61(2):1044–1052
23. Wei Q, Liu D (2012) An iterative ϵ -optimal control scheme for a class of discrete-time nonlinear systems with unfixed initial state. *Neural Netw* 32:236–244
24. Wei Q, Liu D (2013) Numerical adaptive learning control scheme for discrete-time nonlinear systems. *IET Control Theory Appl* 7(11):1472–1486
25. Wei Q, Liu D (2013) A novel iterative θ -adaptive dynamic programming for discrete-time nonlinear systems. *IEEE Trans Automa Sci Eng* 11(4):1176–1190
26. Wei Q, Liu D (2014) Data-driven neuro-optimal temperature control of water-gas shift reaction using stable iterative adaptive dynamic programming. *IEEE Trans Ind Electron* 61(11):6399–6408
27. Wei Q, Wang D, Zhang D (2013) Dual iterative adaptive dynamic programming for a class of discrete-time nonlinear systems with time-delays. *Neural Comput Appl* 23(7–8):1851–1863
28. Wright GT, Edgar TF (1989) Adaptive control of a laboratory water-gas shift reactor with dynamic inlet condition. In: Proceedings of the American control conference, pp 1828–1833
29. Xiao S, Li Y (2013) Optimal design, fabrication, and control of an micropositioning stage driven by electromagnetic actuators. *IEEE Trans Ind Electron* 60(10):4613–4626
30. Yang Q, Jagannathan S (2012) Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators. *IEEE Trans Syst Man Cybern Part B Cybern* 42(2):377–390
31. Yin S, Ding SX, Haghani A, Hao H, Zhang P (2012) A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process. *J Process Control* 22(9):1567–1581
32. Yin S, Luo H, Ding SX (2014) Real-time implementation of fault-tolerant control systems with performance optimization. *IEEE Trans Ind Electron* 61(5):2402–2411
33. Zhang H, Wei Q, Liu D (2011) An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games. *Automatica* 47(1):207–214
34. Zhang H, Wei Q, Luo Y (2008) A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm. *IEEE Trans Syst Man Cybern Part B Cybern* 38(4):937–942

Index

A

- Action network, 10, 11, 59, 102, 161
- Action-dependent, 13, 259
- Action-dependent heuristic dynamic programming, 483, 484, 489
- Action-value function, or state-action value function, 3, 6
- Activation function, 50, 269, 275, 287, 294, 302, 316, 324, 329, 334, 340, 351, 396
- Actor–critic architecture, 316
- Adaptive critic designs, 2, 9, 10, 22
- Adaptive dynamic programming, 2, 7, 9, 10, 25, 26, 37
- Admissibility property, 180, 203
- Admissible control, 39, 80, 92, 134, 142, 152–154, 156, 157, 179, 183, 189, 192, 225, 232, 247, 251, 252, 350, 366, 390, 395, 402, 459, 460, 470, 523, 524
- Affine nonlinear systems, 39, 267, 286, 302, 431
- Algebraic Riccati equation, 163, 193
- AlphaGo, 1
- Approximate dynamic programming, 9, 10, 23
- Approximate optimal control, 267
- Approximate optimistic policy iteration, 237, 239, 243
- Approximate policy iteration, 231, 235, 259
- Approximate value iteration, 226, 228
- Approximation error, 91, 94, 96, 224
- Asymptotic dynamic programming, 10
- Asymptotically stable, 47, 48, 79

B

- Backpropagation, 12, 22, 49, 100, 273, 329, 503, 530, 540
- Backward-in-time approach, 12, 13
- Battery \mathbb{N} , 516, 526
- Battery model, 487, 496, 514
- Battery storage system, 484, 487
- Bellman equation, or Bellman optimality equation, 4, 9, 15, 39, 52, 92, 152, 178, 225, 247, 248, 497, 515, 577
- Bellman residual error, 317, 318
- Bellman’s principle of optimality, 9, 18, 52, 92, 152, 178, 225, 247, 515, 550, 577
- Brain-like intelligence, 22

C

- Cauchy–Schwarz inequality, 313, 542
- Coal combustion reaction, 538
- Coal gasification process, 538, 539
- Coal quality function, 538, 543, 544
- Constrained inputs, 56, 291, 298, 310
- Convergence criterion, 133, 211
- Cost function, 8, 15, 496, 497, 514
- Cost-to-go, 8, 489
- Critic architecture, 294
- Critic network, 10, 11, 57, 58, 101, 161, 333
- Curse of dimensionality, 9, 18

D

- Decentralized control, 388
- Decentralized stabilization, 389
- Deep learning, 1, 2

DeepMind, 1
 Discount factor, 3, 8, 223
 Discounted optimal control, 247
 Distributed iterative adaptive dynamic programming, 484, 515, 519, 526
 Disturbance policy, 419
 Dual heuristic dynamic programming, 13, 57, 59
 Dual iterative Q-learning, 483, 496, 497, 504, 505
 Dynamic programming, 2, 8, 9

E

Eligibility trace, 6
 Error bound, 96, 98, 226, 231, 236, 240, 249, 254, 255, 257, 551, 578
 Euclidean norm, 268, 271, 331
 Exploratory signal, 277, 288, 297, 302, 324, 325, 336, 343, 405, 411

F

Finite approximation errors, 91, 125
 Finite-step policy evaluation, 188
 Forward-in-time approach, 12
 Fréchet derivative, 461
 Frobenius matrix norm, 268, 271, 286, 288, 331

G

Gâteaux derivative, 461
 Game algebraic Riccati equation, 419
 General value iteration, 38, 51, 139
 Generalized Bellman equation, 153, 179, 180, 204
 Generalized policy iteration, 179, 191, 196, 214, 223
 Global convergence criterion, 208
 Global convergence property, 522
 Globalized dual heuristic programming, 13, 38, 57, 58
 Gradient descent algorithm, 276, 296, 297, 317, 336, 573
 Guaranteed cost control, 360

H

Hamilton–Jacobi–Bellman equation, 18, 19, 267, 269, 274, 292, 294, 311, 315, 316, 333, 348, 365, 391, 550
 Hamilton–Jacobi–Isaacs equation, 432, 437–440, 443, 450

Hamiltonian, 225, 274, 276, 292, 295, 310, 333, 335, 348, 351, 352, 365, 369, 370, 391, 396, 397, 433, 460, 469
 Heuristic dynamic programming, 13, 49, 59
 Home energy management system, 515
 Hurwitz matrix, 269, 270, 328
 Hyperbolic tangent function, 57, 269, 293, 311, 315, 323, 329, 340

I

Identification error, 312
 Identifier-critic architecture, 269
 Infinite horizon cost function, 39, 247, 347, 361, 419, 459
 Initial admissible control law, 160, 161
 Initial stabilizing control, 223
 Input constraints, 38, 56, 87
 Integral policy iteration, 401, 418, 420, 421
 Integral Q-learning, 423
 Interconnected nonlinear systems, 388
 Isolated subsystems, 390, 394, 400, 410
 Iteration flowchart, 42
 Iterative θ -ADP algorithm, 80, 83, 93
 Iterative adaptive dynamic programming, 15
 Iterative control law, 16, 69, 71, 72, 79, 94, 99, 107, 108, 110, 116, 117, 119, 124, 126, 135
 Iterative value function, 16, 46, 54–56, 65, 75, 93, 94, 96, 98, 99, 105, 107, 110, 117, 123, 127, 129, 132, 134, 138, 502

K

Kronecker product, 426

L

Lagrange stability, 48, 273, 286, 301, 323, 332, 340, 378
 Large-scale systems, 388, 403, 404
 Levenberg–Marquardt algorithm, 38, 49, 243, 336
 Lipschitz constant, 115–117, 120
 Lipschitz continuous, 39, 92, 111, 152, 178, 224, 247, 268, 292, 310, 347, 361, 390, 459
 Load profile, 485, 486, 493
 Local convergence criterion, 199, 203
 Local convergence property, 520
 Lyapunov equation, 270, 292, 310, 347, 348, 350, 351, 365, 390, 395, 396, 402, 424, 460

Lyapunov function, 47, 48, 80, 81, 110, 154, 204, 270, 274, 282, 293, 298, 313, 319, 330, 337, 353, 359, 366, 373, 393, 398–400, 424, 445, 467, 470, 542, 545, 556

Lyapunov's extension theorem, 48, 273, 286, 301, 323, 332, 340, 354, 378

M

Mean value theorem, 548
Model network, 10, 11, 57
Monte Carlo tree search, 1
Moore-Penrose pseudoinverse, 330, 331
Multi-battery coordination control, 513
Multi-player nonzero-sum games, 459
Multi-player zero-sum games, 431

N

Near optimal control, 51
Neural dynamic programming, 10
Neural network implementation, 48, 100, 241, 257, 350, 396, 425, 444, 464, 503, 554, 582
Neural network observer, 328
Neuro-dynamic programming, 10, 22
Nonaffine nonlinear systems, 52, 223, 291, 309, 310, 327
Numerical control, 107, 111
Numerical iterative θ -adaptive dynamic programming, 107, 120
Numerical iterative θ -ADP algorithm, 107, 111, 120

O

Observer-critic architecture, 309
Off-policy, 6
On-policy, 6
One-step policy evaluation, 41
Online optimal control, 275, 294
Optimal adaptive control, 25
Optimal battery charge/discharge strategy, 484
Optimal tracking control, 52, 223
Optimistic policy iteration, 237

P

Partial differential equation, 267, 274, 293
Particle swarm optimization, 507, 530
Performance index, 8, 419, 489

Persistence of excitation, 277, 279, 283, 288,

297, 302, 318, 336, 343

Policy evaluation, 5, 153, 180, 191, 214, 232

Policy improvement, 5, 41, 154, 180, 190,

191, 194, 214, 227, 232

Policy iteration, 5, 37, 151, 153, 154, 160,

162, 173, 223, 231, 249

Policy update, 41, 154, 236, 240, 255

Q

$Q(\lambda)$, 7
Q-function, 247, 248, 497, 498, 512, 530
Q-learning, 6

R

Reaction temperature, 539, 557, 571, 572, 585

Reference control, 546, 547, 574

Reinforcement learning, 2, 3, 10, 23–25

Relaxed dynamic programming, 10

Residential energy system, 484, 489, 491, 496, 513

Residual error, 275, 276

Robust control, 346

Robust guaranteed cost control, 366, 367, 372

S

Sarsa, 6
 $Sarsa(\lambda)$, 7
Single critic approach, 490, 491
Snyder International, 21
State-action value function, 248
State-value function, 3–5
Steepest descent algorithm, 397
Suboptimal control, 19

T

$TD(\lambda)$, 6
TD-Gammon, 2
Temporal difference, 5
 θ -adaptive dynamic programming, 67, 80, 92, 107
Time-based Q-learning algorithm, 507, 530
Torsional pendulum system, 166
Two-player zero-sum games, 418

U

Uncertain nonlinear systems, 346, 360

Uniformly ultimately bounded, 268, 282, 286, 292, 298, 323
Utility function, 17, 18, 39, 40, 52, 56, 57, 68, 550

V

Value function, 3, 15, 268, 274, 292, 327, 400, 419, 459

Value function update, 5, 41, 53, 188, 227
Value iteration, 5, 37, 40, 223, 227, 248, 310

W

Water gas shift reaction, 572