

AUTOMATION AND CONTROL ENGINEERING

A Series of Reference Books and Textbooks

Series Editors

**FRANK L. LEWIS, PH.D.,
FELLOW IEEE, FELLOW IFAC**

Professor
Automation and Robotics Research Institute
The University of Texas at Arlington

**SHUZHI SAM GE, PH.D.,
FELLOW IEEE**

Professor
Interactive Digital Media Institute
The National University of Singapore

1. Nonlinear Control of Electric Machinery, *Darren M. Dawson, Jun Hu, and Timothy C. Burg*
2. Computational Intelligence in Control Engineering, *Robert E. King*
3. Quantitative Feedback Theory: Fundamentals and Applications, *Constantine H. Houpis and Steven J. Rasmussen*
4. Self-Learning Control of Finite Markov Chains, *A. S. Poznyak, K. Najim, and E. Gómez-Ramírez*
5. Robust Control and Filtering for Time-Delay Systems, *Magdi S. Mahmoud*
6. Classical Feedback Control: With MATLAB®, *Boris J. Lurie and Paul J. Enright*
7. Optimal Control of Singularly Perturbed Linear Systems and Applications: High-Accuracy Techniques, *Zoran Gajic and Myo-Taeg Lim*
8. Engineering System Dynamics: A Unified Graph-Centered Approach, *Forbes T. Brown*
9. Advanced Process Identification and Control, *Enso Ikonen and Kaddour Najim*
10. Modern Control Engineering, *P. N. Paraskevopoulos*
11. Sliding Mode Control in Engineering, *edited by Wilfrid Perruquetti and Jean-Pierre Barbot*
12. Actuator Saturation Control, *edited by Vikram Kapila and Karolos M. Grigoriadis*
13. Nonlinear Control Systems, *Zoran Vukić, Ljubomir Kuljaāa, Dali Donlagiā, and Sejid Tesnjak*
14. Linear Control System Analysis & Design: Fifth Edition, *John D'Azzo, Constantine H. Houpis and Stuart Sheldon*
15. Robot Manipulator Control: Theory & Practice, Second Edition, *Frank L. Lewis, Darren M. Dawson, and Chaouki Abdallah*
16. Robust Control System Design: Advanced State Space Techniques, Second Edition, *Chia-Chi Tsui*
17. Differentially Flat Systems, *Hebertt Sira-Ramirez and Sunil Kumar Agrawal*

18. Chaos in Automatic Control, *edited by Wilfrid Perruquetti and Jean-Pierre Barbot*
19. Fuzzy Controller Design: Theory and Applications, *Zdenko Kovacic and Stjepan Bogdan*
20. Quantitative Feedback Theory: Fundamentals and Applications, Second Edition, *Constantine H. Houpis, Steven J. Rasmussen, and Mario Garcia-Sanz*
21. Neural Network Control of Nonlinear Discrete-Time Systems, *Jagannathan Sarangapani*
22. Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications, *edited by Shuzhi Sam Ge and Frank L. Lewis*
23. Hard Disk Drive: Mechatronics and Control, *Abdullah Al Mamun, GuoXiao Guo, and Chao Bi*
24. Stochastic Hybrid Systems, *edited by Christos G. Cassandras and John Lygeros*
25. Wireless Ad Hoc and Sensor Networks: Protocols, Performance, and Control, *Jagannathan Sarangapani*
26. Modeling and Control of Complex Systems, *edited by Petros A. Ioannou and Andreas Pitsillides*
27. Intelligent Freight Transportation, *edited by Petros A. Ioannou*
28. Feedback Control of Dynamic Bipedal Robot Locomotion, *Eric R. Westervelt, Jessy W. Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris*
29. Optimal and Robust Estimation: With an Introduction to Stochastic Control Theory, Second Edition, *Frank L. Lewis; Lihua Xie and Dan Popa*
30. Intelligent Systems: Modeling, Optimization, and Control, *Yung C. Shin and Chengying Xu*
31. Optimal Control: Weakly Coupled Systems and Applications, *Zoran Gajić, Myo-Taeg Lim, Dobrila Škatarić, Wu-Chung Su, and Vojislav Kecman*

Intelligent Systems

Modeling, Optimization, and Control

Yung C. Shin

*Purdue University
West Lafayette, Indiana, U.S.A.*

Chengying Xu

*University of Central Florida
Orlando, Florida, U.S.A.*



CRC Press

Taylor & Francis Group

Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

© 2009 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works
Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-13: 978-1-4200-5176-6 (Hardcover)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging-in-Publication Data

Shin, Yung C.

Intelligent systems : modeling, optimization, and control / authors, Yung C. Shin and Chengying Xu.

p. cm. -- (Automation and control engineering)

Includes bibliographical references and index.

ISBN 978-1-4200-5176-6 (alk. paper)

1. Soft computing. 2. Expert systems (Computer science) 3. Intelligent control systems. I. Xu, Chengying. II. Title.

QA76.9.S63S55 2009

006.3--dc22

2008029322

Visit the Taylor & Francis Web site at
<http://www.taylorandfrancis.com>

and the CRC Press Web site at
<http://www.crcpress.com>

Dedication

To our families

Contents

Preface

Acknowledgments

Authors

Chapter 1 Intelligent Systems

- 1.1 Introduction
- 1.2 Introduction of Soft Computing Techniques
 - 1.2.1 Neural Networks
 - 1.2.2 Fuzzy Logic
 - 1.2.3 Evolutionary Algorithms
- 1.3 Summary

References

Chapter 2 Modeling of Nonlinear Systems: Fuzzy Logic, Neural Networks, and Neuro-Fuzzy Systems

- 2.1 Fuzzy Systems
 - 2.1.1 Fuzzy Sets
 - 2.1.2 Fuzzy Operations
 - 2.1.3 Membership Functions
 - 2.1.3.1 Triangular Membership Function
 - 2.1.3.2 Trapezoidal Membership Function
 - 2.1.3.3 Gaussian Membership Function
 - 2.1.3.4 Generalized Bell Membership Function
 - 2.1.3.5 Sigmoidal Membership Function
 - 2.1.3.6 Z-Shaped Membership Function
 - 2.1.4 Fuzzy Relations
 - 2.1.5 Fuzzy Inference System
 - 2.1.5.1 Fuzzifier
 - 2.1.5.2 Fuzzy Rule Base
 - 2.1.5.3 Fuzzy Inference Engine
 - 2.1.5.4 Defuzzifier
- 2.2 Artificial Neural Networks
 - 2.2.1 Basic Structure
 - 2.2.2 Multilayer Feedforward Neural Networks
(Backpropagation Neural Networks)
 - 2.2.3 Radial Basis Function Networks
 - 2.2.3.1 Definition and Types of RBF

- 2.2.4 Recurrent Neural Networks
 - 2.2.4.1 Introduction
 - 2.2.4.2 Network Architecture
 - 2.2.4.3 Structure and Parameter Learning
 - 2.2.4.4 Other Issues
- 2.3 Neuro-Fuzzy Systems
 - 2.3.1 Fuzzy Basis Function Networks
 - 2.3.2 ANFIS
- 2.4 Modeling of Dynamic Systems
 - 2.4.1 Dynamic System Identification Using Feedforward Networks
 - 2.4.1.1 Dynamic System Modeling by Radial Basis Function Neural Network
 - 2.4.2 Dynamic System Representation by Recurrent Neural Networks
 - 2.4.3 State Observer Construction
 - 2.4.3.1 State Estimation Using RBFNN
 - 2.4.3.2 Example Applications of the RBFNN State Estimator
- 2.5 Conclusions
- References

Chapter 3 Efficient Training Algorithms

- 3.1 Supervised Algorithm
- 3.2 Unsupervised Algorithm
- 3.3 Backpropagation Algorithm
- 3.4 Dynamic Backpropagation
- 3.5 Orthogonal Least Squares Algorithm
- 3.6 Orthogonal Least Square and Generic Algorithm
 - 3.6.1 OLS Learning Using Genetic Algorithm
 - 3.6.2 Determination of the Number of Hidden Nodes
 - 3.6.3 Performance Evaluation
- 3.7 Adaptive Least-Squares Learning Using GA
 - 3.7.1 Adaptive Least-Squares Learning Using GA
 - 3.7.2 Extension of ALS Algorithm to Multi-Input, Multi-Output Systems
 - 3.7.3 Performance Evaluation in Approximating Nonlinear Functions
 - 3.7.4 Application of FBFN to Modeling of Grinding Processes

References

Chapter 4 Fuzzy Inverse Model Development

- 4.1 Fuzzy Inverse Model Development
- 4.2 Simulation Examples

- 4.2.1 Two-Link Robot Manipulator
- 4.2.2 Five-Link AdeptOne Industry Robot Manipulator
- 4.2.3 Four-Link AdeptOne Industry Robot Manipulator

4.3 Conclusion

References

Chapter 5 Model-Based Optimization

- 5.1 Model Building
- 5.2 Model-Based Forward Optimization
 - 5.2.1 Formulation of the Problem
 - 5.2.2 Optimization Algorithm
 - 5.2.2.1 Standard Evolutionary Strategies for Continuous Variables
 - 5.2.2.2 Handling of Discrete Variables
 - 5.2.2.3 Handling of Constraints
 - 5.2.2.4 Algorithm
- 5.3 Application of ES to Numerical Examples
- 5.4 Application of Model-Based Optimization Scheme to Grinding Processes
 - 5.4.1 Application to Creep Feed Grinding Example
 - 5.4.2 Application to Surface Grinding Example

References

Chapter 6 Neural Control

- 6.1 Supervised Control
- 6.2 Direct Inverse Control
- 6.3 Model Reference Adaptive Control
- 6.4 Internal Model Control
- 6.5 Model Predictive Control
- 6.6 Feedforward Control

References

Chapter 7 Fuzzy Control

- 7.1 Knowledge-Based Fuzzy Control
 - 7.1.1 Fuzzy PID Control
 - 7.1.2 Hybrid Fuzzy Control
 - 7.1.3 Supervisory Fuzzy Control
 - 7.1.4 Self-Organizing Fuzzy Control
 - 7.1.5 Fuzzy Model Reference Learning Control
- 7.2 Model-Based Fuzzy Control
 - 7.2.1 Fuzzy Inverse Control
 - 7.2.2 Fuzzy Inverse Control for a Singleton Fuzzy Model

7.2.3 Fuzzy Model-Based Predictive Control

7.2.4 Fuzzy Internal Model Control

References

Chapter 8 Stability Analysis Method

8.1 Lyapunov Stability Analysis

8.1.1 Mathematical Preliminaries

8.1.2 Lyapunov's Direct Method

8.1.3 Lyapunov's Indirect Method

8.1.4 Lyapunov's Method to the TS Fuzzy Control System

8.1.5 Stability Concepts for Nonautonomous Systems

8.1.5.1 Lyapunov's Direct Method

8.1.5.2 Lyapunov's Indirect Method

8.2 Passivity Approach

8.2.1 Passivity Concept

8.2.1.1 Continuous-Time Case

8.2.1.2 Discrete-Time Case

8.2.2 Sectorial Fuzzy Controller

8.2.2.1 Inputs

8.2.2.2 Rule Base

8.2.2.3 Output

8.2.3 Property of Sectorial Fuzzy Controller

8.2.4 Passivity of Sectorial Fuzzy Controller in Continuous Domain

8.2.5 Passivity of Sectorial Fuzzy Controller in Discrete Domain

8.3 Conclusion

References

Chapter 9 Intelligent Control for SISO Nonlinear Systems

9.1 Fuzzy Control System Design

9.1.1 First Layer Fuzzy Controller

9.1.2 Self-Organizing Fuzzy Controller

9.1.3 Online Scaling Factor Determination Scheme

9.2 Stability Analysis

9.2.1 Multilevel Fuzzy Control Structure

9.2.2 Stability Analysis in Continuous-Time Case

9.2.3 Stability Analysis in Discrete-Time Case

9.3 Simulation Examples

9.3.1 Cargo Ship Steering

9.3.2 Fuzzy Cruise Control

9.3.3 Water Level Control

9.4 Implementation—Force Control for Grinding Processes

9.4.1 Hardware Configuration

9.4.2 Monitoring and Workpiece Setup

- 9.4.3 Experimental Implementation Results
- 9.4.4 Wheel Wear Experiments
- 9.5 Simulation and Implementation—Force Control for Milling Processes
 - 9.5.1 Simulation Examples
 - 9.5.2 Experimental Setup—Hardware Configuration
 - 9.5.3 Experimental Implementation Results
- 9.6 Conclusion

References

Chapter 10 Intelligent Control for MISO Nonlinear Systems

- 10.1 MLFC-MISO Control System Structure
 - 10.1.1 Control Parameters Initialization
 - 10.1.2 Fuzzy Adaptive PD–PI Controller
- 10.2 Stability Analysis
- 10.3 Simulation Examples
 - 10.3.1 Magnetic Bearing System
 - 10.3.2 Fed-Batch Reactor
- 10.4 Conclusion

References

Chapter 11 Knowledge-Based Multivariable Fuzzy Control

- 11.1 Complexity Reduction Methods
 - 11.1.1 Rule Base Simplification
 - 11.1.2 Dimensionality Reduction
 - 11.1.3 Structured Systems
- 11.2 Methods to Optimize Multivariable Fuzzy Inferencing Calculation
 - 11.2.1 Intersection Coefficients
 - 11.2.2 Decomposition of a Multidimensional Fuzzy Rule Base
 - 11.2.3 Simplification of a Multidimensional Fuzzy Rule Base
- 11.3 Multivariable Fuzzy Controller to Deal with the Cross-Coupling Effect
 - 11.3.1 Mixed Fuzzy Controller
 - 11.3.2 Multiobjective Fuzzy Controller
- 11.4 Conclusion

References

Chapter 12 Model-Based Multivariable Fuzzy Control

- 12.1 Fuzzy Model of Multivariable Systems
- 12.2 Multivariable Interaction Analysis
 - 12.2.1 Relative Gain Array
 - 12.2.1.1 Relative Gain Array for Square Systems
 - 12.2.1.2 Relative Gain Array for Nonsquare Systems

- 12.2.2 Interaction Analysis in Multivariable Fuzzy Models
 - 12.2.3 Simulation Examples
 - 12.2.4 Conclusion
 - 12.3 Multivariable Fuzzy Control Design
 - 12.3.1 Horizontal Fuzzy Control Engine
 - 12.3.2 Perpendicular Fuzzy Control Engine
 - 12.4 Stability Analysis
 - 12.5 Simulation Examples
 - 12.5.1 Distillation Column
 - 12.5.2 Chemical Pressure Tank System
 - 12.6 Conclusion
- References

Preface

In recent years, there has been a dramatic increase in the interest and use of various soft computing techniques for scientific and engineering applications. “Intelligent systems” is a very broad term, which covers approaches to design, optimization, and control of various systems without requiring mathematical models, in a way similar to how humans work, and typically involves many fields such as neural networks, fuzzy logic, evolutionary strategy, and genetic algorithm, and their hybrids and derivatives. A number of books have been written on various disciplines of these soft computing techniques. However, most of them focus only on certain areas of soft computing techniques and applications. Effective intelligent systems can be constructed by combining appropriate soft computing techniques based on the problems to be solved. Thus, the purpose of this book is to show how to use these various disciplines in an integrated manner in realizing intelligent systems.

This book is dedicated to providing the highlights of current research in the theory and applications of these soft computing techniques in constructing intelligent systems. It is unique in the sense that it concentrates on building intelligent systems by combining methods from diverse disciplines. We intend to clearly describe the theoretical and practical aspects of these systems. This book focuses on various approaches based on different soft computing techniques developed by the authors and others to modeling of nonlinear systems, optimization, and control of various engineering problems. It gives a thorough coverage of the entire field for an advanced college-level course or at the graduate level. It should also be very useful as a reference book for industrial researchers and practitioners who want to implement intelligent systems.

The book begins with an introduction to the field of various soft computing techniques, including neural networks, fuzzy logic, and evolutionary computing techniques. Chapter 2 covers various neuro-fuzzy schemes and their applications to modeling of nonlinear systems. Chapter 3 presents different training algorithms used for various neuro-fuzzy systems, and also features a practical application example on modeling of grinding processes. Chapter 4 describes the novel inverse model construction process based on the forward fuzzy model established from input-output data and illustrates its effectiveness with inverse-kinematic solutions of multi-degree-of-freedom robot arms. Chapter 5 presents an effective optimization technique that can handle constrained mixed integer problems, which are known to be most difficult among optimization problems, based on extended evolutionary strategies. It also shows various examples on optimal mechanical system design and optimization of complex manufacturing processes. Chapters 6 through 12 deal with control system design. Chapter 6 presents an overview of different neural control schemes that can be used when mathematical models are not available. Chapter 7 describes another class of rule-based intelligent controllers using fuzzy rules and logic. Chapter 8 provides two important stability theories that can be used in constructing stable, intelligent controllers. Chapter 9 presents a stable adaptive

fuzzy controller that can be applied to a large class of single-input, single-output nonlinear systems. It describes the design methodology, stability analysis, and various illustrative simulation examples including cargo ship steering, cruise control, water level control, as well as experimental applications to manufacturing processes. In Chapter 10, the multi-input single-output control design technique is presented by extending the fuzzy control scheme described in Chapter 9 with the inverse modeling procedure described in Chapter 4. Chapters 11 and 12 are devoted to intelligent multivariate control system techniques. Chapter 11 provides an overview of various knowledge-based intelligent control techniques, while Chapter 12 describes the model-based intelligent control design methodology for multi-input multi-output systems.

Overall, the book shows the issues encountered in the development of applied intelligent systems and describes a wide range of intelligent system design techniques. While it is nearly impossible to include all different techniques in one book, our goal was to cover key elements in developing different applications of intelligent systems. Throughout this book, the theory and algorithms are illustrated by simulation examples, as well as practical experimental results. We have tried to describe each technique in sufficient detail so that readers can develop intelligent systems for real applications.

**Y.C. Shin
C. Xu**

Acknowledgments

We are thankful for the current and former students of our group, who have contributed to the development of material included in this book. Our thanks go to Professor C. Lee, and P. Vishnupad, N. Subrahmanya, and T. Davis, who have provided some materials.

Authors



Yung C. Shin received his PhD in mechanical engineering from the University of Wisconsin, Madison, Wisconsin in 1984, and is currently a professor of mechanical engineering at Purdue University, West Lafayette, Indiana. Before taking up his current designation he worked as a senior project engineer at the GM Technical Center from 1984 to 1988 and as faculty at the Pennsylvania State University from 1988 to 1990.

His research areas include intelligent and adaptive control, process monitoring and diagnostics, laser processing of materials, high speed machining, process modeling, and simulation. He has published over 200 papers in archived journals and refereed conference proceedings and has authored chapters in several engineering handbooks. He has coedited two books and has coauthored *Intelligent Systems: Modeling, Optimization and Control* (CRC Press, 2008). He has organized or co-organized many conferences and symposia in his areas of research including the first and second Artificial Neural Networks in Engineering Conference, and Symposium on Neural Networks in Manufacturing and Robotics and Symposium on Intelligent Design and Manufacturing at the ASME Winter Annual Meetings.



Chengying Xu is currently an assistant professor at the University of Central Florida, Orlando, Florida. She received her PhD in 2006 in mechanical engineering from Purdue University, West Lafayette, Indiana, and her MS in 2001 in mechanical manufacturing and automation from Beijing University of Aeronautics and Astronautics, Beijing, China. Her research interests include intelligent systems, control, manufacturing, system dynamics, online monitoring and diagnostics, robotics, mechatronics, and automation. She has authored and coauthored around 20 publications in archived journals and refereed conference proceedings. She has served as

an organizing committee member and session cochair for a number of national and international conferences, and has reviewed papers for the American Society of Mechanical Engineers (ASME) and the Institute of Electrical and Electronics Engineers (IEEE) transactions journals. She is an active member for several professional societies, such as the ASME, the IEEE, the Society of Manufacturing Engineers, and the Society for Experimental Mechanics. Dr. Xu has been an associate editor of the *International Journal of Nanomanufacturing* since 2008.

1 Intelligent Systems

1.1 INTRODUCTION

Advances in artificial intelligence (AI), soft computing, and related scientific fields have brought new opportunities and challenges for researchers to deal with complex and uncertain problems and systems, which could not be solved by traditional methods. Many traditional approaches that have been developed for mathematically well-defined problems with accurate models may lack in autonomy and decision-making ability and hence cannot provide adequate solutions under uncertain and fuzzy environments. Intelligent systems represent a new approach to addressing those complex problems with uncertainties. Intelligent systems are defined with such attributes as high degree of autonomy, reasoning under uncertainty, higher performance in a goal seeking manner, high level of abstraction, data fusion from a multitude of sensors, learning and adaptation in a heterogeneous environment, etc. (Shoureshi and Wormley, 1990).

In the real world, we can find many problems and systems that are too complex or uncertain to represent them in complete and accurate mathematical models. And yet, we still have the need to design, optimize, or control the behavior of such systems. As an example, humans can learn information through their cognitive process, make decisions, and even adapt to the dynamic environment without using mathematical models. The motivation for designing intelligent systems comes from the efforts to mimic such a process without requiring precise mathematical models. There are three major components for developing intelligent systems: knowledge acquisition and representation, inferencing, and decision-making processes.

We can consider three possible types of knowledge or information representation describing the input–output or cause–effect relationships of nonlinear systems. They are quantitative analytical models, data, and heuristic rules as shown in [Figure 1.1](#). When none of these information types can provide the pertinent complete and sufficient knowledge about the system behavior by itself alone, it will be necessary to combine all three heterogeneous domains of information. Therefore, we will need to consider different modeling strategies for capturing different types of information.

Neural networks (NNs) are suitable for representing the input–output relationships of nonlinear systems. Due to its smoothing and global approximation capabilities, if a sufficient amount of data is available, NNs can be trained to provide a suitable mapping between input and output variables. On the other hand, heuristic information can be captured by using fuzzy logic in the form of if-then rules. The uncertainty or fuzziness associated with each linguistic description is handled by using the so-called membership functions. By combining these fuzzy rules, we can construct a fuzzy basis function network similar to NNs. This concept will be further illustrated in [Chapter 2](#).

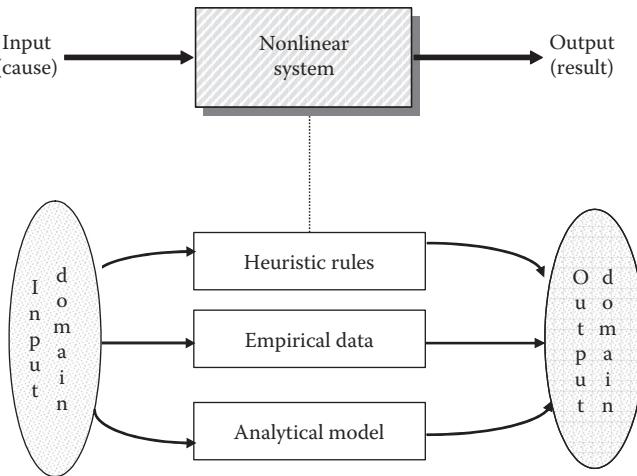


FIGURE 1.1 Three different knowledge representation schemes.

Examples of systems that would require and benefit from intelligent control and optimization include electric power plants and their distributed networks, military command and control systems, air traffic control systems, biological systems, and manufacturing systems. The goal of this book is to cover essential elements of knowledge and tools for developing intelligent systems and its application to complex and uncertain problems and systems.

Activities in the development of intelligent systems are scattered over several disciplines. In general, intelligent systems are based on expert systems, or soft computing techniques such as fuzzy logic, NNs, and evolutionary strategies. Applications of expert system programming techniques to assist in a control design process and in process monitoring and diagnostics are perhaps the most advanced area in intelligent systems. There have been a number of success stories in applications of expert systems including CRAFR, GEMS, GENAID, and TEXSYS (Shoureshi and Wormley, 1990). However, current applications use rule-based algorithms that are hand crafted, and typically lack the ability to add knowledge to their databases in autonomous manner or adapt to the dynamic change of environment. On the other hand, soft computing techniques provide potential for autonomous learning, systematic reasoning, and adaptability, and hence offer a promise for the construction of intelligent systems.

Since the mid-1980s, there has been a major explosion in the research related to NNs, fuzzy logic, and evolutionary strategies. The most notable investigations are by Werbos (1989) on back propagation, Widrow (1985) on formulation of Adaline, and Narendra and Parthasarathy (1990) on NNs for system identification and control. Another area that has potential for advancement of intelligent systems is fuzzy logic and fuzzy control. In contrast to classical logical systems, fuzzy logic is aimed at a formulation of models of reasoning that are approximate rather than exact. Fuzzy logic provides tolerance for imprecision. This characteristic has attracted attentions of researchers and engineers in various disciplines to design rule-based methodologies

for uncertain systems (Zadeh, 1973; Kosko, 1992). Therefore, we briefly discuss the basic concepts of these soft computing techniques. Evolutionary strategies provide optimal point search techniques via evolution and adaptation and are useful for the optimization of complex or ill-defined problems. The goal of this book is to cover essential knowledge and methodologies for developing intelligent systems and its application to design, optimization, and control of complex and uncertain problems and systems.

1.2 INTRODUCTION OF SOFT COMPUTING TECHNIQUES

In recent years, various soft computing based techniques have emerged as useful tools for solving various scientific and engineering problems that were not possible or convenient to handle by traditional methods. The soft computing techniques provide computationally efficient and yet effective means of modeling, analysis, and decision making of complex phenomena. In Wikipedia, soft computing is defined as “a collection of computational techniques in computer science, AI, machine learning and some engineering disciplines, which attempt to study, model, and analyze very complex phenomena: those for which more conventional methods have not yielded low cost, analytic, and complete solutions.” The typical techniques that belong to the soft computing arena include artificial neural networks (ANNs), fuzzy sets and systems, evolutionary computation including evolutionary strategies (ESs), swarm intelligence and harmony search, Bayesian network, chaos theory, etc. Much of these soft computing techniques are inspired by biological processes or are the results of attempts to emulate such processes.

Bäck (1996) provides further descriptions on soft computing:

Unlike hard computing schemes, which strive for exactness and full truth, soft computing techniques exploit the given tolerance of imprecision, partial truth, and uncertainty for a particular problem. In addition, soft computing offers computational efficiency in inferencing compared with conventional techniques. Another common contrast comes from the observation that inductive reasoning plays a larger role in soft computing than in hard computing.

These attributes of computational efficiency and its ability to deal with imprecision make soft computing very attractive techniques for dealing with complex, nonlinear, and uncertain systems, which do not lend themselves to precise mathematical expressions. In the next sections, the concepts of various commonly used soft computing techniques are briefly described.

1.2.1 NEURAL NETWORKS

Artificial neural networks, often just called neural networks, are a generic method of mapping or representing input and output relationships or patterns of nonlinear functions or data through a single or multiple layers of an interconnected group of artificial neurons ([Figure 1.2](#)). ANNs consist of processing nodes, called neurons, which collect incoming signals, process them, and then generate output, and links

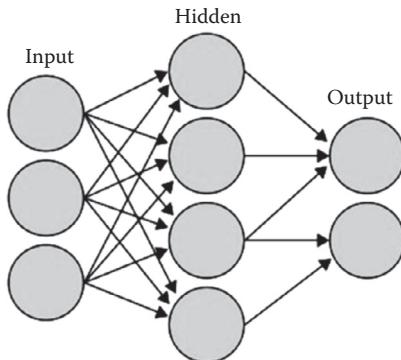


FIGURE 1.2 Illustration of ANN.

providing connections among the layers of neurons. The neurons usually process the incoming information via a certain activation function and generate output signal when the strength of the incoming signals exceeds a specified threshold, which is then passed to other neurons through connections. The neurons and connections among the adjacent layers of neurons provide the ANNs computational ability to approximate arbitrary–nonlinear relationships.

Artificial neural networks can be grouped into unsupervised and supervised networks depending on their training methods. Unsupervised networks refer to those NNs that do not require output data during learning or training such as adaptive resonance networks. Supervised networks such as multilayer feedforward NNs however require both input and output data during training of the networks. ANNs can also be grouped into static and dynamic networks depending on their structures. Static networks provide mapping between input and output data without any time-dependent connections between layers or neurons in the network structure. On the other hand, the dynamic networks have embedded feedback connections with delays and hence can provide a more compact structure for representing the dynamic relationships. For example, popularly used backpropagation NNs or radial basis function NNs belong to the former, while recurrent NNs belong to the latter.

1.2.2 Fuzzy Logic

Uncertainties or complexity of many physical systems or processes often lead to a difficulty in developing accurate analytical models. Nonetheless, humans can still describe the behavior of such systems through their cognitive processes, at least qualitatively. An expert might be able to control a process based on his knowledge and observation of the process even without any mathematical model. Fuzzy set theory is “a body of concepts and techniques that give a form of mathematical precision to human cognitive processes that are in many ways imprecise and ambiguous by the standards of classical mathematics” (Kaufmann and Gupta, 1988). In effect, this theory allows one to deal with fuzziness by grouping elements, which do not have clear boundaries, into different classes. Fuzzy logic uses fuzzy set membership functions, whose value range from 0 to 1, and allows for capturing linguistic representations of

knowledge. The imprecision of knowledge is handled by the membership functions associated with each linguistic variable. In a narrow sense, fuzzy logic is a logical system, which is a generalization of multivalued logic (Zadeh, 2007). Due to this reason, the fuzzy theory has been applied to various engineering problems that are too complex or ill-defined for the conventional mathematical analysis.

In a fuzzy system, knowledge is represented by if-then rules associated with fuzzy variables. These rules along with the membership functions are processed through the so-called compositional rule of inference. Unlike other reasoning processes where qualitative reasoning is used with pure linguistic rules, fuzzy inference logic involves numerical synthesis based on membership functions to form a fuzzy decision table. Since the quantities synthesized in a fuzzy reasoning procedure are generally fuzzy, the final decision is also fuzzy. Fuzzy inferencing is, therefore, usually called approximate reasoning; that is, it matches process quantities with the rules in the rule base to perform fuzzy inferencing by using the compositional rule of inference.

1.2.3 EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EAs) are computational algorithms that are inspired by the nature's evolutionary process. They emulate the principles of natural selection, which favor the stronger species and guide further evolution such that they survive in their environmental conditions. The primary focus of EAs is the search of global optimal point. EAs or evolutionary computations are therefore suitable for optimization problems with applications including, but not limited to, design optimization, optimal parameter learning of ANNs or fuzzy basis function networks, or optimal design of control parameters.

Evolutionary algorithms rely on the so-called subsymbolic, that is, numerical, representation of knowledge. Contrary to the early days of AI that emphasized the symbolic representations using predicate logic, semantic nets, or frames, EAs emulate evolutionary processes in computational forms. Typical EAs include ESs, evolutionary programming (EP), genetic algorithm (GA), and learning classifier systems. Among them, the genetic algorithms are the most popular EAs and have been actively used in various application areas. The three algorithms of ES, EP, and GA are similar in nature since all of them are developed from the same idea of applying the principle of organic evolution to optimization problems. However, they are quite different in implementation procedures such as in the representation scheme and self-adaptation abilities. For example, ESs use real vectors as coding representation, and primarily mutation and selection as search operators, while GAs use, at least initially, binary string representations and EP finite state machines. Please refer to Bäck (1996) for an extensive comparison of these three algorithms.

Despite the differences in specific approaches, EAs in general share some common basic features (Menon, 2004):

- Population of individuals
- Notion of fitness
- Notion of population dynamics biased by fitness
- Notion of inheritance of properties from parent to child

In the 1990s, interactions among the three different communities began to take place, and consequently many attempts have been made in each community to supplement its algorithm by adopting desirable components from other algorithms. Introduction of floating variables instead of binary strings to GA is one of those examples (Michalewicz, 1996). The adoption of uniform distribution for the mutation of integer variables in ES seems to have borrowed its idea from GA (Bäck and Schütz, 1995). The boundaries among the three algorithms are fading away and researchers are adopting representation schemes and operators from various EAs, where they are appropriate to their application in order to obtain the best possible results.

Contemporary derivatives of evolutionary computation include those such as genetic programming, ES using a population of μ parents and also recombination as an additional operator (called $(\mu/\rho+, \lambda)$ -ES), and compact GA, among many.

While not covered in this section, a notable evolutionary technique for optimization also includes swarm intelligence such as ant colony optimization and particle swarm optimization, which are known to work quite effectively for high-dimensional problems.

1.3 SUMMARY

As described in this chapter, intelligent systems can be designed by various soft computing techniques. Although it would be possible to adopt other techniques such as expert systems and symbolic AI, the focus of this book is to construct various intelligent system approaches based on soft computing techniques with applications to design optimization, modeling, and control of complex systems and processes. In the remainder of the book, readers will find the combined use of these soft computing techniques for constructing intelligent systems approaches.

REFERENCES

- Bäck, T., *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.
- Bäck, T. and Schütz, M., Evolution strategies for mixed-integer optimization of optical multilayer systems, *Evolutionary Programming IV: Proceedings of the 4th Annual Conference on Evolutionary Programming*, pp. 33–51, 1995.
- Kaufmann, A. and Gupta, M.M., *Fuzzy Mathematical Models in Engineering and Management Science*, Elsevier Science, New York, 1988.
- Kosko, B., *Neural Networks and Fuzzy Systems—A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall, Englewood Cliffs, NJ, 1992.
- Menon, A., *Frontiers of Evolutionary Computation*, Kluwer Academic Publishers, Boston/Dordrecht/London, 2004.
- Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed., Springer-Verlag, Reading, United Kingdom, New York, 1996.
- Narendra, K. and Parthasarathy, K., Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks*, 1(1), 4–27, March 1990.
- Shoureshi, R. and Wormley, D., Intelligent control systems, *Final Report of NSF/EPRI Workshop*, October, 1990.
- Werbos, P., Backpropagation and neural control: A review and prospectus, *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, New York, June 1989.

Widrow, B. and Steams, S.D., *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

Wikipedia, the free encyclopedia on web, <http://www.wikipedia.org/>.

Zadeh, L.A., Outline of new approach to the analysis of complex structures and decision processes, *IEEE Transaction on Systems, Man and Cybernetics*, SMC-3, 28–44, 1973.

Zadeh, L.A., From fuzzy logic to extended fuzzy logic—The concept of f-validity and the impossibility principle, *FUZZ-IEEE 2007*, Imperial College, London, United Kingdom, July 23, 2007.

2 Modeling of Nonlinear Systems: Fuzzy Logic, Neural Networks, and Neuro-Fuzzy Systems

2.1 FUZZY SYSTEMS

Fuzzy set theory was first proposed by L. A. Zadeh in 1965 as a way to characterize nonprobabilistic uncertainties. In this section, we introduce the principal concepts and mathematical notations of fuzzy set theory and fuzzy inference system. Various fuzzy set operations are explained with graphical illustrations. The basic concepts and operations on fuzzy relations are presented. The set-relation and relation-relation compositions and their usage in the compositional rule of inference are explained, which play an important role in various research areas such as fuzzy system analysis, design of fuzzy controllers, and decision-making processes. The basic architecture and the design methodology of a fuzzy inference system are presented with practical examples. The content is explained in clear notations to facilitate readers' easy reading and can be served as an introductory foundation to understanding various techniques on fuzzy modeling and control later in this book.

2.1.1 FUZZY SETS

A fuzzy set is a set that does not have a crisp boundary. That is, the transition from belonging to a set to not belonging to a set is gradual and is characterized by a membership function (MF). Consider a universe of discourse X , whose elements are denoted as x . A fuzzy set A in X may be defined as

$$A = \{(x, \mu_A(x)) | x \in X\} \quad (2.1)$$

where $\mu_A(x)$ is the MF of x in A , which represents the grade (degree) of the element x belongs to A . The MF $\mu_A(\cdot)$ maps each element in X onto a continuous unit interval $[0, 1]$. Specifically,

$$X \xrightarrow{\mu_A(x)} [0, 1] \quad (2.2)$$

An MF value of unity implies that this specific element x is definitely an element within this fuzzy set A . An MF value of zero means that this element x is definitely not an element of this fuzzy set A . An MF degree greater than zero and less than

unity indicates that this element x falls on the fuzzy boundary of this fuzzy set A . It is obvious that a fuzzy set is an extension of a classical crisp set by generalizing the range of the characteristic function from the crisp numbers $\{0, 1\}$ to the unit interval $[0, 1]$. If the value of the MF $\mu_A(x)$ is restricted to either 0 or 1, then A is reduced to a classical crisp set and $\mu_A(x)$ is the characteristic function of A .

Example 2.1 Fuzzy Sets with Discrete X

Let $X = \{1, 2, 3, 4, 5, 6, 7\}$ be the day within a week ($1 = \text{Monday}, \dots, 7 = \text{Sunday}$). The fuzzy set $A = \text{estimated traffic flow for a specific interstate road}$ can be described as

$$A = \{(1, 0.57), (2, 0.91), (3, 0.93), (4, 0.95), (5, 0.98), (6, 1), (7, 0.64)\} \quad (2.3)$$

The fuzzy set A can be shown in Figure 2.1a.

Example 2.2 Fuzzy Sets with Continuous X

Let $X = R^+$ be the set of possible ages of human beings. The fuzzy set $B = \text{around 40 years old}$ may be expressed as

$$B = \{(x, \mu_B(x)) | x \in X\} \quad (2.4)$$

with

$$\mu_B(x) = \frac{1}{1 + \left(\frac{x-40}{3}\right)^4} \quad (2.5)$$

The fuzzy set B is shown in Figure 2.1b.

An alternative way of representing a fuzzy set A is by using the concept of support, which is the crisp set of all $x \in X$ such that $\mu_A(x) > 0$. That is,

$$\text{supp}(A) = \{x \in X | \mu_A(x) > 0\} \quad (2.6)$$

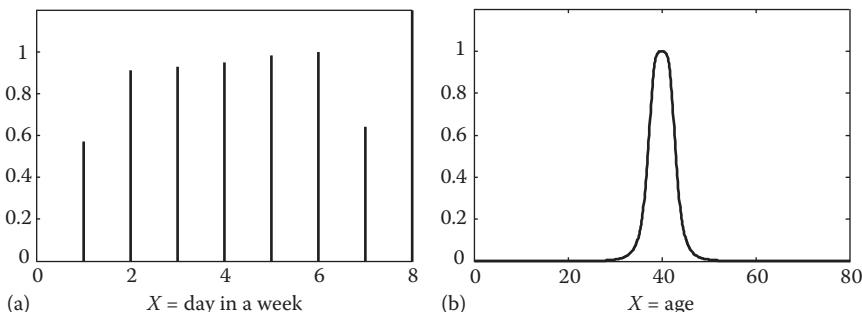


FIGURE 2.1 (a) Fuzzy set with discrete X and (b) fuzzy set with continuous X .

The representation of a fuzzy set A can be expressed in terms of the support of the fuzzy set as

$$A = \begin{cases} \sum_{x_i \in X} \mu_A(x_i)/x_i & \text{if } X \text{ is discrete} \\ \int_{x \in X} \mu_A(x)/x & \text{if } X \text{ is continuous} \end{cases} \quad (2.7)$$

Note that \sum and \int are symbolic representations of the union of the elements in A , where $\mu_A(x_i)$ is the degree of the membership of x_i with $\mu_A(x_i) > 0$. With this notation, the fuzzy sets in Examples 2.1 and 2.2 can be rewritten as

$$A = 0.6/1 + 0.7/2 + 1/3 + 0.9/4 + 0.7/5 + 0.3/6 + 0.2/7 \quad (2.8)$$

and

$$B = \int_{R^+} \frac{1}{1 + \left(\frac{x-40}{3}\right)^4} / x \quad (2.9)$$

A fuzzy set A whose support is a single point in the universe of discourse X with $\mu_A(x) = 1$ is defined as a fuzzy singleton. The kernel of a fuzzy set A consists of all the elements $x \in X$ whose membership degree is 1 as $\text{kernel}(A) = \{x \in X \mid \mu_A(x) = 1\}$. The cardinality of a fuzzy set A is defined as the summation of the membership grades of all the elements of x in A as $|A| = \sum_{x \in X} \mu_A(x)$. The height of a fuzzy set A is the maximum of $\mu_A(x)$ over the span of the universe of discourse as $\text{height}(A) = \max_{x \in X} \mu_A(x)$. A fuzzy set A is normalized when the height of the fuzzy set is unity as $\text{height}(A) = 1$; otherwise it is subnormal. A nonempty fuzzy set A can be normalized as $A' = \text{normal}(A) \leftrightarrow \mu_{A'}(x) = \mu_A(x)/\text{height}(A)$. A fuzzy set A is convex if and only if

$$\mu_A[\lambda x_1 + (1 - \lambda)x_2] \geq \min(\mu_A(x_1), \mu_A(x_2)) \quad x_1, x_2 \in X, \lambda \in [0, 1] \quad (2.10)$$

A continuous, normalized, and convex fuzzy set defined on the real line \mathfrak{R} is called a fuzzy number, which is commonly used for system modeling and control.

2.1.2 FUZZY OPERATIONS

With these basic definitions and notation of fuzzy sets, we are now ready to introduce some basic set-theoretic operations for fuzzy reasoning and fuzzy control. Suppose A and B are two fuzzy sets within the universe of discourse X .

1. Complement: The complement of fuzzy set A is denoted as \bar{A} (or not A), which is defined by its MF as

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \quad (2.11)$$

2. Intersection: The intersection of two fuzzy sets A and B is a fuzzy set C , which can be written as $C = A \cap B$ (or $C = A$ and B). The MF of fuzzy set C is related to those of A and B as

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x) \quad (2.12)$$

where \wedge denotes the min operation between these two MFs. The element x has to belong to both fuzzy sets simultaneously, which means that the smaller value of the two membership degrees will be obtained. It is obvious that

$$A \cap B \subseteq A \quad \text{and} \quad A \cap B \subseteq B \quad (2.13)$$

3. Union: The union of two fuzzy sets A and B is a fuzzy set C , which can be written as $C = A \cup B$ (or $C = A$ or B). The MF of fuzzy set C is given by

$$\mu_C(x) = \max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x) \quad (2.14)$$

where \vee is the max operation between these two MFs. The element x may be in one set or the other, and the larger value of the two membership degrees will be obtained. It is also clear that

$$A \subseteq A \cup B \quad \text{and} \quad B \subseteq A \cup B \quad (2.15)$$

4. Equality: The two fuzzy sets A and B are equal if and only if

$$A = B \Leftrightarrow \mu_A(x) = \mu_B(x) \quad (2.16)$$

5. Subset: Fuzzy set A is a subset of B (or A is smaller than or equal to B) if and only if $\mu_A(x) \leq \mu_B(x)$ for all x , such that

$$A \subseteq B \Leftrightarrow \mu_A(x) < \mu_B(x) \quad (2.17)$$

If $A \subseteq B$ and $A \neq B$, then A is a proper subset of B (or $A \subset B$).

6. Cartesian product: Suppose A_1, A_2, \dots, A_n are fuzzy sets in the universe of discourse X_1, X_2, \dots, X_n , respectively. The Cartesian product $C = A_1 \times A_2 \times \dots \times A_n$ is a fuzzy subset of the Cartesian product space $X_1 \times X_2 \times \dots \times X_n$ with the membership degree as min intersection:

$$\mu_{A_1 \times A_2 \times \dots \times A_n}(x_1, x_2, \dots, x_n) = \min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \dots, \mu_{A_n}(x_n)) \quad (2.18)$$

or algebraic product

$$\mu_{A_1 \times A_2 \times \dots \times A_n}(x_1, x_2, \dots, x_n) = \mu_{A_1}(x_1) \cdot \mu_{A_2}(x_2), \dots, \mu_{A_n}(x_n) \quad (2.19)$$

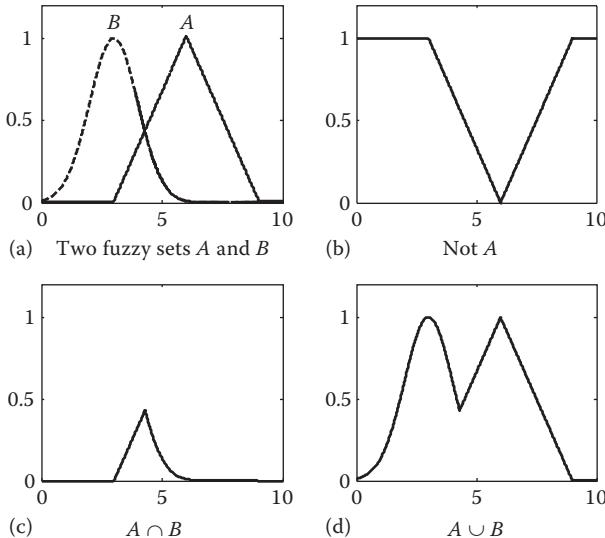


FIGURE 2.2 (a) Two fuzzy sets A and B , (b) complement of fuzzy set A , (c) intersection of two fuzzy sets $A \cap B$, and (d) union of two fuzzy sets $A \cup B$.

Figure 2.2 illustrates two fuzzy sets A and B , and the three fuzzy set operations: complement of fuzzy set A , intersection of two fuzzy sets A and B , and union of two fuzzy sets $A \cup B$.

2.1.3 MEMBERSHIP FUNCTIONS

In the following, several classes of parameterized functions are introduced, which are used extensively in defining MFs in fuzzy inference systems.

2.1.3.1 Triangular Membership Function

A triangular MF is specified by three parameters $\{a, b, c\}$, which determine the x coordinates of three corners as

$$\text{trimf}(x; a, b, c) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & c \leq x \end{cases} \quad (2.20)$$

or more compactly as

$$\text{trimf}(x; a, b, c) = \max \left(\min \left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right) \quad (2.21)$$

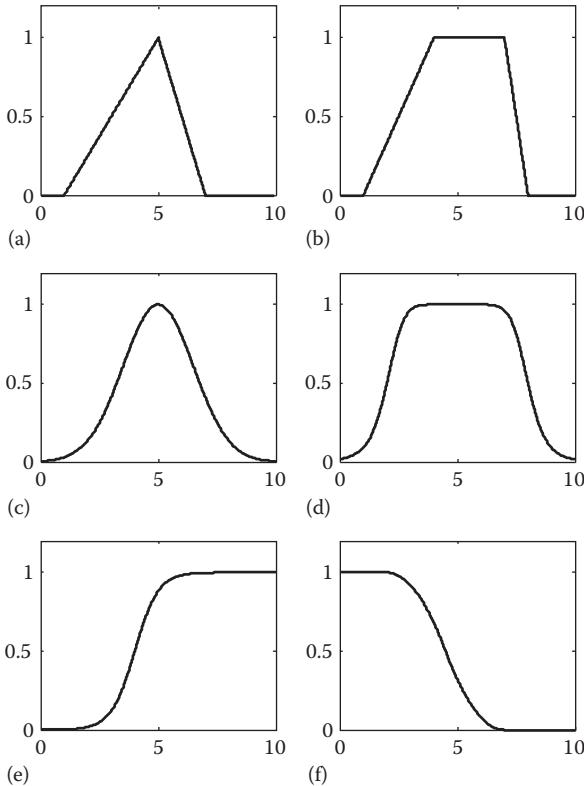


FIGURE 2.3 (a) Triangular MF, (b) trapezoidal MF, (c) Gaussian MF, (d) generalized bell MF, (e) sigmoidal MF, and (f) Z-shaped MF.

The parameters a and c locate the x coordinates of the feet of the triangle and the parameter b locates the x coordinate of the peak of the triangle. Figure 2.3a illustrates an example of a triangular MF, which is defined by trimf (x : 1, 5, 7).

2.1.3.2 Trapezoidal Membership Function

A trapezoidal MF is defined by four parameters $\{a, b, c, d\}$ as

$$\text{trapmf}(x: a, b, c) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \frac{d-x}{d-c}, & c \leq x \leq d \\ 0, & d \leq x \end{cases} \quad (2.22)$$

or more compactly as

$$\text{trapmf}(x; a, b, c) = \max \left(\min \left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right), 0 \right) \quad (2.23)$$

The parameters a and d locate the x coordinates of the feet of the trapezoid and the parameters b and c locate the x coordinate of the shoulders. [Figure 2.3b](#) illustrates an example of a trapezoidal MF, which is defined by $\text{trapmf}(x; 1, 4, 7, 8)$. It is obvious that the triangular MF is a special case of the trapezoidal MF with the shoulders' distance as zero.

Both triangular MF and trapezoidal MF are linear functions and have been used extensively especially in real-time implementation due to the easy computational requirement. However, for complex systems, nonlinear functions will be beneficial in representing system's smooth transition at switching points, which will be introduced next.

2.1.3.3 Gaussian Membership Function

A Gaussian MF is specified by two parameters $\{\sigma, c\}$:

$$\text{gaussmf}(x; \sigma, c) = \exp \left[\frac{-(x-c)^2}{2\sigma^2} \right] \quad (2.24)$$

where

c determines the center of the MF

σ represents the width

Figure 2.3c depicts a Gaussian MF, which is defined by $\text{gaussmf}(x; 1.5, 5)$.

2.1.3.4 Generalized Bell Membership Function

A generalized bell MF depends on three parameters $\{a, b, c\}$ as

$$\text{gbellmf}(x; a, b, c) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}} \quad (2.25)$$

where the parameter c determines the center of the curve, and the parameters a and b adjust the shape of the bell function. The bell MF has one more parameter than the Gaussian MF, so it can approximate a nonfuzzy set by tuning the parameters a and b . Figure 2.3d shows a generalized bell MF defined by $\text{gbellmf}(x; 3, 4, 5)$. Because of their concise notation, Gaussian and bell MFs are popularly used in specifying fuzzy sets, which have the advantage of being smooth and differentiable at all points.

2.1.3.5 Sigmoidal Membership Function

A sigmoidal MF depends on two parameters $\{a, c\}$ as

$$\text{sigmf}(x; a, c) = \frac{1}{1 + e^{-a(x-c)}} \quad (2.26)$$

where the sign of the parameter a determines the open-end direction of the sigmoidal MF. If a is a positive number, the MF will open to the right. If a is negative, the MF will open to the left. By this property, it is convenient to represent the fuzzy concepts such as extreme positive or extreme negative. [Figure 2.3e](#) illustrates a sigmoidal MF defined by $\text{sigmf}(x; [2, 4])$.

2.1.3.6 Z-Shaped Membership Function

A Z-shaped MF depends on two parameters $\{a, b\}$, which locate the extremes of the sloped portion of the curve as

$$\text{zmf}(x; a, b) = \begin{cases} 1, & x \leq a \\ 1 - 2\left(\frac{x-a}{b-a}\right)^2, & a \leq x \leq \frac{a+b}{2} \\ 2\left(b - \frac{x}{b-a}\right)^2, & \frac{a+b}{2} \leq x \leq b \\ 0, & b \leq x \end{cases} \quad (2.27)$$

[Figure 2.3f](#) illustrates a Z-shaped MF defined by $\text{zmf}(x; [2, 7])$.

2.1.4 FUZZY RELATIONS

A fuzzy relation among fuzzy sets X_1, X_2, \dots, X_n is a fuzzy subset on the Cartesian product space $X_1 \times X_2 \times \dots \times X_n$ and can be denoted by $R(X_1, X_2, \dots, X_n)$ as

$$R(X_1, X_2, \dots, X_n) = \int_{X_1 \times X_2 \times \dots \times X_n} \mu_R(x_1, x_2, \dots, x_n) / (x_1, x_2, \dots, x_n) \quad (2.28)$$

where $\mu_R(x_1, x_2, \dots, x_n)$ is a MF of the relation R , which represents the degree of association (correlation) among the elements of the different domain X_i . It is a mapping from the Cartesian space $X_1 \times X_2 \times \dots \times X_n$ onto a continuous unit interval $[0, 1]$ as

$$R: X_1 \times X_2 \times \dots \times X_n \rightarrow [0, 1] \quad (2.29)$$

Note that a fuzzy relation is in fact a fuzzy set. In general, the MF $\mu_R(x_1, x_2, \dots, x_n)$ is a hyperplane in the $(n+1)$ -dimensional space. The simplest fuzzy relation is called a binary fuzzy relation, which is a fuzzy relation between two sets X_1 and X_2 , and can be denoted as

$$R(X_1, X_2) = \{(x_1, x_2), \mu_R(x_1, x_2)) | (x_1, x_2) \in X_1 \times X_2\} \quad (2.30)$$

Suppose $X_1 = \{x_{11}, x_{12}, \dots, x_{1n}\}$ and $X_2 = \{x_{21}, x_{22}, \dots, x_{2m}\}$. The fuzzy relation $R(X_1, X_2)$ can be expressed as an $n \times m$ fuzzy matrix as

$$R(X_1, X_2) = \begin{bmatrix} \mu_R(x_{11}, x_{21}) & \mu_R(x_{11}, x_{22}) & \cdots & \mu_R(x_{11}, x_{2m}) \\ \mu_R(x_{12}, x_{21}) & \mu_R(x_{12}, x_{22}) & \cdots & \mu_R(x_{12}, x_{2m}) \\ \vdots & \vdots & \ddots & \vdots \\ \mu_R(x_{1n}, x_{21}) & \mu_R(x_{1n}, x_{22}) & \cdots & \mu_R(x_{1n}, x_{2m}) \end{bmatrix} \quad (2.31)$$

Example 2.3

Consider two universes of discourse in discrete domain as $X_1 = \{1, 2, 3, 4\}$ and $X_2 = \{1, 2, 3, 4, 5\}$. Let $R(X_1, X_2)$ represents the fuzzy relation point (x_1, x_2) that is close to $(3, 2)$, which might be defined by the following MF $\mu_R(x_1, x_2)$ as in Table 2.1.

Or it can be represented in another form with the support notation as

$$\begin{aligned} R(X_1, X_2) = & 0/(1, 1) + 0.1/(2, 1) + 0.3/(3, 1) + 0.1/(4, 1) + 0/(1, 2) + 0.3/(2, 2) \\ & + 1/(3, 2) + 0.3/(4, 2) + 0/(1, 3) + 0.1/(2, 3) + 0.3/(3, 3) + 0.1/(4, 3) \\ & + 0/(1, 4) + 0/(2, 4) + 0/(3, 4) + 0/(4, 4) + 0/(1, 5) + 0/(2, 5) \\ & + 0/(3, 5) + 0/(4, 5) \end{aligned} \quad (2.32)$$

Having explained the concept and formulation of fuzzy relation $R(X_1, X_2, \dots, X_n)$, in the following we introduce some operations that are specific to fuzzy relations:

1. Projection: Given a fuzzy relation $R(X_1, X_2)$ in the Cartesian product space $X_1 \times X_2$, the projection of R onto X_2 is a fuzzy set in X_2 , which is denoted as $[R \downarrow X_2]$ and its MF is defined by

$$\mu_{[R \downarrow X_2]}(x_2) = \max_{x_1} \mu_R(x_1, x_2) \quad (2.33)$$

Consider the fuzzy relation $R(X_1, X_2)$ in Table 2.1, the projection of $R(X_1, X_2)$ onto X_1 and X_2 are obtained individually as

$$[R \downarrow X_1] = 0/x_{11} + 0.3/x_{12} + 1/x_{13} + 0.3/x_{14} \quad (2.34)$$

TABLE 2.1
Fuzzy Relation Point (x_1, x_2) Is Close to $(3, 2)$

	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
	1	2	3	4	5
x_{11}	1	0	0	0	0
x_{12}	2	0.1	0.3	0.1	0
x_{13}	3	0.3	1	0.3	0
x_{14}	4	0.1	0.3	0.1	0

$$[R \downarrow X_2] = 0.3/x_{21} + 1/x_{22} + 0.3/x_{23} + 0/x_{24} + 0/x_{25} \quad (2.35)$$

2. Cylindrical extension: Given a fuzzy relation $R(X_1)$ or a fuzzy set R on X_1 , the cylindrical extension of R into X_2 is a fuzzy relation in $X_1 \times X_2$, which is denoted as $[R \uparrow X_2]$ and the MF is defined by

$$\mu_{[R \uparrow X_2]}(x_1, x_2) = \mu_R(x_1) \quad (2.36)$$

Note that a cylindrical extension is a fuzzy set in the Cartesian product domain and is the inverse of the projection operation.

Example 2.4

Continuing Example 2.3, we get the two cylindrical extensions in Cartesian product space $X_1 \times X_2$ as $[[R \downarrow X_1] \uparrow X_2]$ and $[[R \downarrow X_2] \uparrow X_1]$, which are shown in Tables 2.2 and 2.3, respectively.

The concept of cylindrical extension can be used to extend an r -ary relation $R(X_1, X_2, \dots, X_r)$ to an n -ary relation $R(X_1, X_2, \dots, X_r, \dots, X_n)$ when $n > r$.

3. Composition of fuzzy relations: Two important composition methods of fuzzy relation are the max–min composition and max–product composition,

TABLE 2.2
[[R \downarrow X₁] \uparrow X₂]

	x ₂₁	x ₂₂	x ₂₃	x ₂₄	x ₂₅
	1	2	3	4	5
x ₁₁	1	0	0	0	0
x ₁₂	2	0.3	0.3	0.3	0.3
x ₁₃	3	1	1	1	1
x ₁₄	4	0.3	0.3	0.3	0.3

TABLE 2.3
[[R \downarrow X₂] \uparrow X₁]

	x ₂₁	x ₂₂	x ₂₃	x ₂₄	x ₂₅
	1	2	3	4	5
x ₁₁	1	0.3	1	0.3	0
x ₁₂	2	0.3	1	0.3	0
x ₁₃	3	0.3	1	0.3	0
x ₁₄	4	0.3	1	0.3	0

which can be applied to both set-relation compositions and relation-relation compositions. For set-relation compositions, suppose A is a fuzzy set on X_1 and $R(X_1, X_2)$ is a fuzzy relation on the Cartesian product space $X_1 \times X_2$. The max-min composition of A and $R(X_1, X_2)$ can be denoted as $A \circ R(X_1, X_2)$ and is defined by

$$\mu_{A \circ R}(x_2) = \max_{x_1 \in X_1} \min[\mu_A(x_1), \mu_R(x_1, x_2)] \quad (2.37)$$

The max-product composition of A and $R(X_1, X_2)$ can be denoted as $A \odot R(X_1, X_2)$ and is defined by

$$\mu_{A \odot R}(x_2) = \max_{x_1 \in X_1} [\mu_A(x_1) \cdot \mu_R(x_1, x_2)] \quad (2.38)$$

For relation-relation compositions, suppose $P(X_1, X_2)$ and $Q(X_2, X_3)$ are two fuzzy relations on the Cartesian product spaces $X_1 \times X_2$ and $X_2 \times X_3$. The max-min composition of $P(X_1, X_2)$ and $Q(X_2, X_3)$ can be interpreted as the strength of the relational chain between the elements in X_1 and X_3 . It can be denoted as $P(X_1, X_2) \circ Q(X_2, X_3)$ and is defined as

$$\mu_{P \circ Q}(x_1, x_3) = \max_{x_2 \in X_2} \min[\mu_P(x_1, x_2), \mu_Q(x_2, x_3)] \quad (2.39)$$

The max-product composition of $P(X_1, X_2)$ and $Q(X_2, X_3)$ can be denoted as $P(X_1, X_2) \odot Q(X_2, X_3)$ and is defined by

$$\mu_{P \odot Q}(x_1, x_3) = \max_{x_2 \in X_2} [\mu_P(x_1, x_2) \cdot \mu_Q(x_2, x_3)] \quad (2.40)$$

Both the max-product and max-min composition satisfy the following properties as

$$P \circ Q \neq Q \circ P \quad (2.41)$$

$$(P \circ Q)^{-1} = Q^{-1} \circ P^{-1} \quad (2.42)$$

$$(P \circ Q) \circ T = P \circ (Q \circ T) \quad (2.43)$$

Example 2.5

Consider three universes of discourse $X_1 = \{x_{11}, x_{12}\}$, $X_2 = \{x_{21}, x_{22}, x_{23}\}$, and $X_3 = \{x_{31}, x_{32}\}$. Given two fuzzy relations $P(X_1, X_2) = \begin{pmatrix} 0.2 & 0 & 0.3 \\ 0.8 & 1 & 0.1 \end{pmatrix}$ and $Q(X_2, X_3) = \begin{pmatrix} 0.4 & 1 \\ 0.1 & 0.9 \\ 0.3 & 0.6 \end{pmatrix}$. Then the fuzzy relation between the elements in X_1 and X_3 is

$$\begin{aligned}
& P(X_1, X_2) \circ Q(X_2, X_3) \\
&= \begin{pmatrix} 0.2 & 0 & 0.3 \\ 0.8 & 1 & 0.1 \end{pmatrix} \circ \begin{pmatrix} 0.4 & 1 \\ 0.1 & 0.9 \\ 0.3 & 0.6 \end{pmatrix} \\
&= \begin{pmatrix} (0.2 \wedge 0.4) \vee (0 \wedge 0.1) \vee (0.3 \wedge 0.3) & (0.2 \wedge 1) \vee (0 \wedge 0.9) \vee (0.3 \wedge 0.6) \\ (0.8 \wedge 0.4) \vee (1 \wedge 0.1) \vee (0.1 \wedge 0.3) & (0.8 \wedge 1) \vee (1 \wedge 0.9) \vee (0.1 \wedge 0.6) \end{pmatrix} \\
&= \begin{pmatrix} 0.2 \vee 0 \vee 0.3 & 0.2 \vee 0 \vee 0.3 \\ 0.4 \vee 0.1 \vee 0.1 & 0.8 \vee 0.9 \vee 0.1 \end{pmatrix} \\
&= \begin{pmatrix} 0.3 & 0.3 \\ 0.4 & 0.9 \end{pmatrix}
\end{aligned} \tag{2.44}$$

or

$$\begin{aligned}
& P(X_1, X_2) \odot Q(X_2, X_3) \\
&= \begin{pmatrix} 0.2 & 0 & 0.3 \\ 0.8 & 1 & 0.1 \end{pmatrix} \odot \begin{pmatrix} 0.4 & 1 \\ 0.1 & 0.9 \\ 0.3 & 0.6 \end{pmatrix} \\
&= \begin{pmatrix} (0.2 \times 0.4) \vee (0 \times 0.1) \vee (0.3 \times 0.3) & (0.2 \times 1) \vee (0 \times 0.9) \vee (0.3 \times 0.6) \\ (0.8 \times 0.4) \vee (1 \times 0.1) \vee (0.1 \times 0.3) & (0.8 \times 1) \vee (1 \times 0.9) \vee (0.1 \times 0.6) \end{pmatrix} \\
&= \begin{pmatrix} 0.08 \vee 0 \vee 0.09 & 0.2 \vee 0 \vee 0.18 \\ 0.32 \vee 0.1 \vee 0.03 & 0.8 \vee 0.9 \vee 0.06 \end{pmatrix} \\
&= \begin{pmatrix} 0.09 & 0.2 \\ 0.32 & 0.9 \end{pmatrix}
\end{aligned} \tag{2.45}$$

4. Compositional rule of inference: In knowledge-based systems, a fuzzy IF-THEN rule may be expressed in the form as

$$\text{Rule: IF } x \text{ is } A \text{ THEN } y \text{ is } B \tag{2.46}$$

where the phase “ x is A ” is named as antecedent, precondition, or premise, and “ y is B ” is called as consequence or conclusion. In essence, the fuzzy IF-THEN rule can be viewed as a fuzzy implication $R = A \rightarrow B$, which can be defined as a binary fuzzy relation $R(X, Y)$ on the Cartesian product space $X \times Y$. There are two typical operations to implement the fuzzy implication as

$$\begin{aligned}
R_c: \text{ Min operation (Mamdani)} \quad a \rightarrow b = a \wedge b \\
\text{where } \mu_{A \rightarrow B}(x, y) = \mu_A(x) \wedge \mu_B(y)
\end{aligned} \tag{2.47}$$

$$\begin{aligned}
R_p: \text{ Product operation (Larsen)} \quad a \rightarrow b = a \cdot b \\
\text{where } \mu_{A \rightarrow B}(x, y) = \mu_A(x) \cdot \mu_B(y)
\end{aligned} \tag{2.48}$$

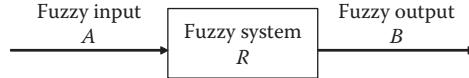


FIGURE 2.4 Fuzzy system expression based on input–output relation.

Thus, the set–relation composition of A and $R(X, Y)$ results in the fuzzy set B in Y based on max–min composition:

$$A \circ R(X, Y) = B \quad (2.49)$$

with the MF expressed as

$$\mu_B(y) = \mu_{A \circ R}(y) = \max_{x \in X} \min[\mu_A(x), \mu_R(x, y)] \quad (2.50)$$

Or it can become based on max–product composition as

$$A \odot R(X, Y) = B \quad (2.51)$$

with

$$\mu_B(y) = \mu_{A \odot R}(y) = \max_{x \in X} [\mu_A(x) \cdot \mu_R(x, y)] \quad (2.52)$$

Equations 2.49 and 2.51 are the so-called fuzzy relation equations, which describe the characteristics of a fuzzy system via its input–output relation as shown in Figure 2.4. Given a fuzzy input A to a fuzzy system R , the fuzzy output B can be directly calculated by a series of fuzzy operations.

Example 2.6

Suppose a fuzzy relation $R(X, Y)$ in the Cartesian product space $X \times Y$ as $R(X, Y) = \begin{pmatrix} 0.2 & 0.8 & 0.3 & 0.9 \\ 0.4 & 0.9 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.3 & 0.5 \end{pmatrix}$. This matrix represents the fuzzy rule-based system “if x is A then y is B . ” Now let a fuzzy set A in X be given as the input to this fuzzy system as $A = 0.3/x_1 + 1/x_2 + 0.7/x_3 = (0.3 \ 1 \ 0.7)$. Then we can easily derive the fuzzy output as

$$\begin{aligned}
B &= A \circ R \\
&= (0.3 \ 1 \ 0.7) \circ \begin{pmatrix} 0.2 & 0.8 & 0.3 & 0.9 \\ 0.4 & 0.9 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.3 & 0.5 \end{pmatrix} \\
&= (0.2 \vee 0.4 \vee 0.2 \ 0.3 \vee 0.9 \vee 0.5 \ 0.3 \vee 0.1 \vee 0.3 \ 0.3 \vee 0.4 \vee 0.5) \\
&= (0.4 \ 0.9 \ 0.3 \ 0.5) \\
&= 0.4/y_1 + 0.9/y_2 + 0.3/y_3 + 0.5/y_4
\end{aligned} \quad (2.53)$$

or

$$\begin{aligned}
 B &= A \odot R \\
 &= (0.3 \ 1 \ 0.7) \odot \begin{pmatrix} 0.2 & 0.8 & 0.3 & 0.9 \\ 0.4 & 0.9 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.3 & 0.5 \end{pmatrix} \\
 &= (0.06 \vee 0.4 \vee 0.14 \quad 0.24 \vee 0.9 \vee 0.35 \quad 0.09 \vee 0.1 \vee 0.21 \quad 0.27 \vee 0.4 \vee 0.35) \\
 &= (0.4 \quad 0.9 \quad 0.21 \quad 0.35) \\
 &= 0.4/y_1 + 0.9/y_2 + 0.21/y_3 + 0.35/y_4
 \end{aligned} \tag{2.54}$$

2.1.5 FUZZY INFERENCE SYSTEM

Linguistic variable is an important concept in a fuzzy inference system. Basically, it is a variable whose values are words, phases, or sentences in natural language. This concept was first introduced by Zadeh (1975) to provide a way of expressing a phenomenon that is too complex or ill-defined to describe in a conventional crisp manner. In order to illustrate the concept of linguistic variable, first, let us define the fuzzy variable, which could be characterized by three factors: the name of variable, the universe of discourse, and the description for this fuzzy variable. For example, the fuzzy variable's **high score** can be described as $0/60 + 0.1/70 + 0.5/80 + 0.9/90 + 1/100$ in the universe of discourse $\{60, 70, 80, 90, 100\}$, and the fuzzy variable's **low score** can be defined as $1/60 + 0.9/70 + 0.5/80 + 0.1/90 + 0/100$. A linguistic variable is in a higher level than a fuzzy variable, which takes fuzzy variables as its values. For example, for the linguistic variable **score** within the universe of discourse $\{60, 70, 80, 90, 100\}$, the value of the linguistic variable **score** could be {very low, low, middle, high, very high}, each of which is a fuzzy variable with the value as defined previously.

The basic idea of a fuzzy inference system is to incorporate human's knowledge into a set of fuzzy IF-THEN rules, which involve operations on linguistic variables. A typical structure of a fuzzy inference system consists of four components: a fuzzifier, a fuzzy rule base, a fuzzy inference engine, and a defuzzifier. The general fuzzy inference system can be shown in Figure 2.5.

2.1.5.1 Fuzzifier

The fuzzifier converts the numeric input into a linguistic variable. In usual case, it converts a crisp value x_0 into a fuzzy singleton A within the specified universe of

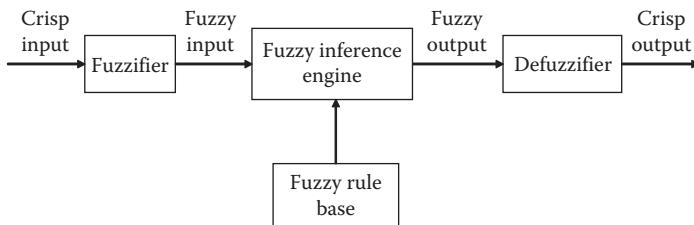


FIGURE 2.5 General structure of a fuzzy inference system.

discourse for easy calculation, where the MF $\mu_A(x_0) = 1$ at the specific point x_0 , and at the other points it is equal to zero. In real applications with noise, Gaussian fuzzifier and triangular fuzzifier may be used to characterize the preciseness of the input variable, which map the crisp number into the corresponding MFs. The vertex of the Gaussian function (or triangle) represents the mean value of the sensor measurements and the base is defined as a function of the standard deviation.

2.1.5.2 Fuzzy Rule Base

The fuzzy rule base is constructed by a collection of fuzzy IF-THEN rules, which can be expressed as a Mamdani linguistic fuzzy model:

$$R^i: \text{IF } x_1 \text{ is } A_{1i} \text{ AND } \dots \text{ AND } x_n \text{ is } A_{ni}, \text{ THEN } y \text{ is } B \quad (2.55)$$

or as a Takagi–Sugeno (TS) model:

$$R^i: \text{IF } x_1 \text{ is } A_{1i} \text{ AND } \dots \text{ AND } x_n \text{ is } A_{ni}, \text{ THEN } y = f_i(x_1, \dots, x_n) \quad (2.56)$$

where

x_1, \dots, x_n are the input linguistic variables

y is the output linguistic variable

A_{1i}, \dots, A_{ni} are the values for each input linguistic variables on the universes of discourse X_1, \dots, X_n

In Equation 2.55, B is the value for output y on the universes of discourse Y ; in Equation 2.56, $f_i(x_1, \dots, x_n)$ is a predefined function of the input variables for output y . A simple but practically useful expression is the affine form with linear parameters as

$$R^i: \text{IF } x_1 \text{ is } A_{1i} \text{ AND } \dots \text{ AND } x_n \text{ is } A_{ni}, \text{ THEN } y = a_{1i}x_1 + \dots + a_{ni}x_n + b_i \quad (2.57)$$

where

a_{1i}, \dots, a_{ni} are the linear parameters

b_i is a scalar offset

This model is called an affine TS model. In the fuzzy rule base expression, R^i ($i = 1, \dots, l$) represents the i th fuzzy rule. The entire fuzzy rule base R can be denoted as the union of the l individual relations R^i as

$$R = \bigcup_{i=1}^l R^i \quad (2.58)$$

2.1.5.3 Fuzzy Inference Engine

This is the kernel of the fuzzy inference system in performing fuzzy operations to map from a given input to an output, which is based on the generalized modus ponens as

$$\begin{array}{l}
 \text{Premise 1: IF } x \text{ is } A \text{ THEN } y \text{ is } B \\
 \text{Premise 2: } x \text{ is } A' \\
 \hline
 \text{Conclusion: } y \text{ is } B'
 \end{array} \tag{2.59}$$

For a fuzzy system with a set of rules as “if x is A then y is B ” and an input linguistic variable as A' , according to the compositional rule of inference in Equations 2.49 and 2.51, the fuzzy output B' will be derived as

$$B' = A' \circ R \tag{2.60}$$

$$\mu_{B'}(y) = \max_{x \in X} [\mu_{A'}(x) \wedge \mu_{A \rightarrow B}(x, y)] \tag{2.61}$$

for Mamdani's min implication R_c , or

$$B' = A' \odot R \tag{2.62}$$

$$\mu_{B'}(y) = \max_{x \in X} [\mu_{A'}(x) \cdot \mu_{A \rightarrow B}(x, y)] \tag{2.63}$$

for Larsen's produce implication R_p .

Example 2.7

Consider a general Mamdani linguistic fuzzy model in the case of two-input single-output system as

$$\begin{aligned}
 & R^1: \text{IF } x \text{ is } A_1 \text{ AND } y \text{ is } B_1, \text{ THEN } z \text{ is } C_1 \\
 & \text{ALSO } R^2: \text{IF } x \text{ is } A_2 \text{ AND } y \text{ is } B_2, \text{ THEN } z \text{ is } C_2 \\
 & \text{ALSO ...} \\
 & \text{ALSO } R^l: \text{IF } x \text{ is } A_l \text{ AND } y \text{ is } B_l, \text{ THEN } z \text{ is } C_l
 \end{aligned} \tag{2.64}$$

If the fuzzy inputs are x is A' and y is B' , with min intersection Cartesian product operation and Mamdani's min fuzzy implication R_c , the fuzzy output C' from the fuzzy inferencing will be calculated as

$$\begin{aligned}
 \mu_{C'}(z) &= [\mu_{A'}(x) \wedge \mu_{B'}(y)] \circ \bigcup_{i=1}^l \mu_{R^i}(x, y, z) \\
 &= [\mu_{A'}(x) \wedge \mu_{B'}(y)] \circ \bigcup_{x,y,z} [\mu_{R^1}(x, y, z), \dots, \mu_{R^l}(x, y, z)] \\
 &= \bigcup_{x,y} \left([\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge \left\{ \bigcup_{x,y,z} [\mu_{R^1}(x, y, z), \dots, \mu_{R^l}(x, y, z)] \right\} \right) \\
 &= \bigcup_{x,y,z} \bigcup_{x,y} (\{[\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge \mu_{R^1}(x, y, z)\}, \dots, \{[\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge \mu_{R^l}(x, y, z)\}) \\
 &= \bigcup_{x,y,z} (\{[\mu_{A'}(x) \wedge \mu_{B'}(y)] \circ \mu_{R^1}(x, y, z)\}, \dots, \{[\mu_{A'}(x) \wedge \mu_{B'}(y)] \circ \mu_{R^l}(x, y, z)\}) \\
 &= \bigcup_{i=1}^l \{[\mu_{A'}(x) \wedge \mu_{B'}(y)] \circ \mu_{R^i}(x, y, z)\}
 \end{aligned} \tag{2.65}$$

where $\mu_R(x, y, z) = \mu_{A_i}(x) \wedge \mu_{B_i}(y) \wedge \mu_{C_i}(z)$. If the input variables are fuzzy singletons as $A' = x_0$ and $B' = y_0$, then the firing strength α_i of the i th rule is expressed as $\alpha_i = \mu_{A_i}(x_0) \wedge \mu_{B_i}(y_0)$, which is a measure of the contribution of the i th rule to the resultant fuzzy output. And the final fuzzy output will be obtained as

$$\begin{aligned}\mu_{C'}(z) &= \bigcup_{i=1}^I \{ [\mu_{A'_i}(x) \wedge \mu_{B'_i}(y)] \circ \mu_R(x, y, z) \} \\ &= \bigcup_{i=1}^I \{ [\mu_{A'_i}(x_0) \wedge \mu_{B'_i}(y_0)] \circ [\mu_{A_i}(x) \wedge \mu_{B_i}(y) \wedge \mu_{C_i}(z)] \} \\ &= \bigcup_{i=1}^I \bigcup_{x,y} \{ [\mu_{A'_i}(x_0) \wedge \mu_{B'_i}(y_0)] \wedge [\mu_{A_i}(x) \wedge \mu_{B_i}(y) \wedge \mu_{C_i}(z)] \} \\ &= \bigcup_{i=1}^I [\mu_{A_i}(x_0) \wedge \mu_{B_i}(y_0) \wedge \mu_{C_i}(z)] \\ &= \bigcup_{i=1}^I [\alpha_i \wedge \mu_{C_i}(z)]\end{aligned}\quad (2.66)$$

In the same principle, with Larsen's product fuzzy implication R_p , the fuzzy output C' will be expressed as

$$\mu_{C'}(z) = \bigcup_{i=1}^I \{ [\mu_{A'_i}(x) \wedge \mu_{B'_i}(y)] \cdot \mu_{C_i}(z) \} = \bigcup_{i=1}^I [\alpha_i \cdot \mu_{C_i}(z)] \quad (2.67)$$

Example 2.8

Consider a general TS fuzzy model in the case of two-input single-output system as

$$\begin{aligned}R^1: \text{IF } x \text{ is } A_1 \text{ AND } y \text{ is } B_1, \text{ THEN } z \text{ is } f_1(x_1, x_2) \\ \text{ALSO } R^2: \text{IF } x \text{ is } A_2 \text{ AND } y \text{ is } B_2, \text{ THEN } z \text{ is } f_2(x_1, x_2) \\ \text{ALSO } \dots \\ \text{ALSO } R^l: \text{IF } x \text{ is } A_l \text{ AND } y \text{ is } B_l, \text{ THEN } z \text{ is } f_l(x_1, x_2)\end{aligned}\quad (2.68)$$

With the firing strength α_i of the i th rule expressed as $\alpha_i = \mu_{A_i}(x_0) \cdot \mu_{B_i}(y_0)$, the inferred values of the fuzzy output from each rule is $\alpha_i f_i(x_1, x_2)$ and the result of the fuzzy inferencing is obtained by weighting the combined membership value from the rule antecedent as

$$z_0 = \frac{\sum_{i=1}^l \alpha_i f_i(x_1, x_2)}{\sum_{i=1}^l \alpha_i} \quad (2.69)$$

2.1.5.4 Defuzzifier

Defuzzifier maps the system output from the fuzzy domain into the crisp domain, which is a necessary step as often a crisp number is required for real application. The center of area (COA) and the mean of maximum (MOM) are the two most commonly used methods in generating the crisp system output.

1. Center of area: This method produces the center of the fuzzy output area.
With a discrete universe of discourse,

$$x^* = \frac{\sum_{i=1}^n x_i \cdot \mu_A(x_i)}{\sum_{i=1}^n \mu_A(x_i)} \quad (2.70)$$

where

n is the number of the discrete elements in the universe of discourse

x_i is the value of the discrete element

$\mu_A(x_i)$ represents the corresponding MF value at the point x_i

If the universe of discourse is continuous, then the crisp system output will be calculated as

$$x^* = \frac{\int_{x \in X} x \cdot \mu_A(x) dx}{\int_{x \in X} \mu_A(x) dx} \quad (2.71)$$

2. Mean of maximum: Denote m as the number of the output points, whose MF values reach the maximum value within the universe of discourse. The MOM strategy calculates the mean value of all the m output points. In the case of a discrete universe, the crisp output is expressed as

$$x^* = \frac{\sum_{i=1}^m x_i}{m} \quad (2.72)$$

where x_i is the support value at these points, whose MF reaches the maximum value $\mu_A(x_i)$. The MOM method does not consider the shape of the fuzzy output but the defuzzification calculation is simplified.

Example 2.9

Consider a fuzzy inference system with the following two fuzzy IF-THEN rules:

$$\begin{aligned} R^1: & \text{ IF } x \text{ is } A_1 \text{ AND } y \text{ is } B_1, \text{ THEN } z \text{ is } C_1 \\ \text{ALSO } R^2: & \text{ IF } x \text{ is } A_2 \text{ AND } y \text{ is } B_2, \text{ THEN } z \text{ is } C_2 \end{aligned} \quad (2.73)$$

where the MFs of A_1 , A_2 , B_1 , B_2 , C_1 , and C_2 are defined as

$$\begin{aligned}\mu_{A_1}(x) &= \begin{cases} \frac{x-1}{3}; & 1 \leq x \leq 4 \\ \frac{7-x}{3}; & 4 \leq x \leq 7 \end{cases} & \mu_{A_2}(x) &= \begin{cases} \frac{x-4}{3}; & 4 \leq x \leq 7 \\ \frac{10-x}{3}; & 7 \leq x \leq 10 \end{cases} \\ \mu_{B_1}(y) &= \begin{cases} \frac{y-3}{3}; & 3 \leq y \leq 6 \\ \frac{9-y}{3}; & 6 \leq y \leq 9 \end{cases} & \mu_{B_2}(y) &= \begin{cases} \frac{y-2}{3}; & 2 \leq y \leq 5 \\ \frac{8-y}{3}; & 5 \leq y \leq 8 \end{cases} \\ \mu_{C_1}(z) &= \begin{cases} \frac{z-0}{3}; & 0 \leq z \leq 3 \\ \frac{6-z}{3}; & 3 \leq z \leq 6 \end{cases} & \mu_{C_2}(z) &= \begin{cases} \frac{z-2}{3}; & 2 \leq z \leq 5 \\ \frac{8-z}{3}; & 5 \leq z \leq 8 \end{cases}\end{aligned}\quad (2.74)$$

Suppose the crisp inputs to the system are $A' = x_0 = 5$ and $B' = y_0 = 6$. The final crisp system output will be computed as below:

First, the crisp inputs $x_0 = 5$ and $y_0 = 6$ have fired the preconditions A_1 and B_1 in Rule 1 and their truth values are represented as $\mu_{A_1}(x_0 = 5) = \frac{2}{3}$ and $\mu_{B_1}(y_0 = 6) = 1$, respectively. The resultant firing strength α_1 is calculated as

$$\alpha_1 = \mu_{A_1}(x_0) \wedge \mu_{B_1}(y_0) = \frac{2}{3} \wedge 1 = \frac{2}{3} \quad (2.75)$$

Based on Mamdani's min fuzzy implication R_c , the system output of Rule 1 is obtained as the dotted trapezoid area in Figure 2.6 as

$$\mu_{C'_1}(z) = \alpha_1 \wedge \mu_{C_1}(z) \quad (2.76)$$

This means that as a result of the crisp input values x_0 and y_0 , a fuzzy output C'_1 is derived from Rule 1 with $\mu_{C'_1}(z)$ as its MF. Similarly, For Rule 2, the truth values of

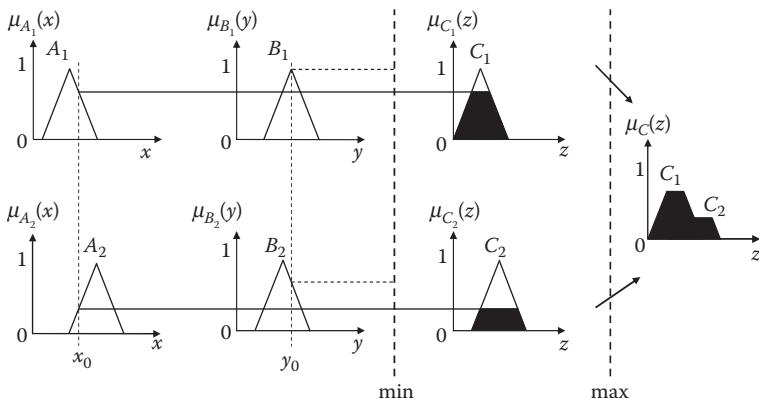


FIGURE 2.6 Fuzzy reasoning process based on Mamdani's min implication R_c .

the preconditions are $\mu_{A_2}(x_0 = 5) = \frac{1}{3}$ and $\mu_{B_2}(y_0 = 6) = \frac{2}{3}$, and the firing strength α_2 is calculated as

$$\alpha_2 = \mu_{A_2}(x_0) \wedge \mu_{B_2}(y_0) = \frac{1}{3} \wedge \frac{2}{3} = \frac{1}{3} \quad (2.77)$$

Applying the firing strength α_2 to the consequence C_2 results in the system output of Rule 2 as the dashed trapezoid area in [Figure 2.6](#) as

$$\mu_{C_2}(z) = \alpha_2 \wedge \mu_{C_2}(z) \quad (2.78)$$

Superimposing the two resultant MFs from the two fuzzy rules by max operation, the combined fuzzy output is obtained as

$$\mu_C(z) = \mu_{C_1}(z) \vee \mu_{C_2}(z) = [\alpha_1 \wedge \mu_{C_1}(z)] \vee [\alpha_2 \wedge \mu_{C_2}(z)] \quad (2.79)$$

This result is a MF and has to be defuzzified to a single crisp value. When the COA defuzzifier is executed, the crisp system output is calculated as

$$z^* = \frac{\sum_{i=1}^n z_i \cdot \mu_C(z_i)}{\sum_{i=1}^n \mu_C(z_i)} = \frac{1 \cdot \frac{1}{3} + 2 \cdot \frac{2}{3} + 3 \cdot \frac{2}{3} + 4 \cdot \frac{2}{3} + 5 \cdot \frac{1}{3} + 6 \cdot \frac{1}{3} + 7 \cdot \frac{1}{3}}{\frac{1}{3} + \frac{2}{3} + \frac{2}{3} + \frac{2}{3} + \frac{1}{3} + \frac{1}{3} + \frac{1}{3}} = 3.7 \quad (2.80)$$

When the MOM defuzzifier is conducted, only the output points at $z = 2, 3, 4$ with membership values as $\frac{2}{3}$, which reaches the maximum value within the universe of discourse, are used in calculating the crisp output:

$$z^* = \frac{\sum_{i=1}^m z_i}{m} = \frac{2 + 3 + 4}{3} = 3 \quad (2.81)$$

Example 2.10

In this example, let us consider the fuzzy reasoning process with Larsen's product fuzzy implication R_p ([Figure 2.7](#)). The firing strength α_i will be calculated the same way as in Example 2.9 for the two fuzzy rules as $\alpha_1 = \mu_{A_1}(x_0) \wedge \mu_{B_1}(y_0) = \frac{2}{3} \wedge 1 = \frac{2}{3}$ and $\alpha_2 = \mu_{A_2}(x_0) \wedge \mu_{B_2}(y_0) = \frac{1}{3} \wedge \frac{2}{3} = \frac{1}{3}$. However, the dotted triangular area and the dashed triangular area will be obtained as $\alpha_i \cdot \mu_{C_i}(z)$ for the consequence C_1 and C_2 , respectively. Finally, the combined fuzzy output is obtained as $\mu_C(z) = \cup_{i=1}^l [\alpha_i \cdot \mu_{C_i}(z)]$ by superimposition of the two triangular areas.

By the COA defuzzifier, the crisp system output is calculated as

$$z^* = \frac{\sum_{i=1}^n z_i \cdot \mu_C(z_i)}{\sum_{i=1}^n \mu_C(z_i)} = \frac{1 \cdot \frac{2}{9} + 2 \cdot \frac{4}{9} + 3 \cdot \frac{2}{3} + 4 \cdot \frac{4}{9} + 5 \cdot \frac{1}{3} + 6 \cdot \frac{2}{9} + 7 \cdot \frac{1}{9}}{\frac{2}{9} + \frac{4}{9} + \frac{2}{3} + \frac{4}{9} + \frac{1}{3} + \frac{2}{9} + \frac{1}{9}} \approx 3.5 \quad (2.82)$$

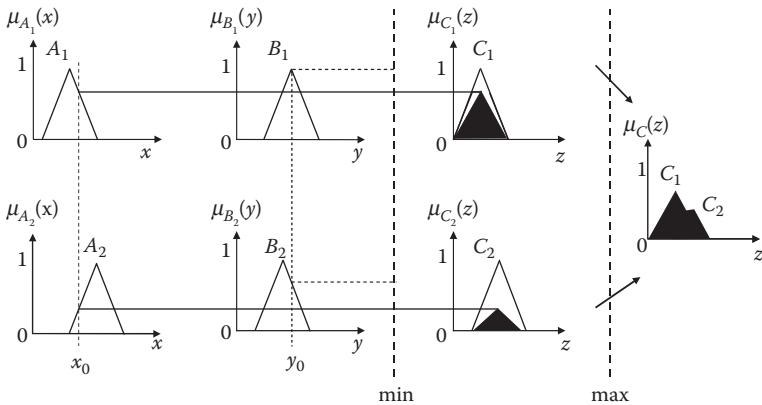


FIGURE 2.7 Fuzzy reasoning process based on Larsen's product implication R_p .

When using the MOM defuzzifier, only the output point at $z = 3$ with membership values as $\frac{2}{3}$, which reaches the maximum value within the universe of discourse, so the crisp output is calculated as

$$z^* = \frac{\sum_{i=1}^m z_i}{m} = \frac{3}{1} = 3 \quad (2.83)$$

2.2 ARTIFICIAL NEURAL NETWORKS

2.2.1 BASIC STRUCTURE

Inspired by biological signal processing, there has been a great resurgence of interest in developing models to learn various nonlinear systems based on data in recent years. These modeling techniques, called artificial neural networks (ANN), are iteratively trained to learn the system to be modeled. Based on the Stone–Weierstrass theorem, several researchers have shown that these neural networks (NN), with a sufficient number of neurons and sufficient training, can model any arbitrary bounded, continuous, nonlinear functions (Cybenko, 1989; Funahashi, 1989).

Artificial neural networks like biological neural systems consist of layers of multiple neurons that receive information and send out data after processing the received data through a certain activation function and weighted connections among these neurons. Neurons receive inputs from a large number of other neurons and communicate with others by sending activation or inhibition through connections. Learning involves modifying the connections among these neurons or sometimes activation or inhibition function itself.

Artificial neural networks have a parallel processing architecture in which knowledge is represented in the form of weights between a set of highly connected processing elements (PE). Thus, knowledge is said to be content addressable as opposed to the context addressable nature of the classical Von Neumann architecture.

Pattern recognition applications often use the binary form of NN representation. Analog ANNs have demonstrated the capability to perform nonlinear pattern association between input and output variables.

Continual attempts to emulate the biological signal processing have resulted in many different paradigms of ANN. While ANN may not accurately represent the actual functions of biological units, we consider them as general modeling and approximation tools applicable to a large class of nonlinear systems and will follow the convention without further justification of the name. Currently, there are many different paradigms of ANN and the commonly used ones are, although the list is not exhaustive, as follows:

- Adaline (Widrow and Hoff, 1960)
- Perceptron (Minsky and Papert, 1960)
- Kohonen networks
- Multilayer feedforward neural networks (FFNN) (Werbos, 1974; Rumelhart et al., 1986)
- Cerebellar model arithmetic computer (Albus, 1975)
- Radial basis function networks (RBFN) (Broomhead and Lowe, 1988)
- Adaptive resonance theory (Carpenter and Grossberg, 1988)
- Hopfield network (Hopfield, 1982)

The choice of any particular paradigm will depend on the problem to be addressed. Among these, FFNN, also called backpropagation networks (BPN), have initially gained wide popularity with the introduction of backpropagation algorithm for training.

The architectures of ANN can be roughly divided into three categories: FFNN, feedback, and cellular networks (Cichocki and Unbehauen, 1993). In FFNN, the neurons are arranged such that outputs from a layer of neurons are fed into the next layer of neurons. On the other hand, feedback networks, the output of one layer of neurons is fed back through the delay operator as input to the same layer of neurons. Such network structures provide a good way of modeling nonlinear dynamic systems, but are subject to stability problems. Cellular networks, however, have the array of neurons spatially distributed, which interact with neighboring neurons directly. These structures are schematically shown in [Figure 2.8](#).

Definition of Terms

Some of the commonly used terms while describing ANNs are presented in this section.

Node: A node or neuron is a static map that accepts multiple inputs and instantaneously produces a single, possibly nonlinear output. The node itself does not have any dynamics associated with it. If there is a node whose inputs are exclusively from the network inputs, such a node is called an input node. A node, whose output is one of the network outputs, is called as an output node. All other nodes are called as hidden nodes.

Connection: A connection is defined as a directed link between two nodes, feeding the output of one to the input of the other. A connection is associated with a weight and a time delay (this is zero for feedforward connections).

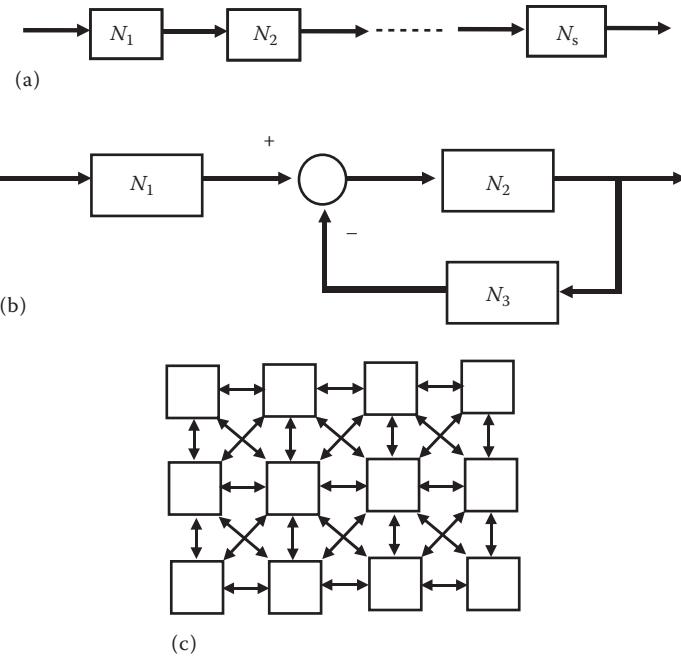


FIGURE 2.8 Schematics of various NN architectures. (a) Feed forward network, (b) feedback network and, (c) cellular network.

Layer: A layer consists of nodes arranged in a one-dimensional array and all the computations of nodes belonging to the same layer are started and finished at the same time.

2.2.2 MULTILAYER FEEDFORWARD NEURAL NETWORKS (BACKPROPAGATION NEURAL NETWORKS)

A multilayer BPNN consists of multiple layers of PE, which are interconnected by weighted arcs through activation functions. Each PE sums the product of its own inputs and the connection weights from the previous layer and then limits it by a nonlinear activating function. The sigmoidal function defined by

$$f(x) = \frac{1}{1 + e^{-\beta x}} \quad (2.84)$$

has been the activation function of choice in many applications. The next layer uses the outputs from the PE of the previous layer and computes the weighted sum limited by a threshold function, etc. A typical structure of a single hidden-layer FFNN is shown in [Figure 2.9](#) and its output can be obtained as follows:

$$y_j = f\left(\sum w_{ji} x_i + \theta_i\right) \quad (2.85)$$

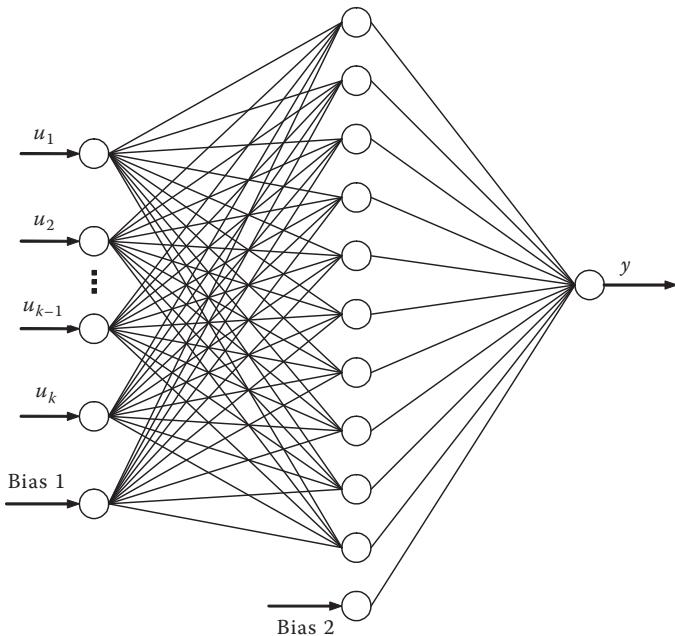


FIGURE 2.9 Single hidden-layer FFNN.

Similarly, multilayer FFNN can be represented in a nested sigmoid expression

$$y_k = f \left\{ \sum w_i' f \left[\sum w_j'' f(\cdots) \right] \right\} \quad (2.86)$$

where $f(\cdot)$ is the sigmoid function defined by Equation 2.84. Hidden layers are not directly visible and hence these networks are often considered black box models. A single hidden layer is usually all needed to approximate many nonlinear functions.

2.2.3 RADIAL BASIS FUNCTION NETWORKS

Radial basis functions (RBFs) have been used as a technique for interpolating multivariable scattered data and have recently attracted attention in the NN community (Broomhead and Lowe, 1988; Poggio and Girosi, 1990; Chichoki and Unbehauen, 1993). Similar to BPNN, RBFs possess the properties of approximating nonlinear functions of several variables. When a network is constructed based on RBF, however, it yields a structure that is linear in the parameters. This property provides a tremendous advantage in computational efficiency and effectiveness when continual adaptation of parameters is required. Convergence properties of the parameters can be guaranteed and the learning algorithm will find the global minimum of the output layer weights. While, in principle, it is possible to construct a multilayer structure (more than two hidden layers) like in BPNN, three-layer structure of RBFN (with one hidden layer) suffices to approximate a large class of nonlinear functions. Properties of RBFs are described below in comparison with BPNN.

Given an input–output pair of data set (u_k, y_k) , $k = 1, \dots, N$; $u_k \in \Re^n$, $y_k \in \Re$, let us assume the existence of a relation of the form

$$y_k = F(x_k) + e_k, \quad k = 1, \dots, N \quad (2.87)$$

where

F is an unknown function

e_k is the unknown error in the measurement of the dependent variable y_k

Conventional surface-fitting techniques would require the segmentation of the independent variable domain in order to successfully approximate the function F . Important decisions in obtaining an approximation to the function F are the choice of a suitable model structure, desired accuracy of fit, and the complexity of computations.

In general, all approximation schemes can be depicted as networks having different basis functions. The approximation process can be considered as a two step process:

1. Choosing a set of basis functions that are defined in the region of interest of the independent variable space
2. Determining the parameters in the basis function to achieve a good fit

Unlike the BPNN expression in an embedded fashion, the approximation representation for the RBFNN is a simple linear combination of the basis functions:

$$y_k = \sum \alpha_i \phi_i(x_k) \quad (2.88)$$

More specifically, we attempt to achieve the approximation in the following way using RBFN defined as follows.

2.2.3.1 Definition and Types of RBF

Definition: Given a continuous function $F: \Re^+ \rightarrow \Re$ and points $\{x_j^c: j = 1, \dots, p\}$, $X_j^c \in \Re^+$, which become dense in the open region D of \Re^+ , there exists a sequence of functions

$$\Phi_p(X) = \sum_{j=1}^p \alpha_j^p \phi\left(\|X - X_j^c\|\right) + \alpha_o^T X \quad (2.89)$$

and some bounded open domain in which

$$|\Phi_p(X) - F(X)| \rightarrow 0$$

as $p \rightarrow \infty$, where $\|\cdot\|$ is the Euclidean norm. The functions $\phi(\cdot)$ are termed RBFs, and the prototypical points, X_j^c , are referred to as the centers of the basis functions.

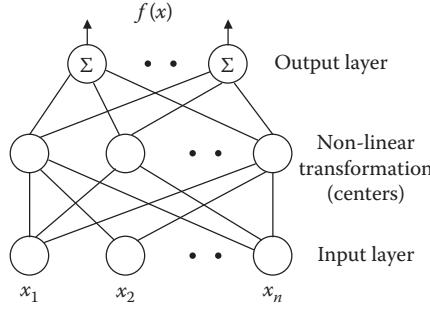


FIGURE 2.10 Radial basis function network

The general results of Kolmogorov's theorem provide the basis for the existence of such a function.

The RBF approach constructs an approximation based on the location of the data points. In typical applications, they are used to approximate a continuous function $F(\cdot)$ based on scattered sampling pair $(X_j, F(X_j))$, where $X_j \in D$. A linear term with the parameter α_0 has been added based on our experiences and does not affect the convergence proofs given by Jackson (1988). There are many different types of functions $\phi(\cdot)$ that satisfy the requirements of this definition (Poggio and Girosi, 1990). Functions found in the literature include thin plate splines ($\phi(r) = r^2 \log r$) (Duchon, 1977), Hardy multiquadratics ($\phi(r) = \sqrt{c + r^2}$) (Hardy, 1971), $\phi(r) = r^\ell$ (ℓ is an odd integer) (Powell, 1981), Gaussian $\phi(r) = \exp(-r^2)$ (Schagen, 1984), etc.

Regardless of the choice of the RBF, the approximated model assumes a linear form in terms of parameters α_j^p , if it is assumed that the centers of the basis functions are fixed beforehand. Based on these basis (activation) functions and parameters (weights), the paradigm RBFNN can be constructed to provide a mapping in domain D as shown in Figure 2.10. In this paradigm, neurons are represented by RBFs with the centers X_i^c , which are interconnected by weights. With a prespecified choice of centers, the structure resembles a single-layered NN, where each node of the hidden layer performs a nonlinear transformation specified by the basis function. There are $p+1$ nodes in the hidden layer, p of which are assigned a center and have an activation function given by $\phi(\cdot)$. One node of unity activation is used to denote the linear term. One of the inherent properties of RBFs in interpolation is that the parameters are uniquely defined if the coefficient matrix

$$A_{ij} = \phi\left(\|X_i - X_j\|\right) \quad (2.90)$$

is nonsingular. Among different RBFs described above, Gaussian RBFs have been the preference of many people.

The generalized RBFN with M hidden nodes, a single output node $y \in \Re$, and an input vector $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in \Re^n$, is described by

$$y = f(\mathbf{x}) = w_0 + \sum_{j=1}^M w_j \phi(\mathbf{x} | \lambda_j) \quad (2.91)$$

where

w_j 's are scalar weights

$\phi(\cdot)$ is a RBF

λ_j 's are a set of parameters for the j th RBF node

There are several types of RBFs that have been used for modeling nonlinear relationships. The popular Gaussian function used as the basis function is defined by

$$\phi(\mathbf{x}|\lambda_j) = \prod_{i=1}^n \exp\left[-\left(\frac{x_i - m_i^j}{\sigma_i^j}\right)^2\right] \quad (2.92)$$

where the nonlinear parameter set $\lambda_j = \{\mathbf{m}^j, \mathbf{o}^j\}$ include the width vector, $\mathbf{o}^j = (\sigma_1^j, \sigma_2^j, \dots, \sigma_n^j)^T \in \mathbb{R}^n$, and the center vector, $\mathbf{m}^j = (m_1^j, m_2^j, \dots, m_n^j)^T \in \mathbb{R}^n$.

For N given input–output training pairs, $(\mathbf{x}(t), d(t))$, $t = 1, 2, \dots, N$, the task of training a RBFN is to design an RBFN $f(\mathbf{x})$ such that

$$d(k) = f[\mathbf{x}(k)] + e(k) = w_0 + \sum_{j=1}^M \phi_j[\mathbf{x}(k)]w_j + e(k) \quad (2.93)$$

where $\phi_j(\cdot) = \phi(\cdot|\lambda_j)$. The above equation is arranged from $k = 1$ to N in matrix form (Chen et al., 1991):

$$\mathbf{d} = \Phi\mathbf{w} + \mathbf{e} \quad (2.94)$$

where

$$\mathbf{d} = [d(1), \dots, d(N)]^T \in \mathbb{R}^N$$

$$\Phi = [\phi_0, \phi_1, \dots, \phi_M] \in \mathbb{R}^{N \times (M+1)} \text{ with } \phi_0 = [1, \dots, 1]^T, \phi_j = \{\phi[\mathbf{x}(1)|\lambda_j], \dots, \phi[\mathbf{x}(N)|\lambda_j]\}^T$$

$$\mathbf{w} = (w_0, w_1, \dots, w_M)^T \in \mathbb{R}^{M+1}$$

$$\mathbf{e} = [e(1), \dots, e(N)]^T \in \mathbb{R}^N$$

Hence, it can be seen that constructing an RBFN becomes a simple linear least-squares problem, once the response matrix Φ , whose column vectors are response vectors of the RBF nodes, is determined. The question, however, is how the RBF nodes, specifically the nonlinear parameter sets $\lambda_j = \{\mathbf{m}^j, \mathbf{o}^j\}$, can be chosen or determined for optimum performance.

Some recent developments have made RBFNN more attractive. Leonard and Kramer (1991) have described a procedure to predict error bounds in approximation for RBFNN. Sanner and Slotine (1992) also provided a means of calculating approximation error bounds of Gaussian RBFN for use in direct adaptive control. Recently, Elanayar and Shin (1994) presented a constructive procedure to obtain error bounds of arbitrary RBFN. Since knowing the approximation error bounds is very critical to construction of robust, stable control laws, these works can contribute significantly to the further development of neuro-control schemes.

TABLE 2.4
Different Gaussian RBFs

Paradigm	$I_k(x)$	Isocontour Characteristics
1	$\lambda_k^2 \ x - x_k^c\ ^2 = \lambda_k^2 \left[(x_1 - x_{k1}^c)^2 + (x_2 - x_{k2}^c)^2 \right]$	Circles
2	$\lambda_k^2 \ D(x - x_k^c)\ ^2 = \lambda_k^2 \left[d_1^2 (x_1 - x_{k1}^c)^2 + d_1^2 (x_2 - x_{k2}^c)^2 \right]$	Ellipses with common aspect ratio, no rotation
3	$\lambda_k^2 \ W^T(x - x_k^c)\ ^2 = \lambda_k^2 \left\{ \left[w_{11}(x_1 - x_{k1}^c)^2 + w_{21}(x_2 - x_{k2}^c)^2 \right]^2 \times \left[w_{12}(x_1 - x_{k1}^c)^2 + w_{22}(x_2 - x_{k2}^c)^2 \right]^2 \right\}$	Ellipses with common aspect ratio, common rotation allowed
4	$\ D_k(x - x_k^c)\ ^2 = d_{k1}^2 (x_1 - x_{k1}^c)^2 + d_{k2}^2 (x_2 - x_{k2}^c)^2$	Ellipses with different aspect ratios, no rotation
5	$\ D_k W^T(x - x_k^c)\ ^2 = \left\{ d_{k1} \left[w_{11}(x_1 - x_{k1}^c)^2 + w_{21}(x_2 - x_{k2}^c)^2 \right] \right\}^2 + \left\{ d_{k2} \left[w_{12}(x_1 - x_{k1}^c)^2 + w_{22}(x_2 - x_{k2}^c)^2 \right] \right\}^2$	Ellipses with different aspect ratios, common rotation allowed
6	$\ D_k(x - x_k^c)\ ^2 = d_{k1}^2 (x_1 - x_{k1}^c)^2 + d_{k2}^2 (x_2 - x_{k2}^c)^2$ where $d_{ki} = \begin{cases} d_{ki}^+ & x_i \geq x_{ki}^c \\ d_{ki}^- & x_i < x_{ki}^c \end{cases}$	Quarter ellipses pieced together

Unfortunately, RBFN is not without downside. Most RBFs, like BPNN using sigmoid functions, also suffer from global effect. In other words, the changes in weights during training can cause global changes in the functional mapping. This behavior can be undesirable since local error can create global changes and unlearning of a previous satisfactory approximation of the function in a subspace of inputs. To overcome these problems, many variations of Gaussian RBF have been proposed. Some representative Gaussian RBFs that all have the same functional form $\phi(x) = e^{-I(x)}$ are summarized in Table 2.4 (Shin, 1998). Obviously, added degrees of freedom provide more flexibility in learning at the cost of increased training time. The issues that are still remaining include the choice of RBF paradigms, the number (order) of hidden nodes (or centers), and the selection of center locations. Currently, the order of the NN is determined based on iterative training by trial and errors.

2.2.4 RECURRENT NEURAL NETWORKS

2.2.4.1 Introduction

The capacity of FFNN such as the multilayer perceptron to approximate any spatially finite static function has been well studied and exploited in a number of applications. Recurrent neural networks (RNNs) are enhanced versions of these

ANN with feedback connections and are mathematical abstractions of biological nervous systems. The feedback connections make the RNN a dynamic system, which is capable of modeling spatial as well as temporal dependencies between input and output sequences. The feedback connections endow RNNs with internal states, which capture sufficient information about the data already processed by the network, while the feedforward connections combine these states with the network inputs at the current time to determine the future evolution of the network states and outputs. RNNs can be used to model both continuous time as well as discrete time systems by using differential equations and difference equations, respectively. For the sake of simplicity, only discrete time RNNs are considered here. By using a suitable number of network nodes, RNNs can approximate any dynamical system with arbitrary precision (Siegelmann and Sontag, 1991) and this makes them suitable for a wide range of applications including time series prediction, dynamic optimization, creation of associative memory banks, and dynamic process modeling and control.

The theoretical guarantees about the capabilities of RNNs however do not always ensure good results in practice. This is because of the difficulty associated with determining a suitable architecture, structure, and parameters for the network. The architecture of a network defines the important properties of an RNN such as the nature of feedback connections considered, the number of time lags considered, and the division of network nodes into input, output, and hidden nodes. For a given architecture, the structure of the RNN defines the number of inputs, the number of hidden layers, the number of nodes in each hidden layer, and the number of outputs. Finally, the parameters of an RNN are the weights associated with the connections and the nodes. Generally, for a given application, the architecture of an RNN is either chosen randomly or based on user's experience. The user then tries a large number of structures for the selected architecture and the parameters for each of these structures. The difficulty in making optimal choices for these properties has limited the application of RNNs in spite of its vast potential. Some details about the determination of these properties of an RNN will be provided in the coming sections along with possible issues for future work.

2.2.4.2 Network Architecture

The high flexibility in choosing the various components of an RNN architecture makes it difficult to classify RNNs based on any single criterion. Having said that, it is still useful to make some classifications as mentioned below.

2.2.4.2.1 Connection-Based Classification

Recurrent neural network architectures may vary from fully connected ([Figure 2.11a](#)) to partially connected nets ([Figure 2.11b](#)). Partially connected nets allow the description of RNN as a generalization of multilayer FFN with distinct input, output, and hidden nodes, whereas fully connected networks do not distinguish between various nodes or layers.

Two simple and popular architectures, which add feedback into a multilayer FFNN, are the Elman network (Elman, 1990) and the Jordan network (Jordan, 1989).

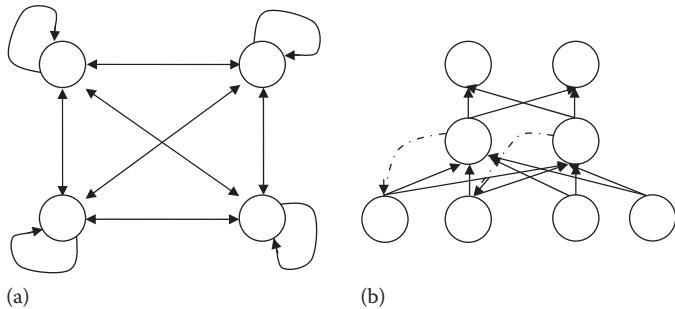


FIGURE 2.11 Connection-based classification of RNN architecture: (a) fully connected net and (b) partially connected net.

The Elman network introduces feedback from the hidden layer to a set of context nodes, whose output is then used as an input to the hidden layer in the next iteration. Context nodes are simply used for storage of the hidden node outputs at the previous iteration. Figure 2.11b is an example of an Elman network. This approach gives more importance to the sequence of input values to determine the temporal context of the network. Jordan networks use feedback from the output layer to a set of context nodes and give more importance to the sequence of output values to determine the temporal context of the network (Medsker and Jain, 2000).

2.2.4.2.2 Connection Delay-Based Classification

It is possible to use tapped delay lines with delays greater than one as connections in an RNN. The standard feedforward connection has a delay value of zero, while a simple feedback connection has a delay value of one. It may be shown that it is possible to achieve the dynamics of an RNN with connections having higher delay values (called a non-Markov RNN) using an RNN with only simple feedforward and feedback connections (called a Markov RNN), although the network size may become larger.

2.2.4.2.3 Node Type-Based Classification

Although a number of different functions, such as the RBFs and spline functions, have been tried successfully for the hidden nodes of feedforward networks, the sigmoid function remains the most popular choice for RNNs. This could possibly be because of the approximately linear or monotonic nature of many dynamic systems, which makes sigmoid functions a better choice for their approximation instead of local basis functions such as splines.

Therefore, a choice of network architecture involves the selection of the connection structure, connection delays allowed, and the type of hidden nodes to use. In the rest of this book, it is assumed that a partially connected Markov RNN with sigmoid nodes is selected as the architecture. This architecture is generic enough to model any dynamic system that can be represented in the state-space form and hence there is no need to consider architectures with additional complexities.

2.2.4.3 Structure and Parameter Learning

For a partially connected Markov RNN, structure learning involves the selection of an appropriate number of hidden layers and the number of hidden nodes in each hidden layer. Parameter learning then involves tuning the weights of the RNN. RNN training methods may broadly be divided into two categories. Earlier attempts for RNN training were based on extensions of gradient-based methods such as backpropagation (Rumelhart et al., 1986) and could only handle parameter learning. Recent efforts, which exploit the derivative free global optimization capabilities of newly emerging population-based methods such as evolutionary algorithms and particle swarm optimization, consider the possibility of learning both the structure and parameters of an RNN simultaneously during training. More details about these training methods are presented below.

2.2.4.3.1 Gradient-Based Methods

Let \mathbf{x} denote the hidden states of an RNN, \mathbf{u} denote the inputs to the RNN, \mathbf{w} denote the parameters of the RNN, and \mathbf{y} denote the outputs of the RNN. Almost any RNN architecture can be expressed in the form shown in Equation 2.95 (Nerrand et al., 1994). The goal of network training is to minimize some measure of the error between the outputs predicted by the network and the desired outputs (call it \mathbf{d}). Typically, the sum of squared error (SSE), as shown in Equation 2.96, is used as the minimization criterion.

$$\begin{aligned}\mathbf{x}(k+1) &= f(\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}) \\ \mathbf{y}(k) &= g[\mathbf{x}(k)]\end{aligned}\tag{2.95}$$

$$J = \sum_i \sum_k [\mathbf{y}_i(k) - \mathbf{d}_i(k)]^2\tag{2.96}$$

Gradient-based methods require the gradient of the error function J , to find the optimal values of the parameters \mathbf{w} . The popular backpropagation algorithm, which calculates the derivatives of the error function for multilayer FFN using the chain rule of differentiation, can be extended to RNNs by using backpropagation through time (BPTT) (Werbos, 1993). BPTT approximates the time evolution of an RNN as a sequence of static networks and then updates the network parameters using the standard backpropagation algorithm. Figure 2.12 shows an example of unfolding the hidden layer of an RNN through four steps in time. A single layer of a simple RNN (no inputs or outputs have been shown to simplify representation) is shown on the left. Moreover, self-connections have not been shown to avoid confusion; that is, other than the connections shown explicitly, it is assumed that each node of this layer is connected to itself through a recurrent connection.

The multilayer FFN on the right of Figure 2.12 is assumed to be equivalent to the RNN on the left. It should be noted that the weights between layers for the equivalent network are constrained to be the same for all layers. To achieve this, BPTT updates all equivalent weights using the sum of the gradients obtained for weights in equivalent layers, which may be shown to be the exact gradient of the error function

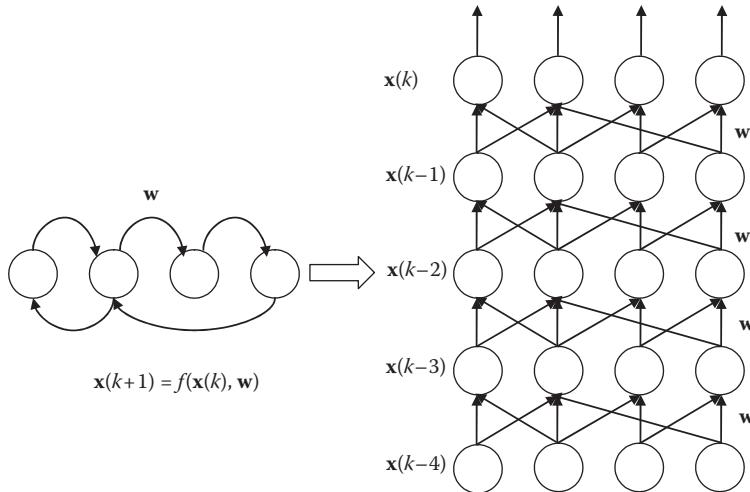


FIGURE 2.12 A single layer of an RNN is shown on the left and an equivalent representation of the network unfolded in time through four steps is shown on the right. (From Pearlmutter, B., *IEEE Trans. Neural Networks*, 6, 1212, 1995. With permission.)

for the RNN. If a network is unfolded only for a fixed number of steps, the derivative calculated using BPTT is not exact. Real-time recurrent learning (RTRL) is an alternative procedure for determining the derivatives of functions of the states of a dynamic system with respect to that system's internal parameters online in a stable fashion, but it is computationally expensive (Williams and Zipser, 1989). Second-order methods such as the Gauss–Newton method and the Levenberg–Marquardt method can be used with the obtained gradients to speed increase the convergence rate during training (Norgaard et al., 2000). RTRL and BPTT calculate the same derivative in slightly different fashions and hence both these algorithms suffer from the same drawbacks of gradient-based methods, that is, they are susceptible to getting stuck in local minima of the error function. Moreover, it has been shown that RNNs trained using gradient-based methods are unable to learn long-term temporal dependencies as the errors flowing back in time either decay exponentially or blow up (Hochreiter et al., 2001).

2.2.4.3.2 Filter-Based Methods

The use of nonlinear filtering techniques has also been suggested for training the weights of RNNs. Online versions of gradient-descent algorithms are inefficient when applied to time series because they depend only on instantaneous estimates of the gradient and ignore history information from the past. In order to overcome this deficiency, the parameter training problem can be treated as an optimal (or suboptimal) filtering problem, which recursively finds a least squares solution to the problem of obtaining the best-fit curve to the given data. The dual extended Kalman filter based training method (Puskorius and Feldkamp, 1994) and the unscented Kalman filter based training method (Haykin, 2001) are examples of training

methods based on nonlinear stochastic filtering. The network parameters, \mathbf{w} , are considered as unknown states of a dynamic system and are estimated from data. The dynamic system considered is shown in Equation 2.97.

$$\begin{aligned}\mathbf{w}(k+1) &= \mathbf{w}(k) \\ \mathbf{x}(k+1) &= f(\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}(k)) \\ \mathbf{y}(k) &= g[\mathbf{x}(k)]\end{aligned}\tag{2.97}$$

Sometimes, considering all the weights and hidden states of an RNN together as shown in Equation 2.97 can make the filtering problem unmanageable even for reasonably sized networks. Hence, the filtering process may be decoupled wherein two interacting filters are run in parallel. The first filter only considers the problem of estimating the hidden states and uses the weight estimates from the second filter at the previous time step. The second filter, on the other hand, only considers the network parameters for estimation and uses the hidden state estimates from the first filter. The computational complexity and memory requirements of this method can be further reduced by considering a node-decoupled version of the above method, wherein a separate filter is used to estimate the weights related to each node while all other weights are assumed constant by this filter. Although this filtering process is not optimal, it has been observed that it is still much more efficient (in terms of the number of training patterns required) than the gradient-based methods.

2.2.4.3.3 Derivative Free Optimization-Based Methods

The use of derivative free optimization methods has the following significant advantages in the context of training RNNs:

- They have a higher probability of finding the global optimum of the error function.
- They can be used to simultaneously optimize the structure as well as the parameters of the network.
- They can overcome to a certain degree the problem of error decay/blow up in RNNs since they do not depend on the derivatives.
- They can be applied to different architectures without any change in the optimization routine itself.

Hence, a large number of these methods have been proposed in recent years. Different optimization methods such as genetic algorithms, evolutionary strategies, particle swarm optimization, differential evolution, and their hybrids have been used to solve the optimization problem. The objective function to be minimized could be the same as Equation 2.96 for parameter learning only or it could be a weighted combination of the training error and a measure of the complexity of the network for simultaneous structure and parameter learning. It is also possible to make use of multiobjective optimization to obtain representative samples along the Pareto-optimal front, which indicates the trade-offs between network complexity and accuracy (Delgado and Pegalajar, 2005). Although the results from these applications prove conclusively that direct optimization using derivative free methods is definitely

superior to gradient-based methods for training RNNs, very little can be said about which is the superior method among these.

2.2.4.4 Other Issues

Since the final model obtained through RNN training is dependent entirely on the supplied data, experimental design for data collection is just as critical as the selection of a proper architecture and training method. RNNs offer the highest advantage when modeling nonlinear systems, but nonlinear systems do not obey superposition or homogeneity. Hence, the inputs to the system should be designed so as to excite the entire range of dynamics of interest. This involves generating input signals that cover all amplitudes and frequencies that will be encountered during actual operation of a system. This can be done using chirp signals with varying amplitudes and by using step variations with random magnitudes and widths (Norgaard et al., 2000). A critical issue here is that as the number of inputs to the system increases, their possible combinations explode in an exponential fashion leading to the curse of dimensionality, which is a major limitation of RNNs while modeling systems with a large number of inputs.

2.3 NEURO-FUZZY SYSTEMS

Neuro-fuzzy networks are based on the fusion of ideas from fuzzy inference and NN. They are based on the observation that fuzzy inference systems can be represented as layered FFNN. Neuro-fuzzy networks represent fuzzy inference systems with learning abilities from data. Contrary to ANN, they provide abilities to incorporate expert knowledge expressed as IF-THEN rules and their parameters can be related to those of physical systems. The concept of neuro-fuzzy systems is schematically illustrated in Figure 2.13.

Various neuro-fuzzy systems have been proposed and can be classified depending on the type of fuzzy systems from which the neuro-fuzzy systems are derived.

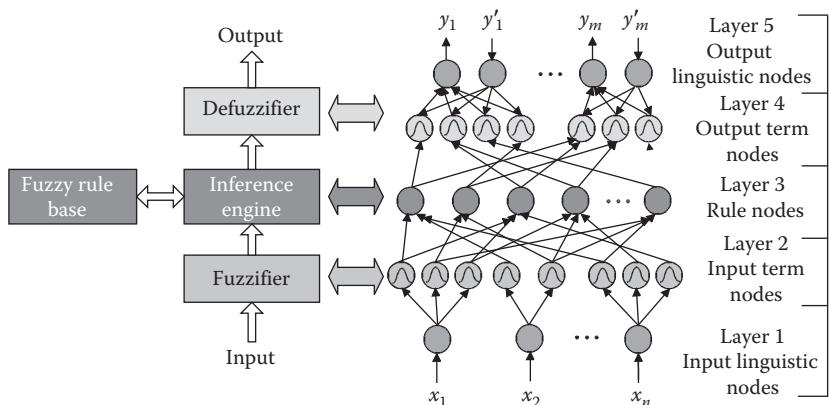


FIGURE 2.13 Neuro-fuzzy system.

Most of them are, however, derived from fuzzy systems whose output MFs are singletons (Wang and Mendel, 1992a,b; Nie and Linkens, 1993; Glorennec, 1994; Shimojima et al., 1995; Koo, 1996) or linear combinations of input variables (Jang, 1993), since they allow the fuzzy systems to be converted to NN of simple structure. Among them, the fuzzy basis function networks (FBFNs), which are similar in structure to RBFNs, have gained much attention (Wang and Mendel, 1992a,b; Nie and Linkens, 1993; Shimojima et al., 1995; Koo, 1996). In addition to their simple structure, FBFNs possess another advantage that they can directly adopt various learning algorithms already developed for RBFNs. It has also been shown that FBFNs are capable of uniformly approximating any continuous nonlinear functions to a prescribed degree of accuracy with a finite number of basis functions (Wang and Mendel, 1992b).

2.3.1 FUZZY BASIS FUNCTION NETWORKS

The FBFN started from an idea that a fuzzy inference system with product inference, singleton output fuzzy MF, a centroid defuzzifier, and Gaussian input MF has a very similar structure to that of RBFN (Wang and Mendel, 1992b). Therefore, FBFN shares many same features with RBFN. The major difference is that FBFN can incorporate heuristic rules into the model and hence can be trained using production rules. In FBFN, fuzzy MFs are used instead, along with fuzzy inferencing, fuzzification, and defuzzification.

Definition of FBFN: Consider a multi-input single-output fuzzy inference system: $X \subset \Re^n \rightarrow Y \subset \Re$. Suppose we have M fuzzy logic rules in the following form:

$$R^j: \text{IF } x_1 \text{ is } A_1^j \text{ AND } x_2 \text{ is } A_2^j \text{ AND } \dots \text{ AND } x_n \text{ is } A_n^j, \text{ THEN } y \text{ is } B^j$$

where A_i^j and B^j are linguistic terms characterized by fuzzy MFs $\mu_{A_i^j}(x_i)$ and $\mu_{B^j}(y)$, respectively, and $j = 1, 2, \dots, M$. It is also assumed that a singleton fuzzifier, product inference, a centroid defuzzifier, Gaussian input MFs, and singleton output MF are used. Then for a given crisp input vector $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in X$, the following defuzzified inferred output can be obtained:

$$y = f(\mathbf{x}) = \frac{\sum_{j=1}^M \bar{y}_j \left[\prod_{i=1}^n \mu_{A_i^j}(x_i) \right]}{\sum_{j=1}^M \left[\prod_{i=1}^n \mu_{A_i^j}(x_i) \right]} \quad (2.98)$$

where

$$f: X \subset \Re^n \rightarrow Y \subset \Re$$

\bar{y}_j is the point in the output space Y at which $\mu_{B^j}(\bar{y}_j)$ achieves its maximum value

$\mu_{A_i^j}(x_i)$ is the Gaussian MF defined by

$$\mu_{A_i^j}(x_i) = \exp \left[-\frac{1}{2} \left(\frac{x_i - m_i^j}{\sigma_i^j} \right)^2 \right] \quad (2.99)$$

where m_i^j and σ_i^j are real-valued parameters.

Hence, FBFs can be defined as

$$p_j(\mathbf{x}) = \frac{\prod_{i=1}^n \mu_{A_i^j}(x_i)}{\sum_{j=1}^M \left[\prod_{i=1}^n \mu_{A_i^j}(x_i) \right]}, \quad j = 1, 2, \dots, M \quad (2.100)$$

It can be observed that the j th FBF can be defined by a nonlinear parameter set $\lambda_j = \{\mathbf{m}^j, \boldsymbol{\sigma}^j\}$, where $\mathbf{m}^j = (m_1^j, m_2^j, \dots, m_n^j)^T \in \Re^n$ and $\boldsymbol{\sigma}^j = (\sigma_1^j, \sigma_2^j, \dots, \sigma_n^j)^T \in \Re^n$ are the center and width vectors of input MFs, respectively. Then, the fuzzy inference system is equivalent to a FBF expansion or FBFN:

$$f(\mathbf{x}) = \sum_{j=1}^M p_j(\mathbf{x}) w_j \quad (2.101)$$

where $w_j = \bar{y}_j \in \Re$ are constants. Therefore, the fuzzy inference system can be viewed as a linear combination of FBFs. Figure 2.14 shows an example of FBFN with four fuzzy rules for three input variables and one output variable.

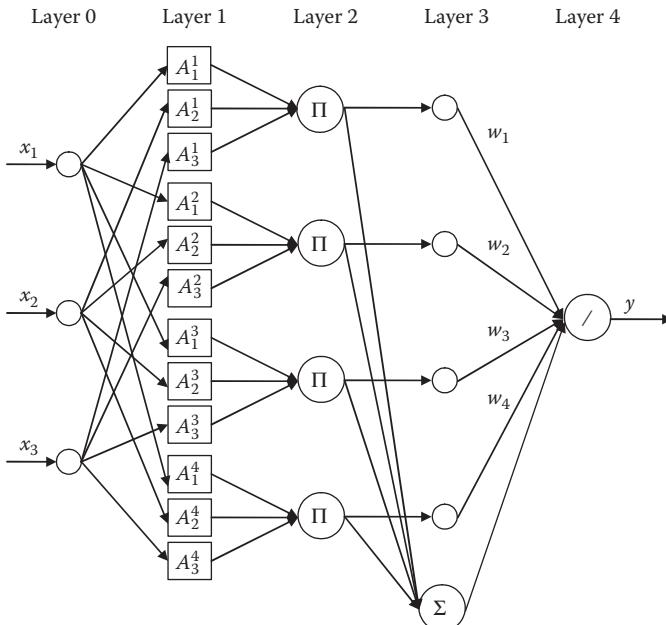


FIGURE 2.14 FBFN with four fuzzy rules.

Assume that N input–output training pairs are given: $(\mathbf{x}(t), d(t))$, $t = 1, 2, \dots, N$. The task of training FBFN is to design a FBFN $f(\mathbf{x})$ such that

$$d(t) = f[\mathbf{x}(t)] + e(t) = \sum_{j=1}^M p_j[\mathbf{x}(t)]w_j + e(t) \quad (2.102)$$

The above equation can be arranged from $h = 1$ to N in matrix form:

$$\mathbf{d} = \mathbf{P}\mathbf{w} + \mathbf{e} \quad (2.103)$$

where

$$\begin{aligned} \mathbf{d} &= [d(1), \dots, d(N)]^T \\ \mathbf{P} &= [\mathbf{p}_1, \dots, \mathbf{p}_M] \text{ with } \mathbf{p}_j = [p_j(1), \dots, p_j(N)]^T \\ \mathbf{w} &= [w_1, \dots, w_M]^T \\ \mathbf{e} &= [e(1), \dots, e(N)]^T \end{aligned}$$

Hence, constructing the FBFN becomes a simple linear least-squares problem once the response vector matrix \mathbf{P} , whose column vectors are response vectors of FBF nodes, is determined. However, the remaining question is how the FBF nodes, specifically the nonlinear parameter sets $\lambda_j = \{\mathbf{m}^j, \mathbf{o}^j\}$, are chosen or determined.

2.3.2 ANFIS

Adaptive neuro-fuzzy inferencing system (ANFIS) is a hybrid of both fuzzy systems and NN to integrate their best features (Figure 2.15). As a special NN, ANFIS can approximate nonlinear systems with less training data, quicker weakening speed, and higher precision. ANFIS takes the advantages of fuzzy systems to represent prior knowledge into a set of constraints (network topology) to reduce the optimization search space, while it also utilizes the features of NN for adaptation of backpropagation to structured network to automate fuzzy system parametric tuning.

Adaptive neural-fuzzy inference system, which is the algorithm first defined by Jang (1993), creates a fuzzy decision tree to classify the data into one of $2n$ (or pn) linear regression models to minimize the SSEs:

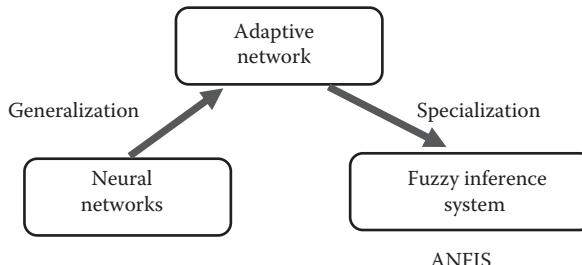


FIGURE 2.15 Basic approach of ANFIS.

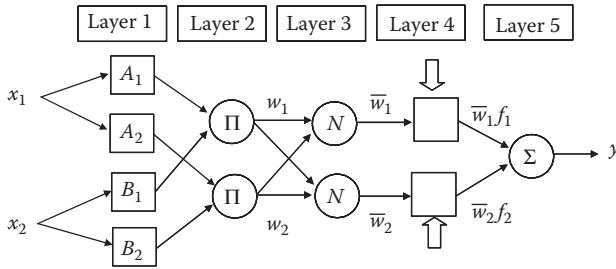


FIGURE 2.16 Structure of ANFIS.

$$SSE = \sum_j e_j^2$$

where

e_j is the error between the desired and the actual output

p is the number of fuzzy partitions of each variable

n is the number of input variables

For simplicity, we will consider two input and one output system. Suppose the fuzzy inference system contains two fuzzy rules of Takagi and Sugeno type, such as

Rule 1: IF x_1 is A_1 AND x_2 is B_1 , THEN $y = f_1(\mathbf{x}) = z_{11}x_1 + z_{12}x_2 + z_{13}$

Rule 2: IF x_1 is A_2 AND x_2 is B_2 , THEN $y = f_2(\mathbf{x}) = z_{21}x_1 + z_{22}x_2 + z_{23}$

Then fuzzy inferencing based on the ANFIS structure shown in Figure 2.16 will produce the output as follows:

$$\begin{aligned} y &= \bar{w}_1 f_1 + \bar{w}_2 f_2 \\ &= \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \end{aligned} \quad (2.104)$$

2.4 MODELING OF DYNAMIC SYSTEMS

Neural networks and neuron-fuzzy networks can also be used to approximate dynamic systems. Static FFN can be used to identify time-invariant dynamic systems. Since FFN have no dynamic memory, all the terms with time delay are used as input to the networks to represent a dynamic system. This increases the number of input neurons and hidden nodes to large numbers, which in turn slows the computation. On the other hand, recurrent networks have embedded connections with time delay and hence result in a more compact representation. They can also be used for modeling of time-variant dynamic systems.

In this section, we will study different approaches to the modeling and identification of dynamic systems using different paradigms of neuron-fuzzy networks.

2.4.1 DYNAMIC SYSTEM IDENTIFICATION USING FEEDFORWARD NETWORKS

When a dynamic system is deterministic and time-invariant, single-input single-output (SISO) system, the input–output model can be represented by

$$y(k) = f(y(k-1), y(k-2), \dots, y(k-n), u(k-1), u(k-2), \dots, u(k-m)) \quad (2.105)$$

where $u(k)$ and $y(k)$ represent the input and output pair at time k . Equation 2.105, for example, can be represented by a schematic diagram of the three-layer neuron-fuzzy network as shown in Figure 2.17. It would be possible to extend this to multi-input multi-output system, despite the large network size.

A dynamic system can also be represented in state-space form. The state-space model for a nonlinear system can be written in general

$$\begin{aligned} \mathbf{x}(k+1) &= f(\mathbf{x}(k), \mathbf{u}(k)) \\ \mathbf{y}(k) &= g[\mathbf{x}(k)] \end{aligned} \quad (2.106)$$

where

$\mathbf{x}(k) = [x_1(k), x_2(k), \dots, x_n(k)]^T$ is the vector containing n state variables

$\mathbf{y}(k) = [y_1(k), y_2(k), \dots, y_m(k)]^T$ is the vector containing all the output variables

In Section 2.4.1.1, we introduce a technique to model a nonlinear system in state-space form using an radial basis function neural network (RBFNN).

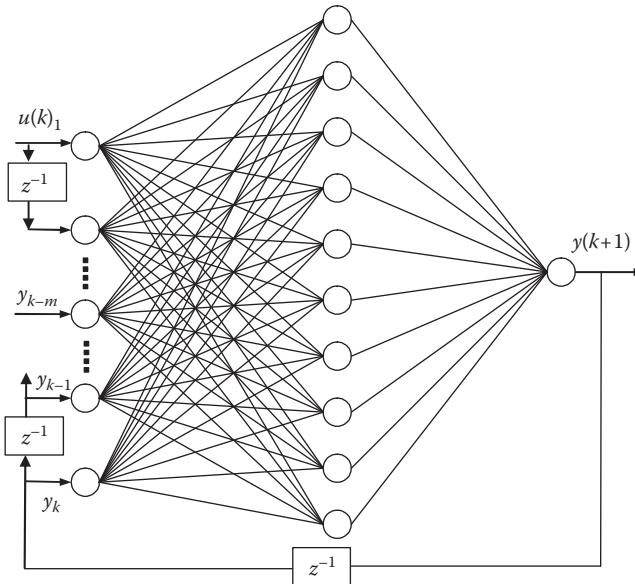


FIGURE 2.17 Representation of a SISO dynamic system by an FFNN.

2.4.1.1 Dynamic System Modeling by Radial Basis Function Neural Network

During training, the dynamic portion of the nonlinear state-space model given in Equations 2.106 can now be represented using the RBFNN as, including a noise term (for approximation error),

$$x_{k+1}^i = [\Lambda \Lambda_0] \begin{bmatrix} \Psi(X_k^i) \\ X_k^i \end{bmatrix} + w_k \quad (2.107)$$

$$z_k^i = x_k^i + \zeta_k^i \quad (2.108)$$

where the superscript i denotes the experiment index. It is assumed that the initial conditions x_0^i belong to the domain D . Note that X_k contains the state variables and input, and $\Psi(X_k^i) = [\phi_1(X_k^i), \dots, \phi_p(X_k^i)]^T$ contains the basis functions corresponding to p centers. Each row of the matrices Λ and Λ_0 correspond to an element of the vector function $\hat{f}(.)$. Figure 2.18 shows a schematic diagram for approximating the complete dynamic stochastic system given by Equations 2.106.

If the vector $\theta_j^T, j = 1, \dots, n$, is used to denote the j th row of the matrix $[\Lambda \Lambda_0]$, Equation 2.107 can be rewritten as

$$x_{k+1}^i = \begin{bmatrix} \Psi_{ik}^T & 0 & \dots & 0 \\ 0 & \Psi_{ik}^T & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & \Psi_{ik}^T \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} + w_k \quad (2.109)$$

for $i = 1, \dots, M$. The notation $\Psi_{ik} \triangleq [\Psi^T(X_k^i) X_k^i]^T$ is used in Equation 2.109. Further simplification of the notation using $\Theta = [\theta_1 \theta_2 \dots \theta_n]^T$, and combining all M experiments, a system with Mn state variables and np parameters

$$\eta_{k+1} = \xi_k(\eta_k, u_k)\Theta + w_k \quad (2.110)$$

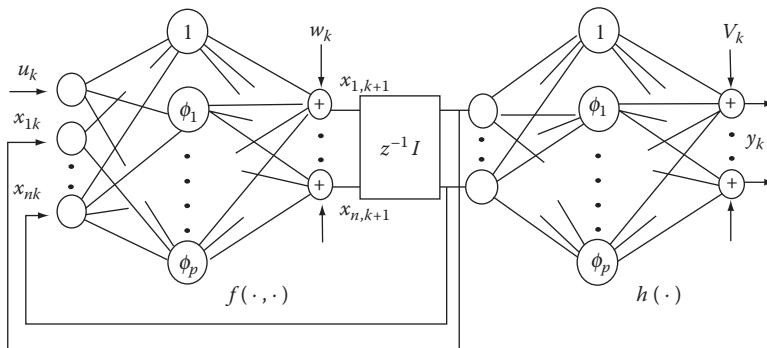


FIGURE 2.18 State-space form representation of a dynamics system with an RBFN.

with measurements

$$\gamma_k = \eta_k + \zeta_k \quad (2.111)$$

is obtained, where $\eta_k^T = [x_{1k}^1, x_{1k}^2, \dots, x_{1k}^M, x_{2k}^1, \dots, x_{nk}^M]$. In the following, knowledge of the statistical properties of ζ_k is not assumed. The matrix

$$\xi_k(\eta_k, u_k) = \begin{bmatrix} \begin{pmatrix} \Psi_{1k}^T \\ \vdots \\ \Psi_{Mk}^T \end{pmatrix} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & \begin{pmatrix} \Psi_{1k}^T \\ \vdots \\ \Psi_{Mk}^T \end{pmatrix} \end{bmatrix} \quad (2.112)$$

is of dimension $Mn \times np$.

2.4.1.1.1 Training of the RBFNN

Once the structural functions of the plant (Equation 2.106) have been replaced with an equivalent system of RBFs, the training algorithm for the RBFNN is formulated as one of the parameter identification. Identification of systems in the form of Equations 2.110 and 2.111 is well known in control theory. Note that, as a result of combining the M experiments, the objective function can be written as

$$J = \sum_{k=1}^N [\gamma_k - \xi_{k-1}(\eta_{k-1}, u_{k-1})\Theta]^T [\gamma_k - \xi_{k-1}(\eta_{k-1}, u_{k-1})\Theta] \quad (2.113)$$

Equating the gradient with respect to Θ to zero gives the least-squares estimation rule

$$\hat{\Theta}_N = \left(\sum_{k=2}^N \xi_{k-1}^T \xi_{k-1} \right)^{-1} \left(\sum_{k=2}^N \xi_{k-1}^T \gamma_k \right) \quad (2.114)$$

However, the strict validity of Equation 2.114 for parameter identification of non-linear systems cannot be uniformly guaranteed. Here, additive noise terms in both state and output measurements are considered. In order to improve convergence of the parameter estimation, the algorithm outlined by Knapp and Pal (1983) is adopted here:

1. Since η_k is not exactly measurable, replace the matrix $\xi_{k-1}(\eta_{k-1}, u_{k-1})$ with $\hat{\xi}_{k-1}(\gamma_{k-1}, u_{k-1})$.
2. An instrumental variable step of replacing ξ_{k-1}^T by ξ_{k-2}^T ensures consistent estimates of the parameter Θ . As a result, the estimation rule (Equation 2.114) can be rewritten as

$$\hat{\Theta}_N = \left(\sum_{k=3}^N \hat{\xi}_{k-2}^T \hat{\xi}_{k-1} \right)^{-1} \left(\sum_{k=3}^N \hat{\xi}_{k-2}^T \gamma_k \right) \quad (2.115)$$

It is known (Knapp and Pal, 1983) that the convergence of the least-squares algorithm ($\hat{\Theta}_N \rightarrow \Theta$ in probability) depends on the satisfaction of two additional conditions, namely,

$$\frac{1}{N} \sum_{k=3}^N \hat{\xi}_{k-2}^T \hat{\xi}_{k-1}^T \rightarrow \sum \quad (2.116)$$

where \sum , an $r \times r$ matrix dependent on Θ , is nonsingular with finite norm and

$$\frac{1}{N} \sum_{k=3}^N \hat{\xi}_{k-2} \left[\gamma_k - \hat{\xi}_{k-1} \Theta \right] \rightarrow 0 \quad (2.117)$$

These conditions are difficult to verify for a general matrix ξ_k , but they will be assumed to be true. The recursive version of algorithm (Equation 2.115) is written as follows:

$$\hat{\Theta}_{N+1} = \hat{\Theta}_N + R_N \hat{\xi}_{N-1}^T \left(I + \hat{\xi}_N R_N \hat{\xi}_{N-1}^T \right)^{-1} \left(\gamma_{N+1} - \hat{\xi}_N \hat{\Theta}_N \right) \quad (2.118)$$

where $R_{N+1} = R_N - R_N \hat{\xi}_{N-1}^T \left[I + \hat{\xi}_N R_N \hat{\xi}_{N-1}^T \right]^{-1} \hat{\xi}_N R_N$. Faster convergence of the training algorithm is obtained by choosing $R_0 = \sigma I$, with a sufficiently large value of σ . Approximation of the static output equations can be carried out in an analogous fashion.

2.4.1.1.2 Approximation Error Estimation

In order to use the RBFNN for state estimation, the maximum error in approximation has to be estimated. Define the maximum error in approximating $f(\cdot, \cdot)$ as

$$e_f = \| f(X) - \hat{f}(X) \|_\infty, \quad X \in D \quad (2.119)$$

where $\|e_1, \dots, e_n\|_\infty = \max\{|e_1|, \dots, |e_n|\}$. When the process and measurement noises are small, an estimate of e_f can be obtained from the M experiments using

$$e_f = \max \| \gamma_k - \hat{\xi}_{k-1}(\gamma_{k-1}, u_{k-1}) \hat{\Theta} \|, \quad k = 1, 2, \dots \quad (2.120)$$

Similarly, the constant e_h is defined as the approximation error for the output equation. To proceed, let system in Equation 2.107 be rewritten as

$$x_{k+1} = f'(x_k, u_k) + \mathbf{F}x_k + \mathbf{b}u_k + \mathbf{w}_k \quad (2.121)$$

where matrix F and vector b are obtained from the matrix Λ_0 . The approximated output equations can also be rewritten as

$$y_k = h'(x_k) + \mathbf{H}x_k + \mathbf{v}_k \quad (2.122)$$

Two Lipschitz constants a and d are defined as

$$\|f'(x_k, u_k) - f'(x_k + \delta_k, u_k)\|_\infty \leq a\|\delta_k\|_\infty \quad (2.123)$$

and

$$\|h'(x_k + \delta_k) - h'(x_k)\|_\infty \leq d\|\delta_k\|_\infty \quad (2.124)$$

These constants a and d can also be estimated using the training set as was done with the constants e_f and e_h .

The training algorithm for the RBFNN is summarized as follows:

1. Choose \mathbf{p} vectors from among the M training experiments in a random fashion. For the dynamic portion, vectors of dimension $n + 1$ are chosen at random times k .
2. Based on the M training experiments, and, using Equation 2.108, the network parameters for the dynamic and static equations are identified.
3. Upper bound errors of approximation are estimated from the training data as given by Equation 2.120. Lipschitz constants a and d are estimated using Equations 2.123 and 2.124.

2.4.2 DYNAMIC SYSTEM REPRESENTATION BY RECURRENT NEURAL NETWORKS

Contrary to an FFNN that can only map an algebraic input–output relationship, a recurrent neuro-fuzzy network as shown in [Figure 2.19](#) can be used to model the dynamics of a nonlinear system. With this recurrent structure, the dynamics of a nonlinear system can be adequately modeled by the dynamic IF-THEN rules embedded in the recurrent neuro-fuzzy rule-base structure. Hence, this changes the difficult modeling process into one of training the feedforward/recurrent neuro-fuzzy systems. That is, the input–output data from the complex nonlinear system can be used to train the recurrent neuro-fuzzy system, and the trained recurrent neuro-fuzzy system will closely approximate the dynamics of the complex system. A thorough survey have been conducted by Tsoi et al. (1994, 1997) on most existing recurrent architectures.

2.4.3 STATE OBSERVER CONSTRUCTION

2.4.3.1 State Estimation Using RBFNN

To estimate the states of a system represented by the RBFNN, a nonlinear filter is to be designed. Performance of the filtering system depends on the accuracy of the

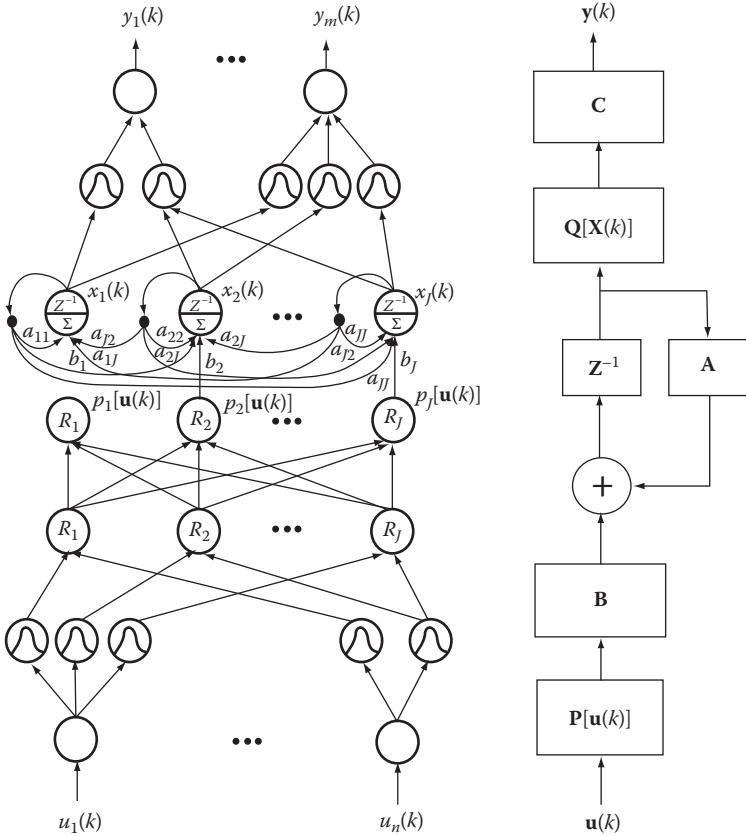


FIGURE 2.19 Recurrent FBFN-based neuro-fuzzy network.

model used. If modeling inaccuracies are not included in the covariance calculations, the calculated covariance matrix becomes unrealistically small, so that undue confidence is placed in the estimates (Fitzgerald, 1971). The estimator ignores subsequent measurements, which results in large estimation errors. This divergence problem can be minimized by accounting for modeling errors.

In order to circumvent this problem, we will use information about the maximum approximation errors in the covariance calculations. An upper bound covariance matrix is obtained, such that minimizing the estimated variance results in minimizing the true variance. For convenience, the approximated system using the RBFNN is rewritten as

$$x_{k+1} = f'(x_k, u_k) + \mathbf{F}x_k + \mathbf{b}u_k + \mathbf{w}_k \quad (2.125)$$

$$y_k = h'(x_k) + \mathbf{H}x_k + \mathbf{v}_k \quad (2.126)$$

Consider a state estimator of the following type:

$$\hat{x}(k+1) = f'(\hat{x}_k, u_k) + \mathbf{F}\hat{x}_k + \mathbf{b}u_k + K_k[y_k - h'(\hat{x}_k) - \mathbf{H}\hat{x}_k] \quad (2.127)$$

Define the estimation error as $\tilde{x}_k = x_k - \hat{x}_k$ and the error in approximating the system dynamic and static equations by $\tilde{f} \stackrel{\Delta}{=} f(x_k, u_k) - f'(x_k, u_k) - Fx_k - bu_k$ and $\tilde{h} \stackrel{\Delta}{=} h(x_k) - h'(x_k) - Hx_k$. The true covariance matrix of the system is denoted by $\rho_k \stackrel{\Delta}{=} E(\tilde{x}_k \tilde{x}_k^T)$. Based on the derivation given in Elanayar and Shin (1994), a recursive upper bound for the covariance matrix is given by the following equation:

$$\begin{aligned} \hat{P}_{k+1} = & l_1(F - K_k H)\hat{P}_k(F - K_k H)^T + l_2 I + l_3 \text{Tr}(\hat{P}_k)I + l_4 K_k K_k^T \\ & + l_5 \text{Tr}(\hat{P}_k)K_k K_k^T + W + K_k V K_k^T \end{aligned} \quad (2.128)$$

with $\hat{P}_0 = \rho_0$. The constants $l_1 - l_5$ are as defined in Elanayar and Shin (1994), and ρ_0 is the known initial covariance matrix. To obtain the minimum variance gain matrix K_k^* , we set the first variation of \hat{P}_{k+1} with respect to K_k to 0 (Aoki, 1989). Thus,

$$-l_1(F - K_k H)\hat{P}_k H^T \delta K_k^T + K_k [l_4 I + l_5 \text{Tr}(\hat{P}_k)I + V] \delta K_k^T = 0 \quad (2.129)$$

Since δK_k is arbitrary, the optimal gain is given by

$$K_k^* = F\hat{P}_k H^T \left\{ \left[\frac{l_4}{l_1} + \frac{l_5}{l_1} \text{Tr}(\hat{P}_k) \right] I + \frac{1}{l_1} V + H\hat{P}_k H^T \right\}^{-1} \quad (2.130)$$

The following theorem based on the results in Gusak (1981) establishes the required properties for the assumed covariance equations. In the following, the matrix inequality $Q_1 \geq Q_2$ is used to mean that $Q_1 - Q_2$ is positive semidefinite (PSD).

THEOREM 2.1

Given the existence of an approximated dynamic system in Equations 2.125 and 2.126, which satisfies approximation error conditions of Equation 2.120, and Lipschitzian continuity conditions in Equations 2.123 and 2.124, then

1. $\hat{P}_k \geq \rho_k$
2. $\hat{P}_{k+1}(K_k) \geq \hat{P}_{k+1}(K_k^*) \geq \rho_{k+1}$, where the optimal gain is given by Equation 2.130

Proof To prove (1), consider the inequality obtained from Equation 2.128 and Equation 36 of Elanayar and Shin (1994).

$$\begin{aligned} \hat{P}_{k+1} - \rho_{k+1} \geq & l_1(F - K_k H)(\hat{P}_k - \rho_k)(F - K_k H)^T + l_3 \text{Tr}(\hat{P}_k - \rho_k)I \\ & + l_5 \text{Tr}(\hat{P}_k - \rho_k)K_k K_k^T \end{aligned} \quad (2.131)$$

Since $\hat{P}_0 = \rho_0$, and the right-hand side of the above inequality is PSD, (1) follows by induction. Note that, the right-hand side of the above inequality is PSD for any gain matrix. Part (2) of the theorem follows from the above argument and derivation of the optimal gain by minimizing \hat{P}_{k+1} . ■

The above derivation of a state estimator for the RBFNN applies to very general nonlinear stochastic systems. For a wide class of problems encountered in practice, the output equations occur in a linear fashion. Furthermore, uncertainties in the output equation are usually less severe than in the dynamic model equations (Maybeck, 1982). For such applications, a simplified filter can be derived as was done in the more general case. Consider an output equation that is linear and known accurately

$$y_k = Hx_k + v_k \quad (2.132)$$

In this case, the estimator is specified by the equations

$$K_k^* = (2 + e_f)F\hat{P}_kH^T [(2 + e_f)H\hat{P}_kH^T + V]^{-1} \quad (2.133)$$

and

$$\begin{aligned} \hat{P}_{k+1} &= (2 + e_f)(F - K_kH)\hat{P}_k(F - K_kH)^T + a(1 + a + e_f)\text{Tr}(\hat{P}_k)I \\ &\quad + ne_f(1 + e_f + a)I + W + K_kV\hat{P}_k^T \end{aligned} \quad (2.134)$$

In this section, a state estimator for the RBFNN based on the upper bound errors in approximation has been derived. Additional terms in the gain equation result from taking into account these modeling errors. Section 2.4.3.2 presents a few example applications of the RBFNN to approximate the static and dynamic equations and to subsequent state estimation.

2.4.3.2 Example Applications of the RBFNN State Estimator

In this section, application of the method to three practical examples is demonstrated. It is not possible to compare the method with any existing methods because of the nature of assumptions made in deriving the estimator. In each of the following examples, the number of basis in the RBFNN was chosen to be 75. Training was carried out until the mean squared errors were below a prespecified value.

Example 2.11

This example considers the estimation of the altitude and velocity of a vertically falling body with an unknown ballistic coefficient (Maybeck, 1982). Radar measurements are corrupted with additive gaussian white noise. The nonlinear model and output equations are as shown below:

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} x_{1k} - Tx_{2k} \\ x_{2k} - 3Tx_{2k}^2 e^{-0.05x_{1k}} \end{bmatrix}$$

$$y_k = \sqrt{10,000 + x_{1k}^2} + v_k$$

The states x_1 and x_2 denote altitude (kft) and velocity (kft/s), respectively. The sampling interval was chosen as $T=0.125$ s and variance of the measurement noise is $V=0.01$ (kft)². Note that the model is inputless and contains no process noise. Five experiments with simulated data were used in performing the training of the dynamic and static equations. These experiments used initial conditions randomly obtained from the intervals $x_{10} \in (297, 303)$, and $x_{20} \in (14, 26)$. A value of 5 was used for the constant c appearing in the Hardy multiquadratics. The estimated values of the constants a , d , e_f , e_h were 2.006, 7.54, 3.31, and 3.21, respectively. After training, the RBFNN states were estimated using the initial conditions $x_0 = \hat{x}_0 = (300 \text{ kft}, 20 \text{ kft/s})$ and initial covariance matrix

$$\rho_0 = \hat{\rho}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$$

Figure 2.20 portrays the estimates of x_1 and x_2 for a single run. It demonstrates the accuracy of the state estimation scheme with the RBFNN. It is to be noted that although stability of the matrix equation for the upper bound covariance matrix in Equation 2.128 cannot be ensured, the estimator gain matrix was observed to converge to a constant matrix in just a few iterations. To prevent the covariance matrix from becoming unbounded, suitable upper bounds were placed on its elements. Figure 2.20 shows the results of state estimation when the initial conditions of the estimator (\hat{x}_0) were chosen as $(0, 0)$, while the true initial conditions of the system (x_0) remained unchanged at the values given above.

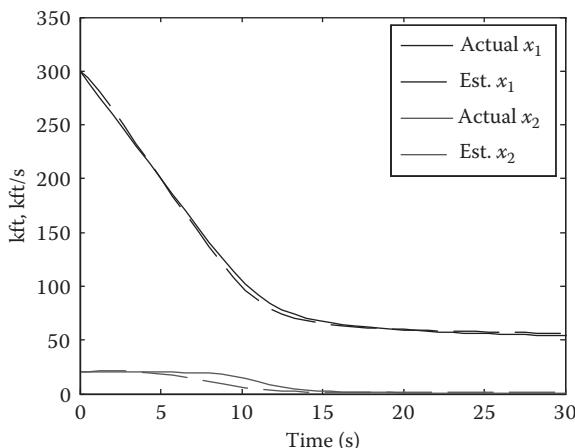


FIGURE 2.20 Estimation with initial condition (300, 20).

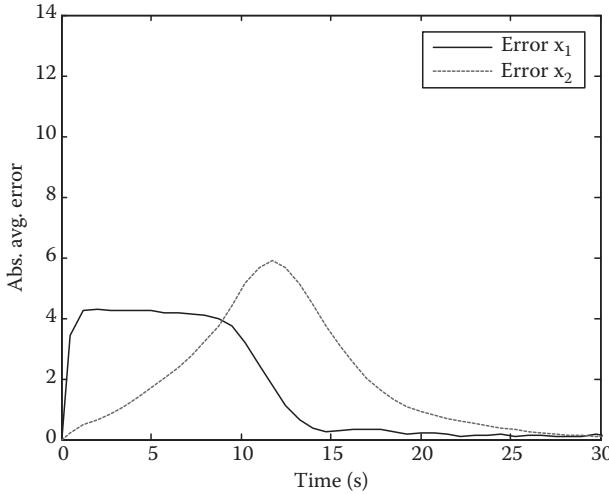


FIGURE 2.21 Absolute average estimation error for 50 experiments.

Next, a Monte Carlo analysis using 50 runs (each with different initial conditions uniformly distributed in the range (297–303, 14–26)) was performed. For each case, the error vector ($x_k - \hat{x}_k$) was generated at all sampling times, and the absolute value of the average errors at these times was obtained. Figure 2.21 shows a plot of the absolute average error in estimating both the state variables. The plot shows similar but improved results over those obtained by using a second-order filter from a known model (Maybeck, 1982). It is worth pointing out that addition of pseudonoise w_k in estimation would help in further tuning the state estimator.

Example 2.12

The second example chosen is that of a nonisothermal, nonadiabatic stirred reaction with an irreversible first-order reaction (Patwardhan et al., 1989). In this case, the nature of experiments that can be performed on the system is assumed to be of a restricted type. The second-order system is given by

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} x_{1k} + T[-x_{1k} + 0.05(1 - x_{1k})e^{x_{2k}}] \\ x_{2k} + T[-x_{2k} + 0.4(1 - x_{1k})e^{x_{2k}} - ux_{2k}] \end{bmatrix}$$

with an assumed measurement equation given by $y_k = [0 \ 1]x_k + v_k$, such that variance of v_k , $V = 0.001$. Here, x_1 is the conversion, and x_2 is the reactor temperature. The sampling interval was chosen to be 0.005 s. It was assumed that only constant inputs could be used in identifying the physical process. In this example, five experiments were conducted with identical initial conditions of $x_0 = (1, 1)$, but with inputs given by $u_k = u_0 + \beta_k$. Nominal inputs of 1.0, 1.5, 2.0, 2.5, and 3.0 were used for u_0 , where β_k is white gaussian noise of variance 0.001.

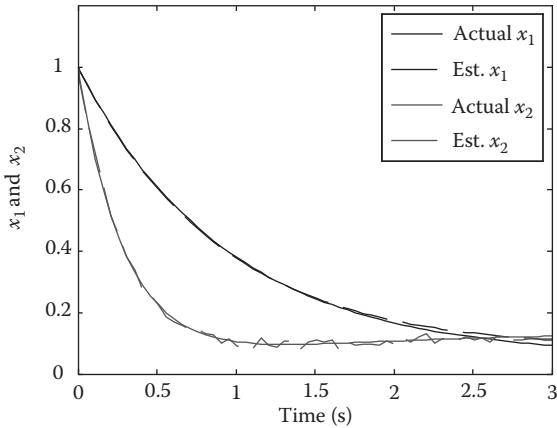


FIGURE 2.22 State estimates of x_1 and x_2 . Estimation with initial condition (1,1).

The inputs were chosen so as to span the region (1.0, 3.0). In this example, the RBFNN training was performed to approximate the function over a region in (x_1, x_2, u) space. The value of the multiquadric constant c was chosen as 0.005. During state estimation, the output equation was assumed to be known and hence Equations 2.133 and 2.134 were used. The constants a and e_f were estimated as 2.766 and 0.8967, respectively. Figure 2.22 depicts the results of the state estimation when an input sequence of the form $u_k = 2.25 + \beta_k$ was used. Initial conditions of (1, 1) and $\rho_0 = 0.1I$ were used. The results show that state estimation is accurate even for input sequences not used during the training stage.

Example 2.13

The final example is that of an inverted pendulum driven by an armature controlled DC motor governed by the equations (Walcott et al., 1987):

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \\ x_{3,k+1} \end{bmatrix} = \begin{bmatrix} x_{1k} + Tx_{2k} \\ x_{2k} + T(9.8 \sin x_{1k} + x_{3k}) \\ x_{3k} - 10T(x_{2k} + x_{3k}) \end{bmatrix}$$

where $T = 0.01$ s, and output $y_k = x_{1k} + v_k$. The state variables x_1 , x_2 , and x_3 are in this case angular position, velocity, and motor current, respectively. Strength of the measurement noise was assumed to be $V = 0.001$. To train the RBFNN, five experiments were used with initial conditions such that $x_{10} \in ((\pi - 1)/2, (\pi + 1)/2)$, and $x_{20}, x_{30} = 0$. During state estimation, numeric values of $a = 38.6$, $e_f = 0.3106$, $c = 0.01$ were used. Figure 2.23 shows the results of state estimation using the following initial values:

$$x_0 = \hat{x}_0 = \begin{pmatrix} \frac{\pi}{2} \\ 0 \\ 0 \end{pmatrix}, \quad \rho_0 = \hat{\rho}_0 = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

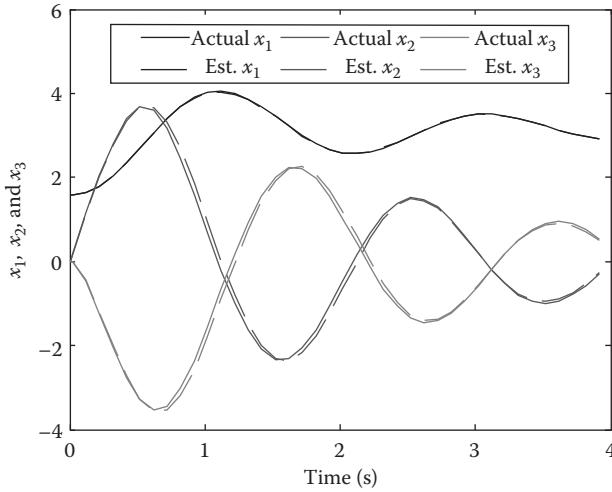


FIGURE 2.23 Estimation with initial condition $(1.57, 0, 0)$.

Estimation of the states is seen to be highly accurate for all three states. As a second variation, the initial conditions of the estimator states were assumed to be $(0, 0, 0)$, while the system remained unchanged with the same initial conditions as above. Figure 2.24 shows that the state estimation performs well even under these conditions. The initial covariance matrix for this case was the same as above, but with $\hat{P}_0(1, 1) = 1$.

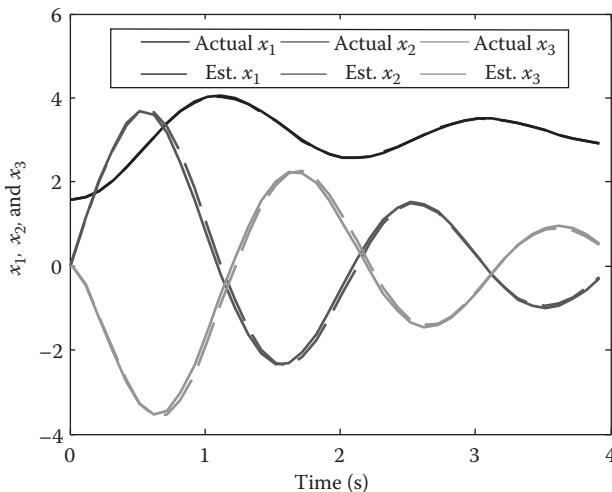


FIGURE 2.24 Estimation with initial condition $(0, 0, 0)$.

2.5 CONCLUSIONS

This chapter covered some of the soft computing tools for modeling of nonlinear systems to construct intelligent systems. Fuzzy logic and different paradigms of NN are commonly used tools for approximating nonlinear systems when analytical models are not available or difficult to obtain based on first principles. Neuro-fuzzy systems are a hybrid concept of NN and fuzzy logics, drawing the advantages of both worlds. Both static and dynamic structures of neuron-fuzzy systems have been described.

This chapter also presented an RBFNN approach to approximately represent general nonlinear stochastic systems in prespecified domains of the state and input space. Multiple experiments are used to train the network. A parameter identification approach to training the network has been presented. Since the parameters appear linearly in the RBFNN, least-squares estimation is possible. The trained network can then be used to predict states for different input sequences and initial conditions than those used during training.

A state estimator has been designed for use with the RBFNN. The gain matrix has been derived on the basis of an upper bound covariance matrix. Explicit inclusion of the approximation error into the estimation algorithm helps in minimizing filter divergence. Three applications of the method to estimate the states of nonlinear systems have been presented. The accuracy of state estimation using the RBFNN has been shown to be very good even when there are uncertainties in the knowledge of initial conditions.

REFERENCES

- Albus, J.S., A new approach to manipulator control: The cerebellar model articulation controller, *Transactions on ASME, Journal of Dynamic Systems, Measurements, and Control*, 97: 220–227, September 1975.
- Aoki, M., *Optimization of Stochastic Systems: Topics in Discrete-Time Dynamics*, Academic Press, San Diego, CA, 1989.
- Broomhead, D.S. and Lowe, D., Multivariable functional interpolation and adaptive networks, *Complex Systems*, 2: 321–355, 1988.
- Carpenter, G.A. and Grossberg, S., The ART of adaptive pattern recognition by a self-organizing neural network, *Computer*, 21(3): 77–88, March 1988.
- Chen, S., Cowan, C.F.N., and Grant, P.M., Orthogonal least squares learning algorithm for radial basis function networks, *IEEE Transactions on Neural Networks*, 2(2): 302–309, 1991.
- Cichocki, A. and Unbehauen, R., *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons, Chichester, United Kingdom, 1993.
- Cybenko, G., Approximations by superposition of a sigmoidal function, *Mathematics of Control, Signals, and Systems*, 2: 303–314, 1989.
- Delgado, M. and Pegalajar, M.C., A multiobjective genetic algorithm for obtaining the optimal size of a recurrent neural network for grammatical inference, *Pattern Recognition*, 38 (9): 1444–1456, 2005.
- Duchon, J., Splines minimizing rotation-invariant semi-norms in Sobolev spaces, *Constructive Theory of Functions of Several Variables (Lecture Notes in Math 571)*, Springer, New York, pp. 85–100, 1977.

- Elanayar, S.V.T. and Shin, Y.C., Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems, *IEEE Transactions on Neural Networks*, 5(4): 594–603, July 1994.
- Elman, J.L., Finding structure in time, *Cognitive Science*, 14: 179, 1990.
- Fitzgerald, R.J., Divergence of the Kalman filter, *IEEE Transactions on Automatic Control*, AC-16: 736–747, December 1971.
- Funahashi, K., On the approximate realization of continuous mappings by neural networks, *Neural Networks*, 2: 183–192, 1989.
- Glorennec, P.Y., Learning algorithms for neuro-fuzzy networks, *Fuzzy Control Systems*, Kandel, A. and Langholz, G. (Eds.), CRC Press, Boca Raton, FL, pp. 3–18, 1994.
- Gusak, P.P., Upper bound for the RMS filtering performance criterion in quasilinear models with incomplete information, *Automation and Remote Control*, 42: 70–76, April 1981.
- Hardy, R.L., Multiquadric equations of topography and other irregular surfaces, *Journal of Geophysical Research*, 76: 1905–1915, 1971.
- Haykin, S., *Kalman Filtering and Neural Networks*, Wiley and Sons, New York, 2001.
- Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J., Gradient flow in recurrent nets: The difficulty of learning long term dependencies, *A Field Guide to Dynamical Recurrent Neural Networks*, Kremer, S.C. and Kolen, J.F. (Eds.), IEEE Press, New York, 2001.
- Hopfield, J.J., Neural networks and physical systems with emergent collective computational capabilities, *Proceedings of the National Academy of Sciences*, 79: 2554–2558, 1982.
- Jackson, I.R.H., Convergence properties of radial basis functions, *Constructive Approximation*, 4: 243–264, 1988.
- Jang, J.-S.R., ANFIS: Adaptive-network-based fuzzy inference system, *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3): 665–685, 1993.
- Jordan, M., Generic constraints on underspecified target trajectories, *Proceedings of the International Joint Conference on Neural Networks*, 1: 217, 1989.
- Knapp C.H. and Pal, P.K., Parameter identification in a class of nonlinear systems, *IEEE Transactions on Automatic Control*, AC-28: 497–503, April 1983.
- Koo, T.-K.J., Construction of fuzz linguistic model, *Proceedings of the 35th Conference on Decision and Control*, Kobe, Japan, December, pp. 98–103, 1996.
- Leonard, J.A. and Kramer, M.A., Radial basis function networks for classifying process faults, *IEEE Control Systems Magazine*, 11: 31–38, 1991.
- Maybeck, P.S., *Stochastic Models, Estimation, and Control*, Vol. 2, Academic Press, New York, 1982.
- Medsker, L.R. and Jain, L.C., *Recurrent Neural Networks—Design and Applications*, CRC Press, New York, 2000.
- Minsky, M. and Papert, S., *Perceptron: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, 1960.
- Nerrand, O., Roussel-Ragot, P., Urbani, D., Personnaz, L., and Dreyfus, G., Training recurrent neural networks: Why and how? An illustration in dynamical process modeling, *IEEE Transactions on Neural Networks*, 5(2): 178–184, 1994.
- Nie, J. and Linkens, D.A., Learning control using fuzzified self-organizing radial basis function networks, *IEEE Transactions on Fuzzy Systems*, 1(4): 280–287, 1993.
- Norgaard, M., Ravn, O., Poulsen, N.K., and Hansen, L.K., *Neural Networks for Modeling and Control of Dynamic Systems*, Springer-Verlag, London, United Kingdom, 2000.
- Patwardhan, A.A., Rawlings, J.B., and Edgar, T.F., Model predictive control of nonlinear processes in the presence of constraints, *IFAC Nonlinear Control System Design Symposium*, Capri, Italy, pp. 345–349, 1989.
- Pearlmutter, B., Gradient calculations for dynamic recurrent neural networks: A survey, *IEEE Transactions on Neural Networks*, 6: 1212–1232, 1995.

- Poggio, T. and Girosi, F., Regularization algorithms for learning that are equivalent to multi-layer networks, *Science*, 247: 4–27, February 1990.
- Powell, M.J.D., *Approximation Theory and Methods*, Cambridge University Press, Cambridge, United Kingdom, 1981.
- Puskorius, G.V. and Feldkamp, L.A., Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent neural networks, *IEEE Transactions on Neural Networks*, 5(2): 279–297, 1994.
- Rumelhart, D.E., Hinton, G.E., and Williams, R.J., Learning internal representations by error propagation, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Rumelhart, D.E. and McClelland, J.L. (Eds.), MIT Press, Cambridge, United Kingdom, p. 45, 1986.
- Sanner, R.M. and Slotine, J.-J.E., Gaussian networks for direct adaptive control, *IEEE Transactions on Neural Networks*, 3(6): 837–862, 1992.
- Schagen, I.P., Sequential exploration of unknown multi-dimensional functions as an aid to optimization, *IMA Journal of Numerical Analysis*, 4: 337–347, 1984.
- Shimojima, K., Fukuda, T., and Hasegawa, Y., RBF-fuzzy system with GA based unsupervised/supervised learning method, *Proceedings of 1995 IEEE International Conference on Fuzzy Systems*, Yokohama, Japan, March 20–24, pp. 253–258, 1995.
- Shin, Y.C., Dynamic nonlinear systems, in *Industrial and Manufacturing Systems*, Leondes, C.T. (Ed), Academic Press, San Diego, CA, pp. 345–387, 1998.
- Siegelmann, H.T. and Sontag, E.D., Turing computability with neural nets, *Applied Mathematics Letters*, 4(6): 77–80, 1991.
- Tsoi, A.C. and Back, A.D., Locally recurrent globally feedforward networks—A critical-review of architectures, *IEEE Transactions on Neural Networks*, 5(2): 229–239, 1994.
- Tsoi, A.C. and Back, A.D., Discrete time recurrent neural networks architectures: A unifying review, *Neurocomputing*, 15(3–4): 183–223, 1997.
- Walcott, B.L., Corless, M.J., and Zak, S.H., Comparative study of non-linear state-observation techniques, *International Journal of Control*, 45(6): 2109–2132, 1987.
- Wang, L.X. and Mendel, J.M., Back-propagation fuzzy systems as nonlinear dynamic system identifiers, *Proceedings IEEE 1992 International Conference on Fuzzy Systems*, San Diego, CA, pp. 1409–1418, 1992a.
- Wang, L.X. and Mendel, J.M., Fuzzy basis functions, universal approximation, and orthogonal least-squares learning, *IEEE Transactions on Neural Networks*, 3(5): 807–814, 1992b.
- Werbos, P., *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, Wiley, New York, 1993.
- Werbos, P.J., Beyond regression: New tools for prediction and analysis in the behavioral sciences, Ph.D. Thesis, Applied Mathematics, Harvard University, Cambridge, MA, November, 1974.
- Widrow, B. and Hoff, M.E., Adaptive switching circuits, *WESCON Convention Record*, Part IV, August 23, pp. 96–104, 1960.
- Williams, R. and Zipser, D., A learning algorithm for continually running fully recurrent neural networks, *Neural Computation*, 1: 270–280, 1989.
- Zadeh, L.A., Fuzzy sets, *Information and Control*, 8: 338–353, 1965.
- Zadeh, L.A., The concept of a linguistic variable and its application to approximate reasoning, *Information Science*, 8: 199–257, 1975.

3 Efficient Training Algorithms

Various neuro-fuzzy networks can be trained using various training algorithms. Training algorithms can be categorized, in general, into three categories: supervised, unsupervised, and reinforced learning algorithms. Use of a learning technique depends on the specific paradigm of the neuro-fuzzy networks.

3.1 SUPERVISED ALGORITHM

Supervised learning is a machine learning technique that relies on the input and desired output data to adjust the weights of neural network (NN) or fuzzy network so as to approximate a nonlinear function or relationship. Typical techniques for supervised learning include (Stacey, 1994)

- Perceptron (Minsky and Papert, 1969)
- Adaline
- Madaline (Widrow and Lehr, 1990)
- Backpropagation (BP) (Rumelhart et al., 1986)
- Boltzmann machine (Ackley et al., 1985)

The output variables can be of continuous (regression) or discrete (classification) values. The goal of learning is to predict output values accurately for a given set of input values after the network is exposed to a number of training data.

3.2 UNSUPERVISED ALGORITHM

Unsupervised learning does not require known output values for training of networks. Instead, it utilizes some internal strategies and local information to create a factorial code of data such that output data are statistically independent. Examples for unsupervised learning include (Stacey, 1994)

- Adaptive resonance theory (ART1 and ART2) (Carpenter and Grossberg, 1988)
- Hopfield networks (Hopfield, 1982)
- Bidirectional associative memory (Kosko, 1988)
- Learning vector quantization (Kohonen, 1988)
- Counter propagation (Hecht-Nielsen, 1987).

Unsupervised learning typically treats the data as random variables and builds a joint density model for the data set. Therefore, it can be used in conjunction with Bayesian inference to produce conditional probabilities.

3.3 BACKPROPAGATION ALGORITHM

For N given input–output training pairs, $(\mathbf{x}(k), \mathbf{d}(k))$, $k = 1, 2, \dots, N$, the task of training is to design a multilayer feedforward network $f(\mathbf{x})$ such that

$$\mathbf{d}(k) = F[\mathbf{x}(k)] + e(k) \quad (3.1)$$

where the vector $\mathbf{x}(k)$ represents a pattern of input to the network and the vector $\mathbf{d}(k)$ represents the corresponding target output data. If we assume that the output of the neuro-fuzzy network is given by $\mathbf{y}(k)$, then the goal is to minimize the error represented by

$$\begin{aligned} e_j(k) &= d_j(k) - y_j(k) \\ E &= \frac{1}{2} \sum_{j=1}^n e_j^2 \end{aligned} \quad (3.2)$$

by a gradient descent method to approximate an unknown function. Let us number the units, and denote the weight from unit i to unit j by w_{ji} , define net_j as the total weighted sum of input signals to neuron j , and also define the output from neuron j as o_j . The update of weights can proceed using the gradient decent method such that

$$w_{ji}(k+1) = w_{ji}(k) + \eta \delta_j o_i \quad \text{or} \quad \Delta w_{ji} = \eta \delta_j o_i \quad (3.3)$$

where η is the parameter denoting the learning rate. The derivative of error E with respect to weight w_{ji} is expressed by

$$-\frac{\partial E}{\partial w_{ji}} = \delta_j o_i \quad (3.4)$$

where $\delta_j = -\frac{\partial E}{\partial \text{net}_j}$.

This can be shown using a chain rule such that

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial w_{ji}} \quad (3.5)$$

The first factor shows how the error changes with the input of the unit j and the second part indicates how much changing w_{ji} changes that input. Since $\text{net}_j = \sum_i w_{ji} o_i$,

$$\frac{\partial \text{net}_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_j w_{ji} o_i = o_i \quad (3.6)$$

To calculate $\delta_j = -\frac{\partial E}{\partial \text{net}_j}$, we can apply the chain rule again

$$\delta_j = -\frac{\partial E}{\partial \text{net}_j} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial \text{net}_j} \quad (3.7)$$

For output neuron,

$$\begin{aligned} \delta_j &= \left(\frac{\partial f}{\partial \text{net}_j} \right) (d_j - y_j) \\ &= y_j(1 - y_j)(d_j - y_j) \end{aligned} \quad (3.8)$$

For hidden neurons, each unit j in the hidden layer is connected to each unit q in the output layer with weight w_{qj}^k for $q = 1, \dots, m$. The BP error up to unit j can be computed

$$\begin{aligned} \delta_j^{(k-1)} &= \left(\frac{\partial f}{\partial \text{net}_j} \right) \sum_q w_{qj}^{(k)} \delta_q^{(k)} \\ &= o_j^{(k-1)} \left(1 - o_j^{(k-1)} \right) \sum_{q=1}^m w_{qj}^{(k)} \delta_q^{(k)} \end{aligned} \quad (3.9)$$

3.4 DYNAMIC BACKPROPAGATION

Dynamic NN that have feedforward and feedback connections are useful for modeling dynamics systems as described in Section 2.4. The dynamic backpropagation (DBP) algorithm can be applied to training such dynamic neuro-fuzzy systems. One of the important considerations in DBP is to ensure the asymptotic stability during the learning process. Consider a general form of a dynamic recurrent NN described in Section 2.4.2.

$$\begin{aligned} \mathbf{x}(k+1) &= f(\mathbf{x}(k), \mathbf{w}, \boldsymbol{\Theta}) \\ \mathbf{y}(k) &= g[\mathbf{x}(k)] \end{aligned} \quad (3.10)$$

Let us assume that weight w_{ji} represents a synaptic connection parameter between the i th neuron and j th neuron and θ_i is a threshold at the i th neuron. Therefore, Equation 3.10 can be rewritten as

$$\begin{aligned} x_i(k+1) &= f_i(\mathbf{x}(k), \mathbf{w}_i, \theta_i) \\ \mathbf{y}(k) &= g[\mathbf{x}(k)] \end{aligned} \quad (3.11)$$

where

$$\mathbf{w} = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix} = [\mathbf{w}_1 \quad \cdots \quad \mathbf{w}_n] \quad (3.12)$$

The goal is to achieve an equilibrium point to minimize the following error function:

$$E = \sum_i \sum_k [\mathbf{d}_i(k) - \mathbf{y}_i(k)]^2 \quad (3.13)$$

The gradient descent in E with respect to the change of w_{ij} can be obtained

$$\begin{aligned} \Delta w_{ji} &= -\eta_w \frac{\partial E}{\partial w_{ji}} \\ &= \eta_w \sum_{j=1}^n \sum_{k=1}^m [d_j(k) - y_j(k)] \frac{\partial g_j(\mathbf{x})}{\partial x_j} \frac{\partial x_j}{\partial w_{ji}} \end{aligned} \quad (3.14)$$

where η_w is the learning rate for the synaptic weights. Similarly, the incremental formulation for the somatic parameter θ_i can be given as

$$\begin{aligned} \Delta \theta_i &= -\eta_\theta \frac{\partial E}{\partial \theta_i} \\ &= \eta_\theta \sum_{j=1}^n \sum_{k=1}^m [d_j(k) - y_j(k)] \frac{\partial g_j(\mathbf{x})}{\partial x_j} \frac{\partial x_j}{\partial \theta_i} \end{aligned} \quad (3.15)$$

The incremental terms can be further derived to be (Jin and Gupta, 1999):

$$\Delta w_{ji} = -\eta_w z_i \frac{\partial f_i(\mathbf{x}, w_i, \theta_i)}{\partial w_{ji}} \quad (3.16)$$

$$\Delta \theta_i = -\eta_\theta z_i \frac{\partial f_i(\mathbf{x}, w_i, \theta_i)}{\partial \theta_i} \quad (3.17)$$

where

$$z_i = \sum_{p=1}^n \frac{\partial f_p}{\partial x_i} z_p + \sum_{l=1}^m (d_l - y_l) \frac{\partial g_l(\mathbf{x})}{\partial x_i} \quad (3.18)$$

This updating rule however does not guarantee the stability of the trained NN. Therefore, if the network becomes unstable, the learning must be repeated with a new learning rate or the stability of the network must be checked at each iterative instant. To overcome this limitation, Jin and Gupta (1999) proposed stable learning methods at the expense of more complexity.

3.5 ORTHOGONAL LEAST SQUARES ALGORITHM

Orthogonal least-squares (OLS) algorithm was first proposed by Chen et al. (1991) as an efficient model selection method for linear regression models and is particularly suitable for radial basis function networks (RBFNs). In this OLS algorithm, all

the training data points become the candidate centers of the RBFN and the width of each candidate RBF node is prefixed by a user. Accordingly, the response vectors of all the candidate RBF nodes are generated and then stored. Among $N - l$ response vectors, \mathbf{p}_j 's, from candidate RBF nodes, where l denotes the number of RBF nodes already found, the OLS algorithm sequentially chooses the best RBF node that produces the maximum error reduction measure, which is determined by the squared norm of the projection of \mathbf{d} onto \mathbf{p}_j . At each iteration, the desired output vector \mathbf{d} and the remaining response vectors \mathbf{p}_j 's are updated by an orthogonalization procedure for the next step. The algorithm is briefly described here.

As described in Section 2.2.3, the goal is to achieve an approximate model such that

$$y = f(\mathbf{x}) = w_0 + \sum_{j=1}^M w_j \phi(\mathbf{x} | \lambda_j) \quad (3.19)$$

This can be formulated as a linear regression problem such as

$$d(k) = f[\mathbf{x}(k)] + e(k) = w_0 + \sum_{j=1}^M p_j[\mathbf{x}(k)]w_j + e(k) \quad (3.20)$$

where

$d(k)$ is the desired output

w_j 's are parameters

$p_i(\cdot)$ are regressors such that $p_j(\cdot) = p(\cdot | \lambda_j)$

The above equation is arranged from $k = 1$ to N in matrix form (Chen et al., 1991):

$$\mathbf{d} = \mathbf{P}\mathbf{w} + \mathbf{e} \quad (3.21)$$

where

$$\mathbf{d} = (d(1), \dots, d(N))^T \in \Re^N$$

$$\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_M] \in \Re^{N \times M} \text{ with } \mathbf{p}_j = \{p[\mathbf{x}(1) | \lambda_j], \dots, p[\mathbf{x}(N) | \lambda_j]\}^T$$

$$\mathbf{w} = (w_1, \dots, w_M)^T \in \Re^M$$

$$\mathbf{e} = (e(1), \dots, e(N))^T \in \Re^N$$

Hence, it can be seen that constructing an RBFN becomes a simple linear least-squares problem, once the response matrix \mathbf{P} , whose column vectors are response vectors of the RBF nodes, is determined.

The OLS method utilizes the transformation of the set of \mathbf{p}_i into a set of orthogonal basis vectors, and thus makes it possible to calculate the contribution of each basis vector to the desired output. The basic idea is to begin with an empty hidden layer and add a new RBF successively. The criterion is to choose an RBF that reduces the sum of square errors given by

$$[\text{err}] = \sum_{k=1}^N [d(k) - y(k)]^2 \quad (3.22)$$

The question is how to select the RBF that reduces [err] most. It has been known that the vector \mathbf{p} having the smallest Euclidean distance to \mathbf{d} corresponds to the orthogonal projection of \mathbf{d} onto \mathbf{P} . Therefore, OLS specifically tries to sequentially locate one hidden node at a time, while maximizing the following error reduction measure:

$$[\text{err}] = \|\text{Proj}_{\mathbf{p}^{(l)}} \mathbf{d}^{(l)}\|^2 \quad (3.23)$$

where $\mathbf{p}^{(l)}$ and $\mathbf{d}^{(l)}$ are the response vector of the hidden unit and the desired output vector, respectively. Here, the contribution of the previously found l orthogonal response vectors is subtracted by the orthogonal projection, and the notation $\text{Proj}_{\mathbf{p}} \mathbf{d}$ denotes $\mathbf{p}(\mathbf{d}^T \mathbf{p} / \|\mathbf{p}\|^2)$.

The following procedure describes the OLS. In OLS, an RBF node is added to the network at each iteration. For simplicity, the algorithm is constructed in such a way that it runs until the number of RBF nodes reaches M , which can be set at an arbitrary value. In the algorithm, l denotes the number of hidden nodes found during the past iterations. The superscript on vectors $\mathbf{d}^{(l)}$ and $\mathbf{p}^{(l)}$ indicates that the contribution of the previously found l RBF nodes have been removed by the orthogonal projection from the desired output vector \mathbf{d} and response vectors \mathbf{p} defined in Equation 3.21. In addition, the vectors, \mathbf{s}_h , $h = 1, 2, \dots, l$, are used to store the orthogonalized response vectors found.

Step 1. Initialize $l = 0$, $\mathbf{p}_0^{(0)} = \mathbf{p}_0$, $\mathbf{d}^{(0)} = \mathbf{d}$.

Step 2. Search for a new RBF node that will maximize the error reduction measure.

Step 3. Calculate the error reduction term of $\mathbf{p}_0^{(l)}$ as follows:

$$\begin{aligned} [\text{err}]_0^{(l)} &= \|\text{Proj}_{\mathbf{p}_0^{(l)}} \mathbf{d}^{(l)}\|^2 \\ \text{If } [\text{err}]_0^{(l)} \leq [\text{err}]^{(l)}, \text{ set } \mathbf{s}_{l+1} &= \mathbf{p}^{(l)} \end{aligned} \quad (3.24)$$

Step 4. The parameters of the selected response vector in Step 3 are stored in a table T .

Step 5. Set $l = l + 1$ and update the desired vector $\mathbf{d}^{(l)}$ by eliminating the contribution of previously found RBF nodes as follows:

$$\mathbf{d}^{(l)} = \mathbf{d}^{(l-1)} - \text{Proj}_{\mathbf{s}_l} \mathbf{d}^{(l-1)} \quad (3.25)$$

Step 6. If $l < M$, go to Step 2.

Step 7. Generate a new \mathbf{P} matrix based on the stored parameters in table T . The weight vector \mathbf{w} can be calculated using the following equation:

$$\mathbf{w} = \mathbf{P}^+ \mathbf{d} \quad (3.26)$$

where \mathbf{P}^+ is the pseudoinverse of, \mathbf{P} which can be calculated by using commercial packages such as MATLAB®.

Since the OLS algorithm utilizes the computationally efficient OLS technique, it has attracted the interest of those who pursue a fast and reliable implementation of RBFNs. However, the performance of the RBFN trained by the OLS algorithm can significantly depend on the prefixed width and centers of hidden units. In addition, uniformly setting all the widths of hidden nodes regardless of the position of centers tends to restrict the performance of the OLS in some applications.

Chng et al. (1996) tried to alleviate the above problems by fine tuning nonlinear parameters of hidden units. Although the same OLS procedure was adopted, the parameters of the hidden units were tuned by perturbation and gradient-based algorithms at each iteration of the OLS algorithm. By utilizing the nonlinear parameters of the selected node in the OLS algorithm as an initial point, a recursive gradient-descent algorithm was employed to minimize the following objective function:

$$E = \sum_{k=1}^N [d(k) - y(k)]^2 \quad (3.27)$$

While the gradient-descent algorithm is applied to the nonlinear parameter set of the current hidden nodes selected by the OLS algorithm, the previously found and tuned nonlinear parameters of hidden nodes remain intact. Hence, the adaptive orthogonal least squares (AOLS) algorithm proposed by Chng et al. (1996) indeed is a cascade learning method, which employs the OLS algorithm to supply the initial point to the gradient-descent algorithm. As with all the gradient-descent algorithms, the performance of the AOLS algorithm would depend on the quality of initial guess in the gradient-descent algorithm. Although the OLS algorithm might provide better initial points than randomly chosen ones, the gradient descent algorithm would not be able to completely overcome the effect of nonlinear parameters prefixed by the OLS algorithm.

3.6 ORTHOGONAL LEAST SQUARE AND GENERIC ALGORITHM

3.6.1 OLS LEARNING USING GENETIC ALGORITHM

The orthogonal least square and generic algorithm (OLSGA) developed by Lee (2000) is described in this section. The OLSGA grows an RBFN starting from zero by adding one hidden node at a time to the network while leaving existing hidden nodes of the RBFN unchanged. The objective of most training algorithms for NNs is to minimize the sum of squared training errors or $\|\mathbf{e}\|^2$ in Equation 3.21. Likewise, when searching for a new hidden node in the OLSGA, the new hidden node is determined such that the sum of squared training errors of the RBFN with the new hidden node is minimized. In OLSGA, the genetic algorithm (GA) is employed as a global search algorithm in order to reduce the risk of local minima and premature convergence.

In the j th iteration run of the OLSGA, a GA is run to search for the j th hidden node, which is defined by its parameters, $\boldsymbol{\lambda}_j = \{\mathbf{m}^j, \sigma^j\}$. The parameter set is encoded into a binary string, which becomes an individual in the population of the GA. For example, if the number of input variables is 2, then the nonlinear parameter set is given by $\boldsymbol{\lambda} = \{(m_1, m_2)^T, (\sigma_1, \sigma_2)^T\}$. Suppose a 4-bit resolution is employed, the parameter set can be encoded into a binary string as follows:

$$\begin{array}{cccc}
 m_1 & m_2 & \sigma_1 & \sigma_2 \\
 \Downarrow \\
 a_4a_3a_2a_1|b_4b_3b_2b_1|c_4c_3c_2c_1|d_4d_3d_2d_1
 \end{array}$$

where a_i , b_i , c_i , and d_i ($i = 1, 2, 3, 4$) are binary digits with the values of 0 or 1. The search space of GA is defined according to the input domain boundary of training data. Assume that the domain of i th input variable has been found to be $[\min(x_i), \max(x_i)]$ from the training data, then the domains of m_i and σ_i are given as $[\min(x_i) - p_i, \max(x_i) + p_i]$ and $(0, [\max(x_i) - \min(x_i)]/2)$, respectively, where p_i is a small positive value. By adjusting p_i , the search space of center m_i can be extended over the domain boundary of training data. For example, the p_i value can be chosen at $[\max(x_i) - \min(x_i)]/10$. After the search space is defined, the initial population is created at random in the given space according to the following encoding scheme.

For the encoding of each parameter, the Gray-coding scheme is adopted in order to improve the convergent properties of GA (Michalewicz, 1996). First, the binary-coding is applied to achieve one-to-one linear mapping between real numbers and binary numbers. For example, if $m_1 \in [-3, 3]$, then the corresponding domain of the 4-bit binary number is given by $[0000_2, 1111_2]$. Then, the Gray-coding scheme is applied to the binary code such that any two points next to each other in the problem space such as -0.2 and 0.2 differ by only one bit. For example, -0.2 and 0.2 are represented as 0111_2 and 1000_2 , respectively, which differ by four bits in the binary-coding system. The Gray-coding scheme converts them into 0100_2 and 1100_2 , respectively, which differ by only one bit.

The population of GA comprises a prefixed number of binary strings. The prefixed number is termed population size and each binary string or individual corresponds to the parameter set of a candidate RBF node, $\lambda_j = \{\mathbf{m}^j, \mathbf{o}^j\}$, according to the encoding scheme shown above. The population of GA evolves from a generation to a next generation by undergoing reproduction, crossover, and mutation for a user-specified number of generations. More specific details on the three operators are given at a later part of this section. The fitness function indicates the goodness of an individual string. The individual with the greatest fitness value during the whole GA procedure is chosen at the end of the GA run to become the j th hidden node.

Due to the requirement for the calculation of the fitness values, application of the GA to the training of NNs tends to require a large amount of computing load. The computing burden can become prohibitively large as the size of the network or the number of training data increases. A natural choice for the fitness function would be the negative sum of squared errors or $-\|\mathbf{e}\|^2$. Such direct adoption of the sum of squared errors, however, may lead to a significant amount of computing load. Suppose a response matrix, \mathbf{P} , is calculated based on the existing hidden nodes plus an individual of the population, the error vector for the least-squares fit is obtained as follows:

$$\mathbf{e} = (\mathbf{I} - \mathbf{P}\mathbf{P}^{-1})\mathbf{d} \quad (3.28)$$

where

\mathbf{I} is the identity matrix

\mathbf{P}^{-1} is the pseudoinverse of the response matrix, \mathbf{P}

It can be seen that the number of operations in Equation 3.28 is of order Nj^2 or $O(Nj^2)$ when $N > j$, where N is the number of training data and j is the number of hidden nodes in the RBFN (Strang, 1980). Since the fitness value needs to be evaluated for all the individuals over the whole GA procedure, the total number of operations for finding the j th hidden node can reach $O(Nj^2sg)$ where s and g are the population size and the number of generations in a GA run, respectively.

In order to reduce the amount of computing load, the error reduction measure based on the orthogonalization procedure is adopted as the fitness function. The basic idea is to separate the contribution of the candidate hidden node to the variation of the output from those of the previously found hidden nodes, thereby effectively reducing the number of operations. More specifically, the fitness function $g(\lambda_j)$ of each individual in the population of the GA is given by the following procedure:

1. Using the λ_j given by decoding the chromosomes, calculate the response vector, \mathbf{p}_j .
2. The contribution of already found hidden nodes is removed from the response vector through orthogonalization as follows:

$$\mathbf{p}'_j = \mathbf{p}_j - \sum_{h=1}^{j-1} \frac{\mathbf{p}_j^T \bar{\mathbf{p}}_k}{\|\bar{\mathbf{p}}_k\|^2} \bar{\mathbf{p}}_k \quad (3.29)$$

where

\mathbf{p}'_j is the orthogonalized vector of \mathbf{p}_j
 $\bar{\mathbf{p}}_k$'s ($k = 1, 2, \dots, j-1$) are the orthogonalized response vectors of the previously found ($j-1$) hidden nodes

3. The fitness function $g(\lambda_j)$ is given by the error reduction term or the squared norm of the projection of the vector \mathbf{d} onto \mathbf{p}'_j as follows:

$$g(\lambda_j) = [\text{err}]_j = \frac{(\mathbf{d}^T \mathbf{p}'_j)^2}{\|\mathbf{p}'_j\|^2} \quad (3.30)$$

A schematic flowchart for calculating the fitness value of a candidate hidden node is shown in [Figure 3.1](#). Once the hidden node with the maximum fitness value is found from the GA run, its orthogonalized response vector, \mathbf{p}'_j , is recorded as $\bar{\mathbf{p}}_j$ to be used for finding the next new hidden node. It can be seen that the number of operations in Equation 3.29 is $O(Nj)$ as opposed to $O(Nj^2)$ in Equation 3.28, which leads a significant reduction in computing time.

More details on reproduction, crossover, and mutation follow. The fitness value indicates the goodness of an individual string and a better string has more chance to be reproduced. This is achieved by the biased roulette wheel method in which the probability of an individual being reproduced is linearly proportional to its fitness divided by the summation of all the fitness values in the population. An exact copy of the chosen individual is made and stored in a temporary mating pool, waiting for the mating process by the crossover operator. This process is repeated until the number of reproduced individuals reaches the population size.

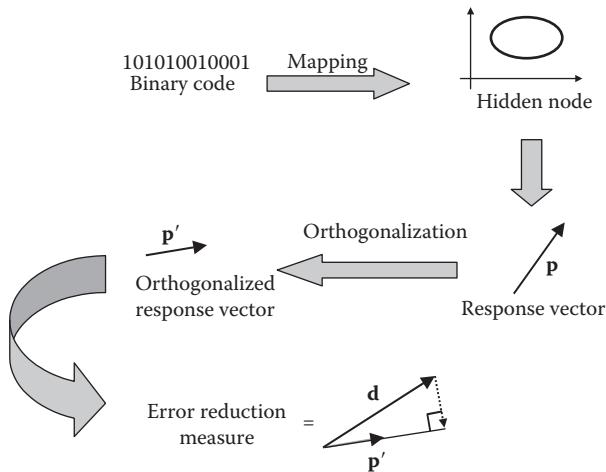


FIGURE 3.1 Procedure for calculating the fitness value of an individual in the GA.

Lee (2000) adopted a two-point crossover operator in order to increase the versatility of possible combinations of genetic information for large strings (Michalewicz, 1996). First, two binary strings are selected at random from the mating pool. It depends on the crossover probability whether the two individuals are to be mated or not. If it is decided that a mating process is to be skipped, the two binary strings become offsprings without any changes. Otherwise, the two individuals go through the following mating process. Consider the two binary strings v_1 and v_2 , for example. Assume that the two crossover points were arbitrarily selected after the fifth bit and the ninth bit as follows:

$$\begin{aligned} v_1 &= + + + + + | + + + + | + + + + + + + \\ v_2 &= - - - - - | - - - - | - - - - - - \end{aligned}$$

Then, the resulting offsprings are given by

$$\begin{aligned} v'_1 &= + + + + + | - - - - | + + + + + + + \\ v'_2 &= - - - - - | + + + + | - - - - - - \end{aligned}$$

The above process is repeated until the number of offsprings reaches the population size. It should be noted here that the crossover points are not limited to the boundaries between parameters such as m_i and σ_i , but can be located anywhere in the binary string. This scheme can increase the chance of obtaining the global optimum by allowing the offsprings to have versatile genetic information.

A mutation process is applied to the offsprings in order to add new genetic information into the binary strings. For each bit in the binary strings, a random number is generated uniformly between 0 and 1. If the random number is smaller than the mutation probability, which is usually preset at a very low value, the corresponding bit is inverted. Otherwise, the bit remains unchanged.

3.6.2 DETERMINATION OF THE NUMBER OF HIDDEN NODES

Since the OLSGA grows the RBFN by one hidden node at each iteration, the optimal number of hidden nodes can be determined in a single run of the algorithm by monitoring the change of modeling errors with increase in the network size. Much research can be found in literature on selection of a model based on various statistical measures from a pool of candidate models. Most popular statistical measures for model selection include the Akaike information criterion (AIC) (Akaike, 1974) based on information theory and the Bayesian criterion (Schwarz, 1978). Any of the statistical measures can be applied to the proposed algorithm to stop growing the RBFN when a certain condition is met. The AIC, for example, for an RBFN with j hidden nodes can be derived as follows:

$$\text{AIC}(j) = N \cdot \log \left(\frac{\|\mathbf{d}\|^2 - \sum_{k=1}^j [\text{err}]_j}{N} \right) + 2j \cdot (2n + 1) \quad (3.31)$$

The minimum AIC represents a model with the maximum entropy level or maximum likelihood. By monitoring the AIC at the end of each iteration of the proposed algorithm, the growth of the network can be stopped when it is determined that the minimum AIC had been reached.

Heuristic methods can also be applied to automatically determine the size of the network during training based on monitoring of change of training errors or testing errors. The training can be stopped if the error reduction rate is less than a predefined value, where the error reduction rate is defined as the percentage change of training or testing errors over increase of one hidden node.

Whichever method among the above is adopted, it should be noted that the network size and the parameters are simultaneously determined by this OLSGA. Any modeling task can be handled in one pass without having to guess, prior training, how many basis function nodes would be needed. Therefore, the OLSGA is regarded as a hybrid structure-parameter learning scheme for RBFNs.

3.6.3 PERFORMANCE EVALUATION

The OLSGA is applied to three numerical examples. Comparisons are made among the OLS algorithm, Chng et al.'s (1996) AOLS algorithm, and the OLSGA. In addition, the performance of OLSGA is compared with those of existing GA-based algorithms in the third example. The AOLS algorithm is implemented following Chng et al.'s (1996) work, and its performance is further improved by replacing their iterative gradient-descent method with the advanced Davidon–Fletcher–Powell method (Rao, 1996). The initial widths σ_0 's in the OLS and the AOLS algorithm are set around the value of $\sqrt{2} \cdot \Delta x/M$, where Δx is the interval of the input domain. These values are commonly used in the application of the OLS algorithm since uniformly distributed RBF nodes in the input space with these widths can cover the entire input domain smoothly.

Following the tactics adopted in the previous studies using the GA (Carse and Fogarty, 1996; Whitehead, 1996), the OLSGA is run four times with independent

TABLE 3.1
Parameters Used for the GA

Population Size	100
Crossover probability	0.95
Mutation probability	0.01
Number of generations	200
Number of bits	$m_i^j = 16$ bits $\sigma_i^j = 16$ bits

seeds for each case, and the average of the four runs is shown as the result of the OLSGA in all three numerical examples in order to minimize the effect of initial populations that are randomly generated. The average of multiple runs, however, has been found to give slightly better performance than a single run in most cases. The parameters of the GA are listed in Table 3.1.

First, two nonlinear functions are chosen to evaluate the performance of the algorithms. Example 3.1 is adopted from Chng et al.'s (1996) paper in order to verify that their algorithms are implemented correctly. In Example 3.2, despite its simple function, the simulation results show that the performance of the AOLS algorithm is not able to match that of the OLSGA algorithm regardless of the initial widths chosen. Example 3.3 compares the generalizing performance between various learning algorithms for the well-known benchmark problem of chaotic time series prediction.

Example 3.1 Approximation of a Nonlinear Function (Chng et al., 1996)

$$F(x_1, x_2) = \sin(x_1 x_2) e^{-|x_1 x_2|}, \quad -3 \leq x_1 \leq 3, \quad -3 \leq x_2 \leq 3 \quad (3.32)$$

With this function, 3,600 uniformly distributed samples are used for learning, while 360,000 data points are used for testing.

Figure 3.2 shows the simulation results for Example 3.1. The horizontal axis indicates the number of RBF nodes used, while the vertical axis represents the nondimensional error index (NDEI) of the testing data, which is calculated by the following equation (Jang, 1993):

$$NDEI = \sqrt{\frac{\sum_{k=1}^N [d(k) - y(k)]^2}{\sum_{k=1}^N [d(k) - \bar{d}]^2}} \quad (3.33)$$

where $\bar{d} \in \Re$ is the mean value of entries in \mathbf{d} . It should be noted that the normalized prediction error defined in Whitehead (1996) is the same as the NDEI defined in this study. Three different initial widths of $\sigma_0 = 0.2, 1$, and 2.5 are tried for the OLS and the AOLS algorithms. The AOLS algorithm shows excellent performance and some of the results are even better than Chng et al.'s

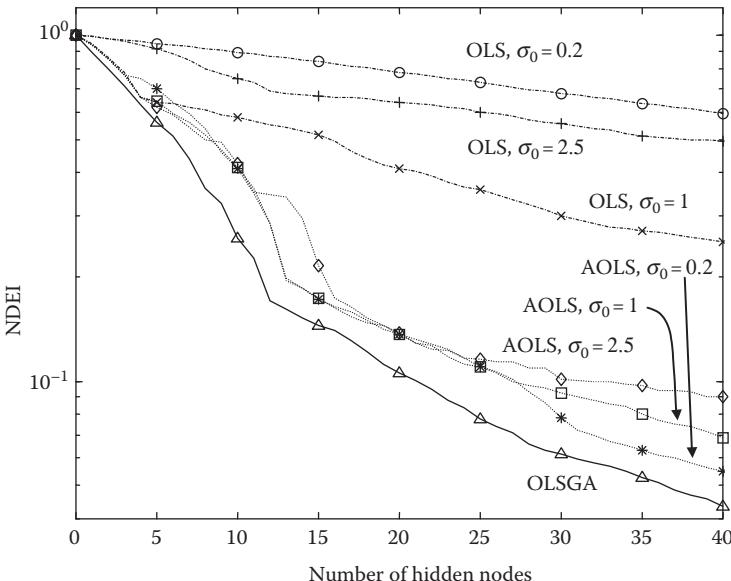


FIGURE 3.2 Comparison of generalizing performance of Example 3.1.

(1996) results in their paper due to the advanced gradient-based algorithm used for the local adaptation in this study. The OLSGA performs as well as the AOLS algorithm for this example. The OLS algorithm, on the other hand, shows its great dependency on the initial width of basis function nodes. The performance of AOLS algorithm is also dependent on the initial choice of function width, although its dependency is smaller than that of the OLS algorithm.

Example 3.2 Approximation of a Nonlinear Function

$$F(x_1, x_2) = e^{-\frac{1}{4}x_1^2 - \frac{25}{9}x_2^2} - 1.2e^{-\frac{25}{4}(x_1 + \frac{2}{5})^2 - x_2^2}, \quad -3 \leq x_1 \leq 3, \quad -3 \leq x_2 \leq 3 \quad (3.34)$$

With this function, 2500 uniformly distributed samples are used for learning, while 250,000 data points are used for testing.

In Figures 3.3 and 3.4, the simulation results of Example 3.2 are shown. Figure 3.3 shows that the performance of the AOLS algorithm, while better than that of the OLS algorithm, greatly depends on the initial width for this problem. The effect of initial width on the performance of the OLS and the AOLS algorithms is further illustrated in Figure 3.4. The initial width is varied from 0.3 to 2.5 with a spacing of 0.1 and the testing errors are obtained by running the algorithms for each initial width until RBFNs with five hidden nodes were generated. It can be seen that the performance of the AOLS algorithm, while much better than that of the OLS algorithm, varies significantly depending on the initial width. The vertical axis shows the testing errors indicating the generalizing performance of each algorithm for 250,000 testing data, which were not used during training. It should be noted that testing errors of the AOLS algorithm are always much larger than those of the OLSGA regardless of the initial width as shown in Figure 3.4.

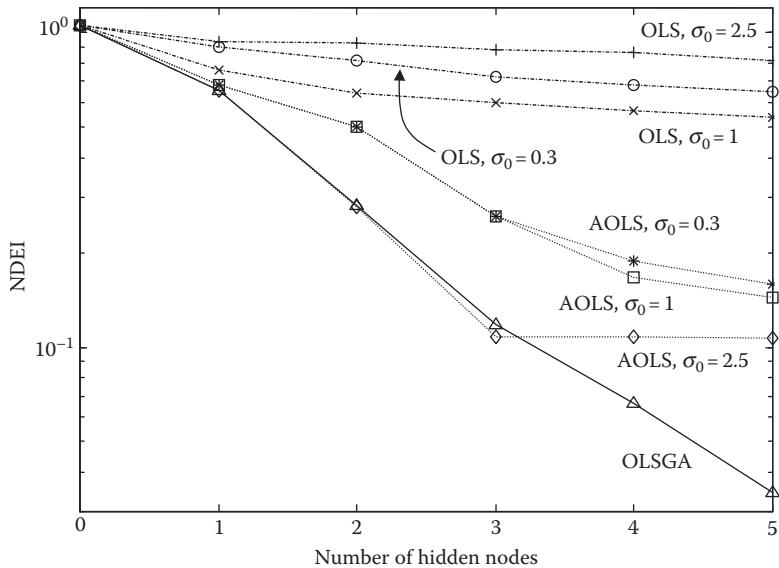


FIGURE 3.3 Comparison of generalizing performance of Example 3.2. The σ_0 denotes the initial width of hidden nodes in the OLS and AOLS algorithms.

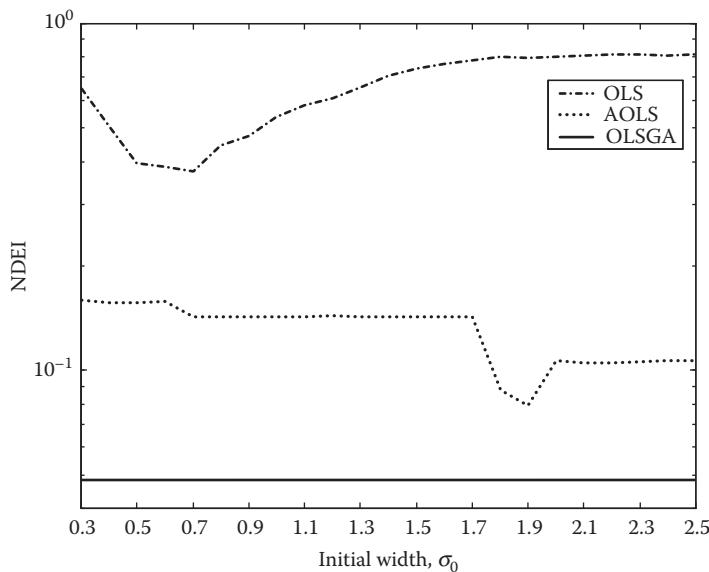


FIGURE 3.4 Effect of the initial width of hidden nodes on the generalizing performance of the OLS and AOLS algorithms compared to that of the OLSGA algorithm for Example 3.2. The NDEI indicates the testing error after the RBFN is trained to have five basis function nodes.

Example 3.3 Prediction of Chaotic Time Series

The performance of various algorithms for training RBFNs is compared using the well-known benchmark problem of predicting Mackey-glass chaotic time series, which is given by the following differential equation (Jang, 1993):

$$\frac{dy(t)}{dt} = -0.1y(t) + 0.2 \frac{y(t-17)}{1+y(t-17)} \quad (3.35)$$

- The goal for training the RBFNs is to predict the value of the times series at point $y(t+85)$ from the earlier points $y(t)$, $y(t-6)$, $y(t-12)$, and $y(t-18)$. Hence, the trained networks have four input nodes and one output node.
- Following the Whitehead's (1996) example, 500 randomly selected points in the time series between $t = 500$ and 4000 are used for learning, while 500 points in sequence for $t > 4000$ are used for testing the prediction capabilities of networks. In this case, the same set of training and testing data used for Whitehead's (1996) example was adopted from his anonymous ftp site for a fair comparison. The RBFNs were trained until algorithms found 50 hidden nodes. For each number of RBF nodes, the prediction error was calculated using the NDEI defined in Equation 3.33.

The generalizing performance of each tested algorithm is shown in Figure 3.5. For the OLSGA, the network was trained four times using independent seeds and the average of prediction errors of four RBFNs is shown as the error curve in

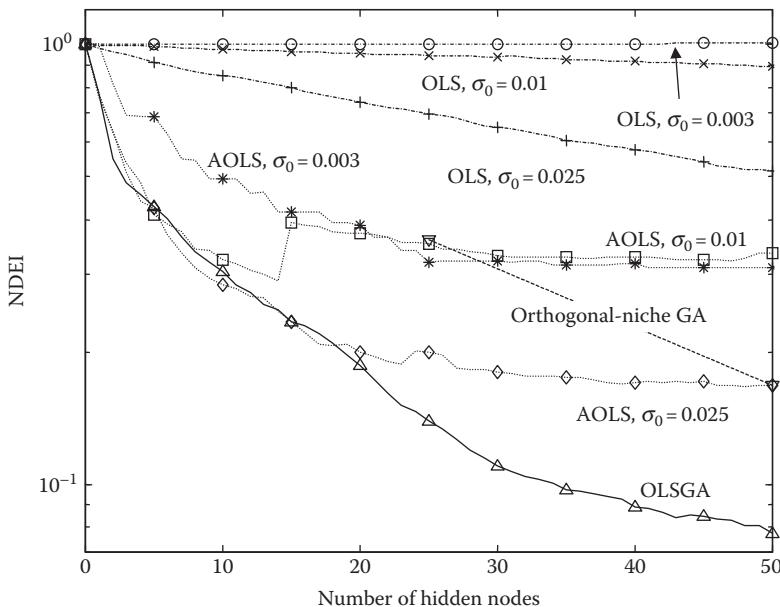


FIGURE 3.5 Comparison of prediction performance in Example 3.3.

TABLE 3.2
Comparison of Prediction Errors among GA-Based Algorithms
for Example 3.3

Number of Hidden Nodes	Fast Evolution (Carse and Fogarty, 1996)	Orthogonal-Niche GA (Whitehead, 1996)	OLSGA
20	0.49	—	0.19
25	—	0.36	0.14
40	0.25	—	0.089
50	—	0.17	0.077
60	0.18	—	—

Figure 3.5. As a reference, the prediction errors from the orthogonal-niche GA calculated by Whitehead (1996) are also shown for 25 and 50 RBF nodes. Figure 3.5 shows the OLSGA has superior performance over other algorithms again. The AOLS algorithm achieved comparable performance only when the initial width was set to 0.025. However, it should be noted that finding the optimal width prior to training is not an easy task without having any knowledge about the training data. When the initial widths are set to 0.01 and 0.003, the AOLS algorithm showed poor prediction performance.

A comparison among GA-based algorithms has also been made as summarized in Table 3.2. In this table, the prediction errors reported by Carse and Fogarty (1996) and Whitehead (1996) are listed. It can be seen that, for the same number of hidden nodes, the OLSGA gives the best prediction performance among the GA-based algorithms. In order to achieve the performance comparable to that of RBFN trained by OLSGA with 20 hidden nodes, fast evolution (Carse and Fogarty, 1996) and orthogonal-niche GA (Whitehead, 1996) required 60 and 50 hidden nodes, respectively.

3.7 ADAPTIVE LEAST-SQUARES LEARNING USING GA

In this section, an effective training method for the fuzzy basis function networks (FBFNs) introduced in Section 2.3.1 is described. As discussed earlier, the structure of the FBFN is very similar to that of the RBFN. Therefore, the learning algorithm of FBFNs described in this section shares common attributes with the OLSGA pro-described in the previous section. Simulation results are presented by applying this training algorithm to various examples.

3.7.1 ADAPTIVE LEAST-SQUARES LEARNING USING GA

Although the FBFN and RBFN are very similar in structure, the normalization term of the FBFN in the denominator of Equation 2.100 prohibits adoption of a model selection method such as the OLS algorithm. The OLS algorithm, when applied to the FBFN (Wang and Mendel, 1992b), fails to retain a meaningful fuzzy system as

shown by Hohensohn and Mendel (1994) since the denominator remains intact during the model selection process. The network produced by the OLS algorithm retains a normalization factor of the original FBFN before the model selection, and hence cannot be converted back to a fuzzy inference system. Therefore, development of a new hybrid structure-parameter learning method for FBFNs that can produce meaningful fuzzy systems is required (Lee and Shin, 2003).

The adaptive least-squares (ALS) algorithm described here produces a meaningful FBFN since the normalization term is regenerated whenever a new fuzzy rule is added to the network. In addition, it achieves a high accuracy in approximating nonlinear relations by searching for the best input membership functions (MFs) in the universe of discourse using the GA, thereby adaptively tuning the nonlinear parameters of basis functions. This will make it possible to construct a parsimonious network at the expense of heavier computing load.

Assume that N input–output training pairs are given: $(\mathbf{x}(h), d(h))$, $h = 1, 2, \dots, N$. The task is to design an FBFN, $f(\mathbf{x})$, such that

$$d(h) = f[\mathbf{x}(h)] + e(h) = \sum_{j=1}^M p_j[\mathbf{x}(h)]w_j + e(h) \quad (3.36)$$

The pseudo-FBF that is the product of all MFs for linguistic terms in the IF part of R^j is defined as follows:

$$q_j(\mathbf{x}) = \prod_{i=1}^n \mu_{A_i^j}(x_i) \quad (3.37)$$

It should be noted here that the pseudo-FBF is exactly the same as the RBF defined in Section 2.2.3.1. Suppose there are M fuzzy rules, and then the FBF can be expressed in terms of pseudo-FBFs as follows:

$$p_j = \frac{q_j(\mathbf{x})}{\sum_{k=1}^M q_k(\mathbf{x})} \quad (3.38)$$

For N input–output training pairs, the response vector of the j th pseudo-FBF is defined as follows:

$$\mathbf{q}_j = [q_j[\mathbf{x}(1)], \dots, q_j[\mathbf{x}(N)]]^T \quad (3.39)$$

Selected M pseudoresponse vectors are arranged to form the following pseudoresponse matrix:

$$\mathbf{Q} = [\mathbf{q}_1, \dots, \mathbf{q}_M] \quad (3.40)$$

Equation 3.36 can be arranged from $h = 1$ to N in matrix form:

$$\mathbf{d} = \mathbf{P}\mathbf{w} + \mathbf{e} \quad (3.41)$$

where

$$\begin{aligned}\mathbf{d} &= [d(1), \dots, d(N)]^T \\ \mathbf{P} &= [\mathbf{p}_1, \dots, \mathbf{p}_M] \text{ with } \mathbf{p}_j = [p_j[\mathbf{x}(1)], \dots, p_j[\mathbf{x}(N)]]^T \\ \mathbf{w} &= [w_1, \dots, w_M]^T \\ \mathbf{e} &= [e(1), \dots, e(N)]^T\end{aligned}$$

The response matrix \mathbf{P} can be easily calculated from the pseudoresponse matrix \mathbf{Q} using the following equation:

$$\begin{aligned}\mathbf{P}(i, j) &= \mathbf{Q}(i, j) / \text{sum}(i) \\ \text{sum}(i) &= \sum_{j=1}^M \mathbf{Q}(i, j)\end{aligned}\tag{3.42}$$

where (i, j) denotes the index of entry in the i th row of the j th column of the matrix. Hence, whenever one wants to add a new fuzzy rule expressed as a pseudo-FBF to the already found fuzzy system, one can simply obtain a new response matrix \mathbf{P} by first adding a new pseudoresponse vector to the pseudoresponse matrix \mathbf{Q} in Equation 3.40 as a new column and then applying Equation 3.42. These features provide a way of sequentially constructing a fuzzy inference system.

Due to introduction of the pseudo-FBF $q_f(\mathbf{x})$, cascade learning of the FBFN becomes a problem of finding in sequence a new pseudo-FBF, which is the product of input MFs of a new fuzzy rule. From Equation 2.99, it can be observed that a pseudo-FBF can be defined by a nonlinear parameter set $\boldsymbol{\lambda} = \{\mathbf{m}, \boldsymbol{\sigma}\}$, where $\mathbf{m} = (m_1, \dots, m_n)^T \in \Re^n$ and $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_n)^T \in \Re^n$ are the center and width vectors of input MFs, respectively. The ALS algorithm tries to locate one pseudo-FBF or parameter set λ at a time in sequence that will maximize the following error reduction measure using the GA:

$$[\text{err}] = \|\mathbf{P}\mathbf{P}^+\mathbf{d}\|^2\tag{3.43}$$

where \mathbf{P}^+ denotes the pseudoinverse of \mathbf{P} . Hence, $\mathbf{P}\mathbf{P}^+$ is the orthogonal projection onto the column space of \mathbf{P} . It should be noted here that the pseudoinverse is not required to be obtained explicitly in order to calculate the error reduction measure, rather $\mathbf{P}\mathbf{P}^+$ can be directly calculated by more efficient methods such as the QR decomposition or singular decomposition method. The advantage of these methods can also be found in the fact that the error reduction measure of a newly added fuzzy rule can be obtained without having to calculate the output MFs explicitly, which are given by $\mathbf{P}^+\mathbf{d}$ as in the least-squares solution.

The procedure for calculating the error reduction measure when a new fuzzy rule is added to the existing fuzzy rule base can be summarized as follows:

1. Using the given nonlinear parameter set $\boldsymbol{\lambda}$ of the new fuzzy rule, construct a new pseudo-FBF and its response vector according to Equations 2.99, 3.37, and 3.39.
2. Add the new pseudoresponse vector to the pseudoresponse matrix \mathbf{Q} as a new column.

3. Apply Equation 3.42 to obtain a new response matrix \mathbf{P} .
4. Calculate the error reduction measure using Equation 3.43.

The ALS algorithm is described below. In the algorithm, l denotes the number of fuzzy rules found. The objective is to find M fuzzy rules when N data points are available.

Step 1. Initialize as $l=0$, the pseudoresponse matrix \mathbf{Q} is an empty matrix.

Step 2. Search for a new pseudo-FBF that will maximize the error reduction measure using the GA. The GA used in the ALS algorithm is similar to that in the OLSGA except the procedure for calculating the error reduction measure, which is described above, and the linear scaling function adopted for the fitness function. The error reduction measure is linearly scaled to produce the fitness value g as follows:

$$g = a \cdot [\text{err}] + b \quad (3.44)$$

where a and b are scalar parameters chosen to prevent premature convergence or random walk (Goldberg, 1989).

Step 3. Insert the response vector \mathbf{q} of a newly found pseudo-FBF into \mathbf{Q} . Calculate the NDEI for training using the error reduction measure obtained from Step 2 as follows:

$$\begin{aligned} \text{NDEI} &= \frac{\sqrt{\sum_{h=1}^N \{f[\mathbf{x}(h)] - d(h)\}^2}}{\sqrt{\sum_{h=1}^N [d(h) - \bar{d}]^2}} \\ &= \frac{\sqrt{\|\mathbf{d}\|^2 - [\text{err}]}}{\|\mathbf{d} - \bar{\mathbf{d}}\|} \end{aligned} \quad (3.45)$$

where $\bar{d} \in \Re$ is the mean value of entries in \mathbf{d} and $\bar{\mathbf{d}} = [\bar{d}, \dots, \bar{d}]^T \in \Re^N$. Set $l=l+1$.

Step 4. If $l < M$, go to Step 2.

Step 5. Generate the \mathbf{P} matrix from \mathbf{Q} using Equation 3.42. The weight vector \mathbf{w} can be calculated using the following equation:

$$\mathbf{w} = \mathbf{P}^+ \mathbf{d} \quad (3.46)$$

Obtain the final FBFN as

$$f(x) = \sum_{j=1}^M p_j(\mathbf{x}) w_j \quad (3.47)$$

Although the only stopping condition for the above algorithm was given by the maximum number of fuzzy rules to be found, the algorithm can be easily modified to run continuously until the training error defined by Equation 3.45 becomes less than

the user's error goal. Hence, a certain error goal can be achieved in one pass without having to guess how many fuzzy rules would be needed. It must be noted that the above algorithm does not require any form of information on the final fuzzy system before learning, while most other parameter or structure-parameter learning algorithms require that the user specify at least one of the parameters of fuzzy network (Wang and Mendel, 1992a,b; Jang, 1993; Nie and Linkens, 1993; Gorenne, 1994; Shimojima et al., 1995; Koo, 1996). Therefore, the new algorithm can be considered as a hybrid structure-parameter learning scheme for FBFNs.

3.7.2 EXTENSION OF ALS ALGORITHM TO MULTI-INPUT, MULTI-OUTPUT SYSTEMS

The ALS algorithm was developed for multi-input single-output (MISO) fuzzy systems in Section 3.7.1. However, there certainly will be a case when a multi-input multi-output (MIMO) representation of a system is required. Hence, the ALS algorithm extended for MIMO FBFNs (Lee and Shin, 2003) is described in this section.

Consider a MIMO fuzzy inference system: $X \subset \Re^n \rightarrow Y \subset \Re^m$. Suppose M fuzzy logic rules are given in the following form:

$$\begin{aligned} R^j: & \text{ IF } x_1 \text{ is } A_1^j \text{ AND } x_2 \text{ is } A_2^j \text{ AND } \dots \text{ AND } x_n \text{ is } A_n^j, \\ & \text{ THEN } y_1 \text{ is } B_1^j \text{ AND } y_2 \text{ is } B_2^j \text{ AND } \dots \text{ AND } y_m \text{ is } B_m^j \end{aligned} \quad (3.48)$$

where A_i^j and B_k^j are linguistic terms characterized by fuzzy MFs $\mu_{A_i^j}(x_i)$ and $\mu_{B_k^j}(y_k)$, respectively, and $j = 1, 2, \dots, M$. Also, assume that a singleton fuzzifier, product inference, a centroid defuzzifier, and Gaussian MFs are used. Then, for a given crisp input vector $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in X$, the following defuzzified, inferred output can be obtained:

$$y_k = f_k(\mathbf{x}) = \frac{\sum_{j=1}^M \bar{y}_k^j \left[\prod_{i=1}^n \mu_{A_i^j}(x_i) \right]}{\sum_{j=1}^M \left[\prod_{i=1}^n \mu_{A_i^j}(x_i) \right]} \quad (3.49)$$

where

$f_k: X \subset \Re^n \rightarrow Y \subset \Re$
 \bar{y}_k^j is the point in the output space Y_k at which $\mu_{B_k^j}(\bar{y}_k^j)$ achieves its maximum value, $k = 1, 2, \dots, m$

$\mu_{A_i^j}(x_i)$ is the Gaussian MF defined by

$$\mu_{A_i^j}(x_i) = \exp \left[-\frac{1}{2} \left(\frac{x_i - m_i^j}{\sigma_i^j} \right)^2 \right] \quad (3.50)$$

where m_i^j and σ_i^j are real-valued parameters.

If FBFs are defined as

$$p^j(\mathbf{x}) = \frac{\prod_{i=1}^n \mu_{A_i^j}(x_i)}{\sum_{j=1}^M \left[\prod_{i=1}^n \mu_{A_i^j}(x_i) \right]}, \quad j = 1, 2, \dots, M \quad (3.51)$$

the fuzzy inference system is equivalent to an FBF expansion or an FBFN:

$$f_k(\mathbf{x}) = \sum_{j=1}^M p^j(\mathbf{x}) w_k^j \quad (3.52)$$

where $w_k^j = \bar{y}_k^j \in \Re$ are constants.

Assume that N input–output training pairs are given: $(\mathbf{x}(h), d(h))$, $h = 1, 2, \dots, N$, where $\mathbf{d}(h) = [d_1(h), \dots, d_m(h)]^\top \in \Re^m$ is the desired output vector of the h th training pair. The task is to design FBFN $f_k(\mathbf{x})$'s such that

$$d_k(h) = f_k[\mathbf{x}(h)] + e_k(h) = \sum_{j=1}^M p^j[\mathbf{x}(h)] w_k^j + e_k(h), \quad k = 1, \dots, m, \quad h = 1, \dots, N \quad (3.53)$$

where $e_k(h) \in \Re$ is the error. Define

$$\mathbf{d}_k = [d_k(1), \dots, d_k(N)]^\top, \quad k = 1, \dots, m \quad (3.54)$$

$$\mathbf{e}_k = [e_k(1), \dots, e_k(N)]^\top, \quad k = 1, \dots, m \quad (3.55)$$

$$\mathbf{w}_k = [w_k^1, \dots, w_k^M]^\top, \quad k = 1, \dots, m \quad (3.56)$$

Then, Equation 3.53 can be arranged from $h = 1$ to N in matrix form:

$$\mathbf{D} = \mathbf{P}\mathbf{W} + \mathbf{E} \quad (3.57)$$

where

$$\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_m]$$

$$\mathbf{P} = [\mathbf{p}^1, \dots, \mathbf{p}^M] \text{ with } \mathbf{p}^j = [p^j(1), \dots, p^j(N)]^\top$$

$$\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_m]$$

$$\mathbf{E} = [\mathbf{e}_1, \dots, \mathbf{e}_m]$$

The error reduction term to be maximized at each step for a MISO system is given by the squared norm of the orthogonal projection of the desired output vector onto the column space of matrix \mathbf{P} . Hence, for a MIMO system, it is natural to maximize the sum of the error reduction term for each desired output vector as follows, which is similar to the one used by Chen et al. (1992) for MIMO RBFNs:

$$[\text{err}] = \sum_{k=1}^m \|\mathbf{P}\mathbf{P}^+ \mathbf{d}_k\|^2 \quad (3.58)$$

where \mathbf{P}^+ is the pseudoinverse of \mathbf{P} . The FBFN is constructed in sequence by adding an FBF that will maximize the above error reduction term using the GA. After the FBFs are found, the weight matrix \mathbf{W} can be obtained by the following equation:

$$\mathbf{W} = \mathbf{P}^+ \mathbf{D} \quad (3.59)$$

3.7.3 PERFORMANCE EVALUATION IN APPROXIMATING NONLINEAR FUNCTIONS

In this section, the ALS algorithm is compared with the BP algorithm, which is modified from the gradient-based algorithm proposed by Wang and Mendel (1992a). Wang and Mendel derived an incremental training algorithm based on the gradient of the objective function, which is the squared error of each training pair. In this study, the sum of squares of entire training error is used instead as the objective function in order to achieve batch-mode training, and corresponding updated equations are given below. Please refer to Wang and Mendel (1992a) for detailed derivation.

$$w_j(\eta + 1) = w_j(\eta) - \alpha \sum_{h=1}^N p_j(h)[f(h) - d(h)] \quad (3.60)$$

$$m_i^j(\eta + 1) = m_i^j(\eta) - \alpha \sum_{k=1}^N p_j(k)[f(k) - d(k)][w_j(\eta) - f(k)] \frac{x_i(k) - m_i^j(\eta)}{\sigma_i^{j2}(\eta)} \quad (3.61)$$

$$\sigma_i^j(\eta + 1) = \sigma_i^j(\eta) - \alpha \sum_{k=1}^N p_j(k)[f(k) - d(k)][w_j(\eta) - f(k)] \frac{[x_i(k) - m_i^j(\eta)]^2}{\sigma_i^{j3}(\eta)} \quad (3.62)$$

where η and α denote the number of epochs and learning constant, respectively. In order to ensure stable convergence and expedite the training, the adaptive learning constant scheme is adopted, which has been widely used to improve the classic gradient-based algorithm (Jang, 1993; Lin and Lee, 1996). If the error decreased consecutively, the learning rate was increased by a factor of 0.05, while the learning rate was decreased by a factor of 0.3 and the update was cancelled if the error increased. Training was terminated when the change of training error was less than 0.0001% for 10 consecutive epochs.

In order to evaluate the performance of the ALS algorithm against the BP algorithm, two nonlinear functions are chosen. The details of the functions and training conditions are listed below.

Example 3.4

$$F(x_1, x_2) = 3(1 - x_1)^2 e^{-x_1^2 - (x_2 + 1)^2} - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right)e^{-x_1^2 - x_2^2} - \frac{1}{3}e^{-(x_1 + 1)^2 - x_2^2}, \quad -3 \leq x_1 \leq 3, \quad -3 \leq x_2 \leq 3 \quad (3.63)$$

Example 3.5

$$F(x_1, x_2) = \sin(x_1 x_2) e^{-|x_1 x_2|}, \quad -3 \leq x_1 \leq 3, -3 \leq x_2 \leq 3 \quad (3.64)$$

From both of the above functions, 900 uniformly distributed samples were generated and used for training, while another 841 uniformly distributed data were used for testing the generalizing capabilities of FBFN. The following parameters of GA were used for both the above examples.

Population size = 200

Generation gap = 1.0

Crossover probability = 1.0

Mutation probability = 0.005

Number of generations = 100

Number of bits used for building chromosomes = 12 bits for each m_i^j and σ_i^j

For the BP algorithm, the number of fuzzy rules had to be fixed before training because BP is strictly a parameter learning algorithm. The initial centers of MFs were chosen to be uniformly distributed in the universe of discourse, while initial widths of MFs were calculated to cover the universe of discourse with ϵ -completeness of $\exp(-0.5)$ (Lin and Lee, 1996). The w_j 's were initialized by applying Equation 3.46.

The simulation results of Example 3.4 are shown in Figures 3.6 through 3.9. Figure 3.6 plots the training and testing errors versus the number of fuzzy rules as the fuzzy network systems are sequentially constructed by the ALS algorithm. The errors on the vertical axis represent the NDEIs, which are calculated by Equation 3.45.

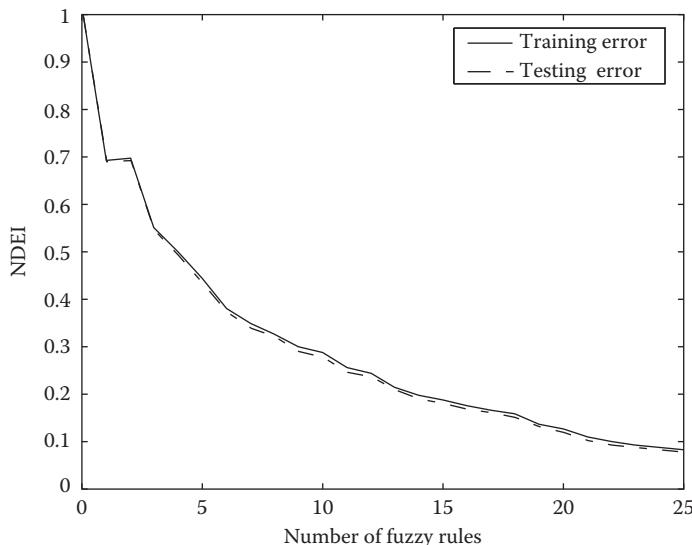


FIGURE 3.6 Error curves for Example 3.4 with the ALS algorithm.

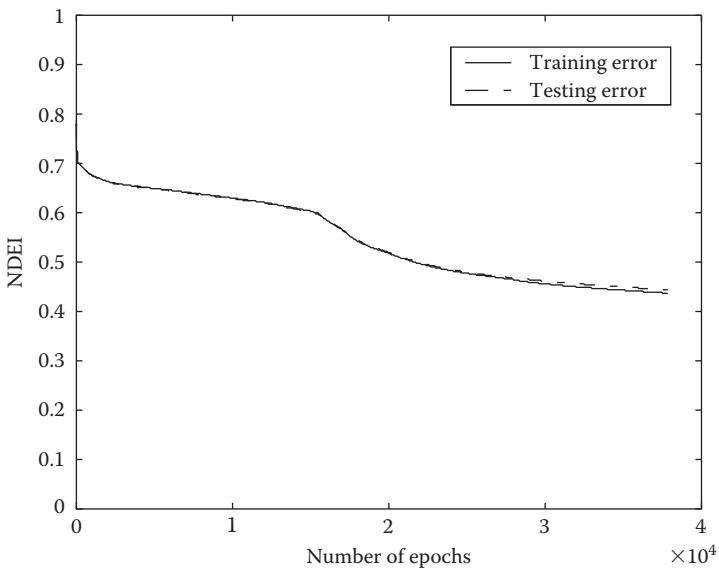


FIGURE 3.7 Example of error curves for Example 3.4 when the BP algorithm is applied to FBFN with 25 fuzzy rules.

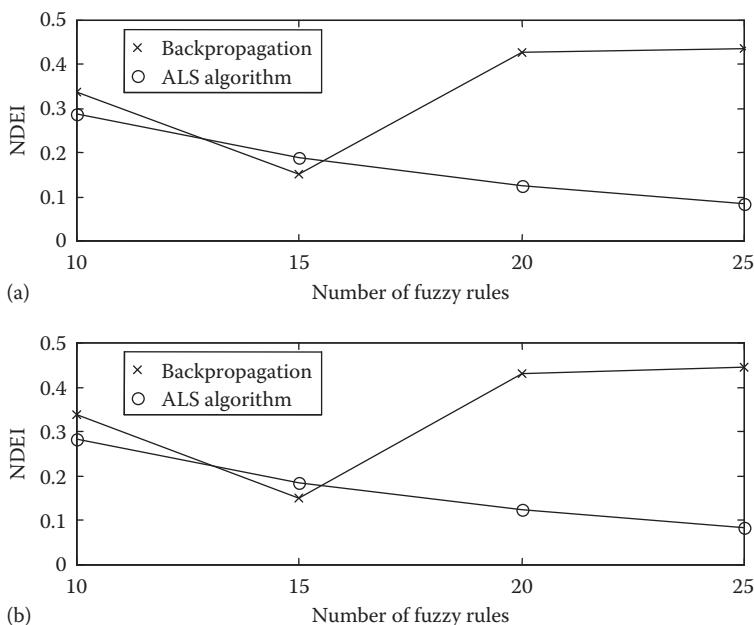


FIGURE 3.8 Comparison between training and testing errors for the ALS algorithm and the BP algorithm applied to Example 3.4: (a) training errors and (b) testing errors.

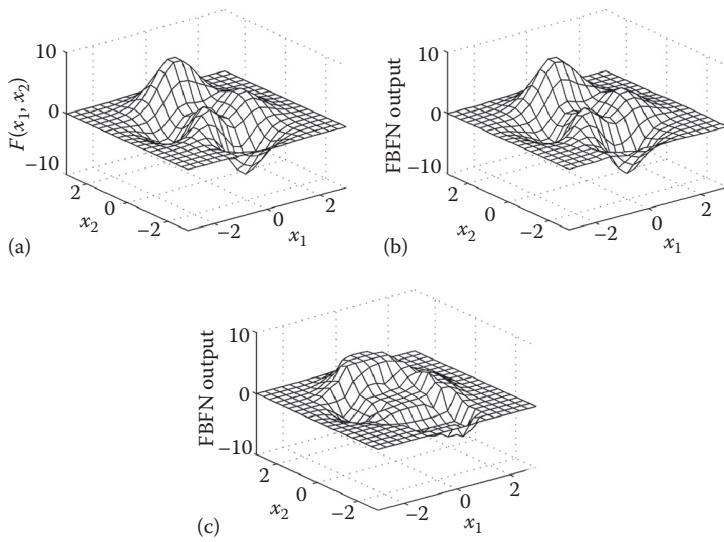


FIGURE 3.9 Comparison between the function for Example 3.4 and simulated outputs from two FBFNs with 25 fuzzy rules: (a) original function, (b) FBFN with the ALS algorithm, and (c) FBFN with the BP algorithm.

Except when the number of fuzzy rules increases from 1 to 2, both the training and testing errors decrease as a new fuzzy rule is added to the previous fuzzy system. Figure 3.7 shows an example of error curves versus the number of epochs for $M = 25$ with the BP algorithm. It can be seen that after the initial decrease of error, it stays at a high level of error even after 30,000 epochs, because learning is trapped in a local minimum. The comparison of training and testing errors between two algorithms for $M = 10, 15, 20$, and 25 are shown in Figure 3.8. The ALS algorithm provides superior results over the BP algorithm except the case when $M = 15$.

It should be noted that the BP algorithm had to be applied individually for each predefined number of fuzzy rules to obtain the plot, while the ALS algorithm provided all the results in one pass. It means that the number of fuzzy rules required to achieve a certain error goal can be determined only by many trials with the BP algorithm, while it can be determined after just one trial with the ALS algorithm. It can be observed from the graphs of testing error that the ALS algorithm also shows outstanding performance in generalization. Figure 3.9 shows the output of FBFNs obtained from two algorithms compared with the original function when $M = 25$. As it can be expected from the error curves in Figure 3.8, the BP algorithm fails to approximate the original function, while the ALS algorithm approximates the function very closely. Figures 3.10 and 3.11 show the simulation results of Example 3.5. The ALS algorithm again shows significantly better results than the BP algorithm.

The simulation results of the two examples show that the ALS algorithm approximates nonlinear functions much better than the BP algorithm. Although

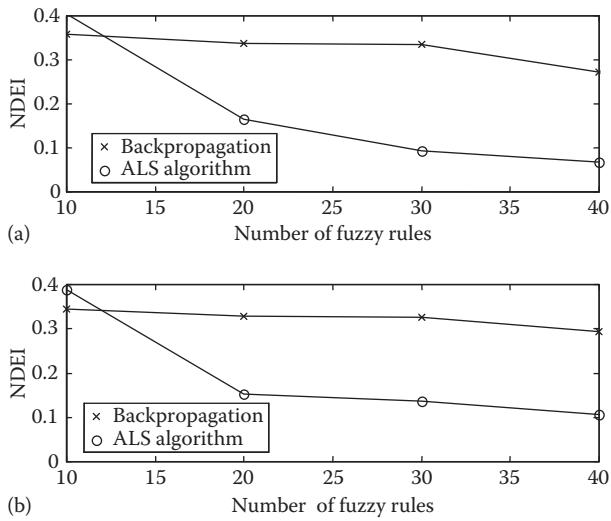


FIGURE 3.10 Comparison between training and testing errors for the ALS algorithm and the BP algorithm applied to Example 3.5: (a) training errors and (b) testing errors.

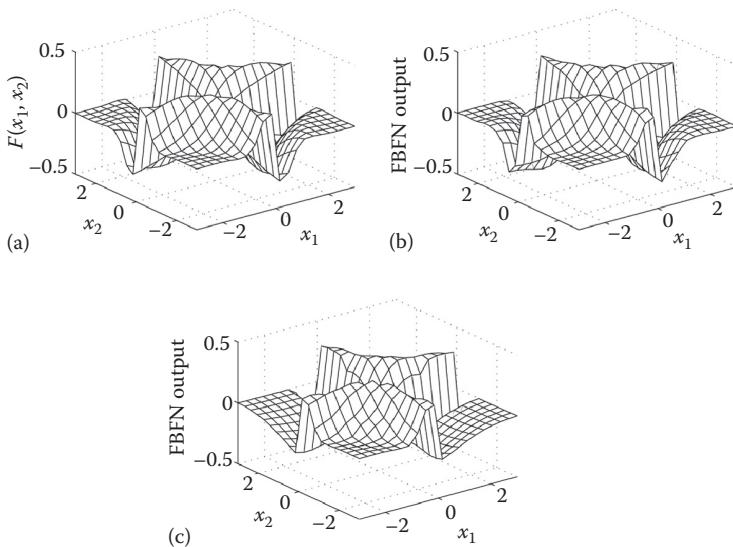


FIGURE 3.11 Comparison between the function for Example 3.5 and simulated outputs from two FBFNs with 40 fuzzy rules: (a) original function, (b) FBFN with the ALS algorithm, and (c) FBFN with the BP algorithm.

a more sophisticated algorithm based on numerical optimization theories with a judicious choice of initial parameters might achieve better performance than the BP algorithm used in this study, the advantage of the proposed algorithm should be stressed by the fact that the ALS algorithm is a hybrid structure-parameter learning scheme, which does not require any parameters of a fuzzy system to be predefined.

3.7.4 APPLICATION OF FBFN TO MODELING OF GRINDING PROCESSES

In this section, two grinding models from various literatures are selected in order to test the capabilities of FBFNs trained by the ALS algorithm for approximating nonlinear grinding processes. The two models are used to compare the performance of the ALS algorithm itself with that of the BP algorithm for individual FBFNs.

Example 3.6 Chip Thickness Model (Tönshoff et al., 1992)

$$h(v_s, v_w, a_e) = C_{gw} \left(\frac{1}{q} \right)^{e_1} \left(\frac{a_e}{d_{eq}} \right)^{\frac{e_1}{2}} \quad (3.65)$$

where

h is the maximum chip thickness (μ)

C_{gw} is a constant of 0.0273

q is the speed ratio = v_s/v_w

v_s is the wheel speed

v_w is the work speed

d_{eq} is the equivalent diameter, which is 350 mm in this simulation

e_1 is another constant of 1/3

a_e is the working engagement (mm)

Hence, this model is a three-input one-output system. The ranges of three-input variables (v_s, v_w, a_e) were chosen to be [15 m/s, 40 m/s] \times [0.02 m/s, 0.4 m/s] \times [4.28 μ m, 34.3 μ m].

Within the ranges, 216 uniformly distributed samples were used for learning, while another 125 uniformly distributed data were used for testing the generalizing capabilities of the FBFN. The parameters of GA used in this study are listed in [Table 3.3](#). For the BP algorithm, the initial parameters of the FBFN were chosen by applying the method given in [Section 3.7.3](#).

The simulation results of Example 3.6 are shown in [Figures 3.12](#) and [3.13](#). The comparisons of training and testing errors between two algorithms for $M=2, 3$, and 4 are shown in Figure 3.12. The errors on the vertical axis represent the NDEIs, which are calculated by applying Equation 3.33. The ALS algorithm provides superior results over the BP algorithm for all the number of fuzzy rules. It can be observed from the graphs of testing error that the ALS algorithm also shows good performance in generalization. Figure 3.13 shows the output of FBFNs obtained from two algorithms compared with the original function when $M=4$.

TABLE 3.3
Parameters Used for the GA

	Example 3.6	Example 3.7
Population size	150	150
Crossover probability	0.95	0.95
Mutation probability	0.008	0.008
Number of generations	100	100
Number of bits	12 bits each	12 bits each

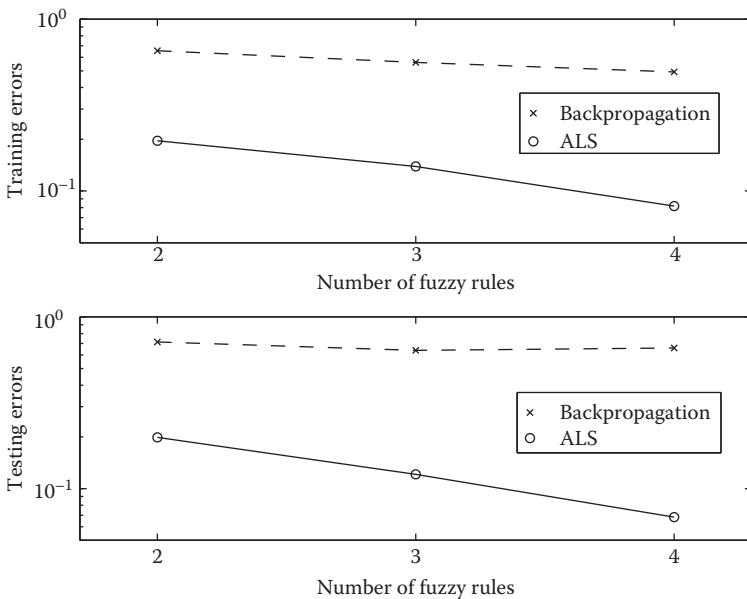


FIGURE 3.12 Comparison of training and testing errors between the ALS algorithm and the BP algorithm applied to Example 3.6.

As it can be expected from the error curves in Figure 3.12, the BP algorithm fails to approximate the original function, while the ALS algorithm approximates the function very closely with only four fuzzy rules.

Example 3.7 Maximum Grinding Temperature Model (Tönshoff et al., 1992)

$$T_{\max}(v_s, v_w, a_e) = A_1 \frac{\alpha^{e_1}}{\lambda} a_e^{e_2} v_w^{e_3} v_s^{e_4} d_{eq}^{e_5} \exp(A_2 a_e^{e_6} v_w^{e_7} d_{eq}^{e_8} z) \quad (3.66)$$

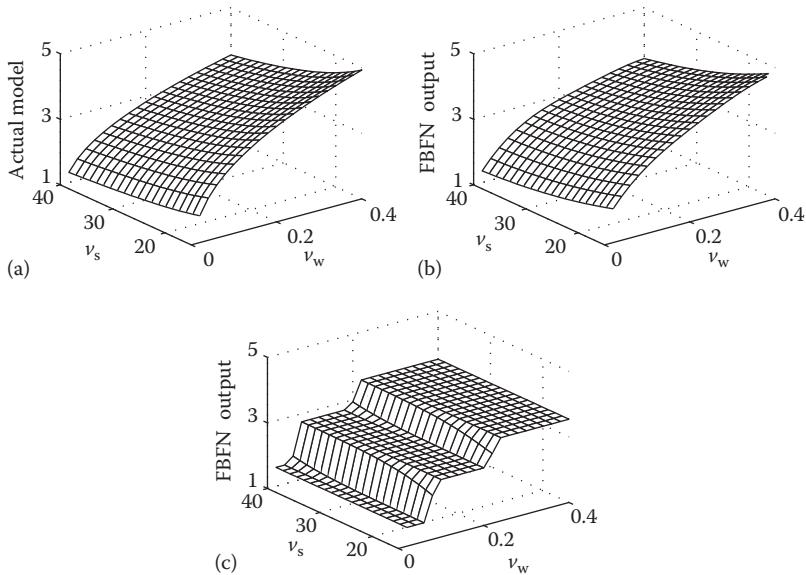


FIGURE 3.13 Comparison between the analytical equation for Example 3.6 and simulated outputs from two FBFNs with four fuzzy rules when $a_e = 17.1 \mu\text{m}$: (a) analytical equation, (b) FBFN with the ALS algorithm, and (c) FBFN with the BP algorithm.

where

T_{\max} is the maximum grinding temperature at depth z ($z=1 \text{ mm}$ for this simulation) ($^{\circ}\text{C}$)

α is the thermal diffusivity (mm^2/s) of the work material

λ is the thermal conductivity of the work material ($\text{W}/\text{m}/\text{K}$)

a_e is the working engagement (mm)

v_w is the work speed (m/s)

v_s is the wheel speed (m/s)

d_{eq} is the equivalent diameter (mm)

$A_1, A_2, e_1, e_2, e_3, e_4, e_5, e_6, e_7$, and e_8 are constants, which depend on grinding conditions and wheel–workpiece combinations. Their values in this simulation are 1094, -0.1475 , 0.47, 0.545, 0.21, 0.32, -0.015 , -0.185 , 0.63, and -0.185 , respectively

The range of three-input variables were chosen to be $(v_s, v_w, a_e) = [15 \text{ m/s}, 40 \text{ m/s}] \times [0.02 \text{ m/s}, 0.4 \text{ m/s}] \times [4.28 \mu\text{m}, 21.4 \mu\text{m}]$. Within the ranges, 216 uniformly distributed samples were used for learning, while another 125 uniformly distributed data were used for testing.

Figures 3.14 and 3.15 show the simulation results of Example 3.7. With this example, only three fuzzy rules were required for the ALS algorithm to achieve a good performance, while they were insufficient for the BP algorithm. The ALS algorithm again shows significantly better results than the BP algorithm both in approximation and generalization performances.

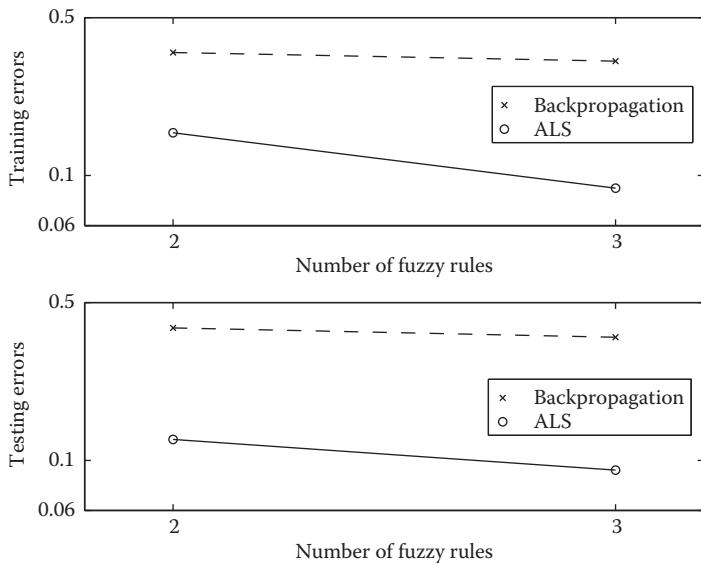


FIGURE 3.14 Comparison of training and testing errors between the ALS algorithm and the BP algorithm applied to Example 3.7.

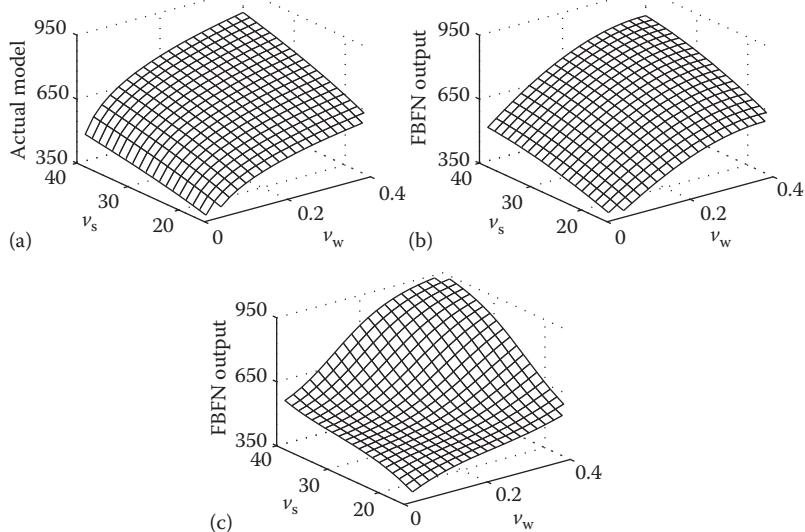


FIGURE 3.15 Comparison between the analytical equation for Example 3.7 and simulated outputs from two FBFNs with three fuzzy rules when $a_e = 17.1 \mu\text{m}$: (a) analytical equation, (b) FBFN with the ALS algorithm, and (c) FBFN with the BP algorithm.

REFERENCES

- Ackley, D.H., Hinton, G.E., and Sejnowski, T.J., A learning algorithm for Boltzmann machines, *Cognitive Science*, 9: 147–169, 1985.
- Akaike, H., A new look at the statistical model identification, *IEEE Transactions on Automatic Control*, 19(6): 716–723, 1974.
- Carpenter, G.A. and Grossberg, S., The ART of adaptive pattern recognition by a self-organizing neural network, *Computer*, 21: 77–88, 1988.
- Carse, B. and Fogarty, T., Fast evolutionary learning of minimal radial basis function neural networks using a genetic algorithm, *Evolutionary Computing*, AISB Workshop, Springer, Brighton, United Kingdom, pp. 1–22, 1996.
- Chen, S., Cowan, C.F.N., and Grant, P.M., Orthogonal least squares learning algorithm for radial basis function networks, *IEEE Transactions on Neural Networks*, 2(2): 302–309, 1991.
- Chen, S., Grant, P.M., and Cowan, C.F.N., Orthogonal least-squares algorithm for training multioutput radial basis function networks, *IEEE Proceedings-F*, 139(6): 378–384, 1992.
- Chng, E.-S., Yang, H.H., and Bos, S., Orthogonal least-squares learning algorithm with local adaptation process for the radial basis function networks, *IEEE Signal Processing Letters*, 3(8): 253–255, 1996.
- Glorennec, P.Y., Learning algorithms for neuro-fuzzy networks, *Fuzzy Control Systems*, CRC Press, Boca Raton, FL, pp. 3–18, 1994.
- Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
- Hecht-Nielsen, R., Counter-propagation networks, *IEEE 1st International Conference on Neural Networks*, San Diego, CA, Vol. 2, pp. 19–32, 1987.
- Hohensohn, J. and Mendel, J.M., Two-pass orthogonal least-squares algorithm to train and reduce fuzzy logic systems, *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, Orlando, FL, pp. 696–700, 1994.
- Hopfield, J.J., Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, 79: 2554–2558, 1982.
- Jang, J.-S.R., ANFIS: Adaptive-network-based fuzzy inference system, *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3): 665–685, 1993.
- Jin, L. and Gupta, M.M., Stable dynamic backpropagation learning in recurrent neural networks, *IEEE Transactions on Neural Networks*, 10(6): 1321–1334, November, 1999.
- Kohonen, T., Barna, G., and Chrisley, R., Statistical pattern recognition with neural networks: Benchmark studies, *Proceedings of the 2nd Annual IEEE International Conference on Neural Networks*, San Diego, CA, Vol. 1, pp. 61–68, 1988.
- Koo, T.-K.J., Construction of fuzz linguistic model, *Proceedings of the 35th Conference on Decision and Control*, Kobe, Japan, December, pp. 98–103, 1996.
- Kosko, B., Bidirectional associative memories, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 18, pp. 49–60, 1988.
- Lee, C.W., Intelligent modeling and optimization of grinding processes, Ph.D. Dissertation, Purdue University, West Lafayette, IN, 2000.
- Lee, C.W. and Shin, Y.C., Construction of fuzzy systems using least-squares method and genetic algorithm, *Fuzzy Sets and Systems*, 137: 297–323, 2003.
- Lin, C.-T. and Lee, C.S.G., *Neural Fuzzy Systems*, Prentice HAT P T R, Reading, NJ, 1996.
- Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd edn., Springer-Verlag, Reading, United Kingdom/New York, 1996.
- Minsky, M.L. and Papert, S.S., *Perceptrons*, MIT Press, Cambridge, MA, 1969.
- Nie, J. and Linkens, D.A., Learning control using fuzzified self-organizing radial basis function networks, *IEEE Transactions on Fuzzy Systems*, 1(4): 280–287, 1993.

- Rao, S.S., *Engineering Optimization: Theory and Practice*, 3rd edn., John Wiley & Sons, Reading, NY, 1996.
- Rumelhart, D., Hinton, G.E., and Williams, R.J., Learning internal representations by error propagation, *Parallel Distributed Processing*, Rumelhart, D.E. and McClelland, J.L. (Eds.), Vol. 1, MIT Press, Cambridge, MA, 1986.
- Schwarz, G., Estimating the dimension of a model, *The Annals of Statistics*, 6(2): 461–464, 1978.
- Shimojima, K., Fukuda, T., and Hasegawa, Y., RBF-fuzzy system with GA based unsupervised/supervised learning method, *Proceedings of 1995 IEEE International Conference on Fuzzy Systems*, Yokohama, Japan, March 20–24, pp. 253–258, 1995.
- Stacey, D., Intelligent systems architecture: Design techniques, *Artificial Neural Networks for Intelligent Manufacturing*, Dagli, C. (Ed.), Chapman & Hall, London, United Kingdom, 1994.
- Strang, G., *Linear Algebra and Its Applications*, 2nd edn., Academic Press, Inc., Reading, NY, 1980.
- Tönshoff, H.K., Peters, J., Inasaki, I., and Paul, T., Modelling and simulation of grinding processes, *Annals of the CIRP*, 41(2): 677–688, 1992.
- Wang, L.X. and Mendel, J.M., Back-propagation fuzzy systems as nonlinear dynamic system identifiers, *Proceedings of the IEEE 1992 International Conference on Fuzzy Systems*, San Diego, CA, pp. 1409–1418, 1992a.
- Wang, L.X. and Mendel, J.M., Fuzzy basis functions, universal approximation, and orthogonal least-squares learning, *IEEE Transactions on Neural Networks*, 3(5): 807–814, 1992b.
- Whitehead, B.A., Genetic evolution of radial basis function coverage using orthogonal niches, *IEEE Transactions on Neural Networks*, 7(6): 1525–1528, 1996.
- Widrow, B. and Lehr, M.A., Thirty years of adaptive neural networks: perceptron, madaline, and backpropagation, *Proceedings of the IEEE*, 78(9): 1514–1542, September 1996.

4 Fuzzy Inverse Model Development

In this chapter, a novel fuzzy inverse model derivation procedure is developed and implemented for a general multi-input single-output (MISO) nonlinear plant (Xu and Shin, 2008). First, the system dynamics is modeled in fuzzy domain with triangular input membership functions (MFs), singleton output MF, and fuzzy-mean defuzzification. And then the fuzzy inverse model is systematically constructed from the obtained fuzzy model, which has the ability of uniquely determining the inverse relationship for each input–output pair. It is derived in a straightforward way and the required input variables can be simultaneously obtained by the fuzzy inferencing calculation to realize the desired output value. Simulation examples are provided to demonstrate the effectiveness of the proposed method to find the inverse kinematics solution for complex multiple degree-of-freedom industrial robot manipulators.

4.1 FUZZY INVERSE MODEL DEVELOPMENT

A typical multivariable inverse problem is computing multiple input values to realize the desired output values of a target system. The inverse model could be used as a decision-making tool for off-line optimization or for online control of a nonlinear plant. However, in most existing inverse methods for MISO systems, only one input variable is inverted in the fuzzy inverse model construction, assuming that the remaining input variables are known beforehand. Their usefulness is largely limited in the sense that only one input variable can be adjusted while the other input variables are fixed as the previous value. In contrast, in this chapter, a fuzzy inverse model is designed for all input variables, so that the desired output can be realized by adjusting the multiple inputs simultaneously. The fuzzy inverse model is automatically constructed from the system forward model and is able to uniquely determine the inverse relationship for each input–output pair. The multiple input variables can be obtained at the same time in order to achieve the desired system output.

Consider a general MISO nonlinear system with p inputs in the state-space form as

$$\dot{\mathbf{x}}(k) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) \quad (4.1a)$$

$$y(k) = g(\mathbf{x}(k), \mathbf{u}(k)) \quad (4.1b)$$

where

$\mathbf{x}(k) = [x_1(k), x_2(k), \dots, x_n(k)]^T \in \mathbb{R}^n$ is the state vector

$\mathbf{x}(0)$ is the state vector at the initial time

$\mathbf{u}(k) = [u_1(k), \dots, u_p(k)]^T \in \mathbb{R}^p$ is the system input vector

$y(k)$ is the scalar output

$\mathbf{f}(\cdot)$ and $g(\cdot)$ are smooth functions representing the system's nonlinear dynamics

This MISO system output Equation 4.1b can be represented by the following fuzzy model as

$$\begin{aligned} R^i: \text{IF } & x_1(k) = B_1^i \text{ AND } x_2(k) = B_2^i \text{ AND } \dots \text{ AND } x_n(k) = B_n^i \text{ AND } \\ & u_1(k) = D_1^i \text{ AND } u_2(k) = D_2^i \text{ AND } \dots \text{ AND } u_p(k) = D_p^i \\ \text{THEN } & y(k) = c^i \end{aligned} \quad (4.2)$$

where

R^i ($i = 1, 2, \dots, l$) represents the i th rule of the fuzzy model, l is the number of fuzzy rules

$B_1^i, B_2^i, \dots, B_n^i, D_1^i, D_2^i, \dots, D_p^i$ are triangular MFs

c^i represents a singleton MF

Define the center of the triangular MF D_r^i at position d_r^i , such that $\ker(D_r^i) = d_r^i$, where $\ker(D_r^i) = \{d_r^i | \mu_{D_r^i}[u_r(k)] = 1\}$, $r = 1, 2, \dots, p$. Assume the l triangular MFs D_r^i for the system input $u_r(k)$ in each rule ($i = 1, 2, \dots, l$) overlap adjacently as $\sum_{i=1}^l \mu_{D_r^i}[u_r(k)] = 1$.

For simplification, a multidimensional fuzzy set X^i is defined for the state vector $\mathbf{x}(k) = [x_1(k), \dots, x_n(k)]^T$ by applying the min operator in the Cartesian product space $X^i = B_1^i \times \dots \times B_n^i$ with the corresponding membership degree as

$$\begin{aligned} \mu_{X^i}[\mathbf{x}(k)] &= \mu_{B_1^i \times \dots \times B_n^i}(x_1(k), \dots, x_n(k)) \\ &= \min(\mu_{B_1^i}[x_1(k)], \dots, \mu_{B_n^i}[x_n(k)]) \\ &= \mu_{B_1^i}[x_1(k)] \wedge \dots \wedge \mu_{B_n^i}[x_n(k)] \end{aligned} \quad (4.3)$$

Therefore, the fuzzy rules for the system dynamics can be simplified as

$$\begin{aligned} R^i: \text{IF } & \mathbf{x}(k) = X^i \text{ AND } \\ & u_1(k) = D_1^i \text{ AND } u_2(k) = D_2^i \text{ AND } \dots \text{ AND } u_p(k) = D_p^i \\ \text{THEN } & y(k) = c^i \end{aligned} \quad (4.4)$$

With min fuzzy intersection operation (Piegat, 2001) and fuzzy-mean defuzzification (Babuška, 1998), when the system input vector $\mathbf{u} = [u_1, \dots, u_p]^T$ is introduced, the output $y(k)$ of this MISO system is calculated as

$$\begin{aligned}
y(k) &= \frac{\sum_{i=1}^l c^i \left\{ \mu_{B_1^i}[x_1(k)] \wedge \cdots \wedge \mu_{B_n^i}[x_n(k)] \wedge \mu_{D_1^i}[u_1(k)] \wedge \cdots \wedge \mu_{D_p^i}[u_p(k)] \right\}}{\sum_{i=1}^l \left\{ \mu_{B_1^i}[x_1(k)] \wedge \cdots \wedge \mu_{B_n^i}[x_n(k)] \wedge \mu_{D_1^i}[u_1(k)] \wedge \cdots \wedge \mu_{D_p^i}[u_p(k)] \right\}} \\
&= \frac{\sum_{i=1}^l c^i \left\{ \mu_{x^i}[\mathbf{x}(k)] \wedge \mu_{D_1^i}[u_1(k)] \wedge \cdots \wedge \mu_{D_p^i}[u_p(k)] \right\}}{\sum_{i=1}^l \left\{ \mu_{x^i}[\mathbf{x}(k)] \wedge \mu_{D_1^i}[u_1(k)] \wedge \cdots \wedge \mu_{D_p^i}[u_p(k)] \right\}} \\
&= \frac{\sum_{i=1}^l c^i \left(\left\{ \mu_{x^i}[\mathbf{x}(k)] \wedge \mu_{D_1^i}[u_1(k)] \right\} \wedge \cdots \wedge \left\{ \mu_{x^i}[\mathbf{x}(k)] \wedge \mu_{D_p^i}[u_p(k)] \right\} \right)}{\sum_{i=1}^l \left(\left\{ \mu_{x^i}[\mathbf{x}(k)] \wedge \mu_{D_1^i}[u_1(k)] \right\} \wedge \cdots \wedge \left\{ \mu_{x^i}[\mathbf{x}(k)] \wedge \mu_{D_p^i}[u_p(k)] \right\} \right)}
\end{aligned} \tag{4.5}$$

Therefore, for the i th rule R^i , a series of trapezoidal MFs $\bar{D}_1^i, \bar{D}_2^i, \dots, \bar{D}_p^i$ are defined for the p input variables and each individual membership degree is denoted as $\mu_{\bar{D}_r^i}[u_r(k)] = \mu_{x^i}[\mathbf{x}(k)] \wedge \mu_{D_r^i}[u_r(k)]$. The height of the fuzzy MF is $\mu_{x^i}[\mathbf{x}(k)]$, which can be calculated from Equation 4.3. The MISO fuzzy system is further expressed as

$$\begin{aligned}
R^i: \text{IF } u_1(k) = \bar{D}_1^i \text{ AND } u_2(k) = \bar{D}_2^i \text{ AND } \dots \text{ AND } u_p(k) = \bar{D}_p^i \\
\text{THEN } y(k) = c^i
\end{aligned} \tag{4.6}$$

THEOREM 4.1

Suppose at any given time instant, the state vector is measurable as $x(k) = [x_1(k), \dots, x_n(k)]^T$ and the desired system output is known as $y_d(k)$. The fuzzy forward model (Equation 4.6) of the dynamic MISO nonlinear system is invertible if and only if in each rule, $d_r^m \geq d_r^n \rightarrow c^m \geq c^n$ (or $d_r^m \leq d_r^n \rightarrow c^m \leq c^n$), $\forall r \in [1, p], \forall m, nm \neq n$. The fuzzy inverse model for each system input $u_r(k)$ is constructed as

$$\begin{aligned}
\text{IR}^i: \text{IF } y(k) = C^i \text{ THEN } u_1(k) = \bar{d}_1^i \\
\text{IF } y(k) = C^i \text{ THEN } u_2(k) = \bar{d}_2^i \\
\quad \dots \quad \dots \\
\text{IF } y(k) = C^i \text{ THEN } u_p(k) = \bar{d}_p^i
\end{aligned} \tag{4.7}$$

where IR^i ($i = 1, 2, \dots, l$) represents the i th rule of the fuzzy inverse model. Denote the output MF vector as $\bar{\mathbf{d}}^i = [\bar{d}_1^i, \dots, \bar{d}_p^i]^T$, and then the fuzzy inverse model can be expressed in a vector form as

$$\text{IR}^i: \text{IF } y(k) = C^i \text{ THEN } \mathbf{u}(k) = \bar{\mathbf{d}}^i \tag{4.8}$$

■

Denotation

C^i : A triangular MF for output $y(k)$, whose center is at position c^i , such that $\ker(C^i) = c^i$ and $\ker(C^i) = \{c^i | \mu_{C^i}[y(k)] = 1\}$. The l triangular MFs C^i in each rule overlap adjacently to each other, such that $\sum_{i=1}^l \mu_{C^i}[y(k)] = 1$. Therefore, at any sampling time, a crisp input can fire maximum two fuzzy sets. Suppose the desired output $y_d(k)$ fires two adjacent fuzzy sets C^j and C^{j+1} in the rules IR j and IR $^{j+1}$. The singleton output MFs \bar{d}_r^j and \bar{d}_r^{j+1} are determined in the following four cases:

- When $\mu_x(\mathbf{x}) > \mu_{C^j}(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) > \mu_{C^{j+1}}(y_d)$

$$\Rightarrow \bar{d}_r^j = d_r^j \quad \text{AND} \quad \bar{d}_r^{j+1} = d_r^{j+1}$$

- When $\mu_x(\mathbf{x}) > \mu_{C^j}(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) < \mu_{C^{j+1}}(y_d)$

$$\Rightarrow \bar{d}_r^j = d_r^j \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} \quad \text{AND} \quad \bar{d}_r^{j+1} = d_r^{j+1} \frac{\mu_{C^{j+1}}(y_d) - \mu_{C^j}(y_d)\mu_{x^{j+1}}(\mathbf{x})}{[\mu_{C^{j+1}}(y_d)]^2}$$

- When $\mu_x(\mathbf{x}) < \mu_{C^j}(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) > \mu_{C^{j+1}}(y_d)$

$$\Rightarrow \bar{d}_r^j = d_r^j \frac{1}{\mu_{C^j}(y_d)} \quad \text{AND} \quad \bar{d}_r^{j+1} = (d_r^{j+1} - d_r^j) \frac{\mu_{x^j}(\mathbf{x})}{\mu_{C^j}(y_d)}$$

- When $\mu_x(\mathbf{x}) < \mu_{C^j}(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) < \mu_{C^{j+1}}(y_d)$

$$\Rightarrow \text{undetermined}$$

For simplification, the notation k in the representations of $\mathbf{x}(k)$, $u(k)$, and $y(k)$ is omitted hereafter. The inversion process can be graphically illustrated in [Figure 4.1](#).

Proof The condition of $d_r^m \geq d_r^n \rightarrow c^m \geq c^n$ (or $d_r^m \leq d_r^n \rightarrow c^m \leq c^n$), $\forall r \in [1, p]$, $\forall m, n$ $m \neq n$ satisfies that the system output $y(k)$ has a monotonically increasing or decreasing relationship with each individual system input $u_r(k)$ (Babuška et al., 1995; Babuška, 1998; Piegat, 2001). Each output $y(k)$ corresponds to a unique input $u_r(k)$ in the consideration range. Thus, the forward-fuzzy rule-base forms a strictly monotonic relationship for each input–output pair and the fuzzy model can be inverted to calculate the required individual system input value $u_r(k)$ to achieve the desired output value $y_d(k)$.

Next, the theorem will be proved in each case as below:

- When $\mu_x(\mathbf{x}) > \mu_{C^j}(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) > \mu_{C^{j+1}}(y_d)$

$$\Rightarrow \bar{d}_r^j = d_r^j \quad \text{AND} \quad \bar{d}_r^{j+1} = d_r^{j+1}$$

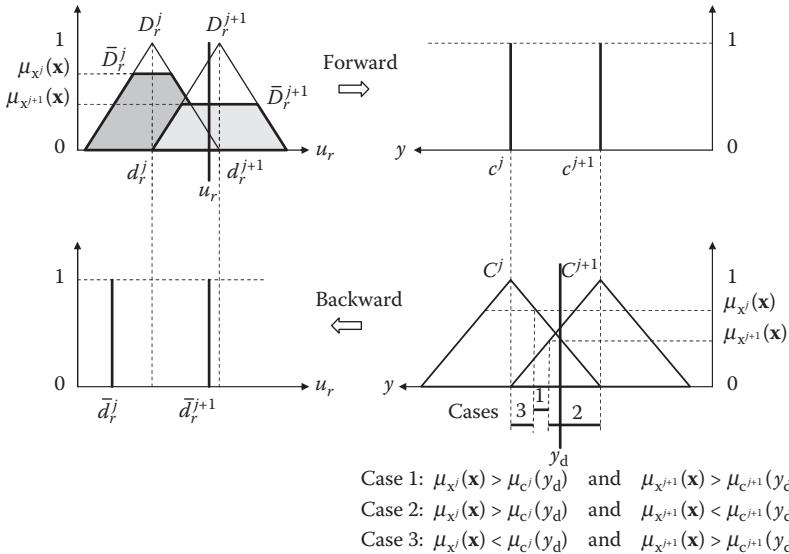


FIGURE 4.1 Graphical illustration for the fuzzy inversion process.

As illustrated in the lower part in Figure 4.1, for the fuzzy inverse model (Equation 4.7), the firing strengths of the desired output $y_d(k)$ in the j th rule and the $(j+1)$ th rule are $\mu_{C^j}(y_d)$ and $\mu_{C^{j+1}}(y_d)$, respectively. With fuzzy-mean defuzzification (Babuška, 1998), the individual system input can be calculated as

$$u_r = \frac{\sum_{i=1}^l \bar{d}_r^i \mu_{C^i}(y_d)}{\sum_{i=1}^l \mu_{C^i}(y_d)} = \frac{\bar{d}_r^j \mu_{C^j}(y_d) + \bar{d}_r^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d) + \mu_{C^{j+1}}(y_d)} \\ = \bar{d}_r^j \mu_{C^j}(y_d) + \bar{d}_r^{j+1} \mu_{C^{j+1}}(y_d) = d_r^j \mu_{C^j}(y_d) + d_r^{j+1} \mu_{C^{j+1}}(y_d) \quad (4.9)$$

where $\mu_{C^j}(y_d) + \mu_{C^{j+1}}(y_d) = 1$ from the definition of the triangular MF C^i . When the resultant system input vector $\mathbf{u} = [u_1, \dots, u_p]^T$ is fed into the system forward model (Equation 4.6), based on the monotonic property, for all the p inputs, the j th and $(j+1)$ th fuzzy sets will be fired. For example, \bar{D}_1^j and \bar{D}_1^{j+1} will be fired for $u_1(k), \dots, \bar{D}_p^j$ and \bar{D}_p^{j+1} will be fired for $u_p(k)$. Thus, there will be totally two fuzzy rules executed: R^j and R^{j+1} . The output of the fuzzy model $\hat{y}(k)$ is calculated as

$$\begin{aligned}\hat{y} &= \frac{\sum_{i=1}^l c^i \left[\mu_{\bar{D}_1^i}(u_1) \wedge \cdots \wedge \mu_{\bar{D}_p^i}(u_p) \right]}{\sum_{i=1}^l \left[\mu_{\bar{D}_1^i}(u_1) \wedge \cdots \wedge \mu_{\bar{D}_p^i}(u_p) \right]} \\ &= \frac{c^j \left[\mu_{\bar{D}_1^j}(u_1) \wedge \cdots \wedge \mu_{\bar{D}_p^j}(u_p) \right] + c^{j+1} \left[\mu_{\bar{D}_1^{j+1}}(u_1) \wedge \cdots \wedge \mu_{\bar{D}_p^{j+1}}(u_p) \right]}{\left[\mu_{\bar{D}_1^j}(u_1) \wedge \cdots \wedge \mu_{\bar{D}_p^j}(u_p) \right] + \left[\mu_{\bar{D}_1^{j+1}}(u_1) \wedge \cdots \wedge \mu_{\bar{D}_p^{j+1}}(u_p) \right]} \quad (4.10)\end{aligned}$$

Assume at the j th rule, the minimum value of the firing strength is $\mu_{\bar{D}_s^j}(u_s)$ as $\min_{r=1,\dots,p} [\mu_{\bar{D}_r^j}(u_r)] = \mu_{\bar{D}_1^j}(u_1) \wedge \cdots \wedge \mu_{\bar{D}_p^j}(u_p) = \mu_{\bar{D}_s^j}(u_s)$, and at the $(j+1)$ th rule, the minimum value of the firing strength is $\mu_{\bar{D}_t^{j+1}}(u_t)$ as $\min_{r=1,\dots,p} [\mu_{\bar{D}_r^{j+1}}(u_r)] = \mu_{\bar{D}_1^{j+1}}(u_1) \wedge \cdots \wedge \mu_{\bar{D}_p^{j+1}}(u_p) = \mu_{\bar{D}_t^{j+1}}(u_t)$. The output \hat{y} from the fuzzy inferencing calculation can be obtained as

$$\hat{y} = \frac{c^j \mu_{\bar{D}_s^j}(u_s) + c^{j+1} \mu_{\bar{D}_t^{j+1}}(u_t)}{\mu_{\bar{D}_s^j}(u_s) + \mu_{\bar{D}_t^{j+1}}(u_t)} = \frac{c^j \left[\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s) \right] + c^{j+1} \left[\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t) \right]}{\left[\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s) \right] + \left[\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t) \right]} \quad (4.11)$$

For the triangular MFs D_s^j and D_t^{j+1} ,

$$\begin{aligned}\mu_{D_s^j}(u_s) &= \frac{d_s^{j+1} - u_s}{d_s^{j+1} - d_s^j} = \frac{d_s^j \mu_C(y_d) + d_s^{j+1} \mu_{C^{j+1}}(y_d)}{d_s^{j+1} - d_s^j} \\ &= \frac{d_s^{j+1} [1 - \mu_{C^{j+1}}(y_d)] - d_s^j \mu_C(y_d)}{(d_s^{j+1} - d_s^j)} = \frac{(d_s^{j+1} - d_s^j) \mu_C(y_d)}{(d_s^{j+1} - d_s^j)} = \mu_C(y_d) \quad (4.12)\end{aligned}$$

$$\begin{aligned}\mu_{D_t^{j+1}}(u_t) &= \frac{u_t - d_t^j}{d_t^{j+1} - d_t^j} = \frac{\left[d_t^j \mu_C(y_d) + d_t^{j+1} \mu_{C^{j+1}}(y_d) \right] - d_t^j}{d_t^{j+1} - d_t^j} \\ &= \frac{d_t^j [\mu_C(y_d) - 1] + d_t^{j+1} \mu_{C^{j+1}}(y_d)}{(d_t^{j+1} - d_t^j)} = \frac{(d_t^{j+1} - d_t^j) \mu_{C^{j+1}}(y_d)}{(d_t^{j+1} - d_t^j)} = \mu_{C^{j+1}}(y_d) \quad (4.13)\end{aligned}$$

Thus, $\mu_{x^j}(\mathbf{x}) > \mu_C(y_d) = \mu_{D_s^j}(u_s)$ and $\mu_{x^{j+1}}(\mathbf{x}) > \mu_{C^{j+1}}(y_d) = \mu_{D_t^{j+1}}(u_t)$. From Equation 4.11, we derive that

$$\begin{aligned}
\hat{y} &= \frac{c^j \left[\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s) \right] + c^{j+1} \left[\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t) \right]}{\left[\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s) \right] + \left[\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t) \right]} = \frac{c^j \mu_{D_s^j}(u_s) + c^{j+1} \mu_{D_t^{j+1}}(u_t)}{\mu_{D_s^j}(u_s) + \mu_{D_t^{j+1}}(u_t)} \\
&= \frac{c^j \mu_C(y_d) + c^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_C(y_d) + \mu_{C^{j+1}}(y_d)} = \frac{c^j \frac{c^{j+1} - y_d}{c^{j+1} - c^j} + c^{j+1} \frac{y_d - c^j}{c^{j+1} - c^j}}{\frac{c^{j+1} - y_d}{c^{j+1} - c^j} + \frac{y_d - c^j}{c^{j+1} - c^j}} \\
&= \frac{c^j c^{j+1} - c^j y_d + c^{j+1} y_d - c^{j+1} c^j}{c^{j+1} - c^j} = y_d
\end{aligned} \tag{4.14}$$

2. When $\mu_x(\mathbf{x}) > \mu_C(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) < \mu_{C^{j+1}}(y_d)$

$$\Rightarrow \bar{d}_r^j = d_r^j \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} \quad \text{AND} \quad \bar{d}_r^{j+1} = d_r^{j+1} \frac{\mu_{C^{j+1}}(y_d) - \mu_C(y_d) \mu_{x^{j+1}}(\mathbf{x})}{[\mu_{C^{j+1}}(y_d)]^2}$$

From the fuzzy inverse model (Equation 4.7), in order to obtain the desired output $y_d(k)$, the individual system input can be calculated as

$$\begin{aligned}
u_r &= \frac{\sum_{i=1}^l \bar{d}_r^i \mu_{C^i}(y_d)}{\sum_{i=1}^l \mu_{C^i}(y_d)} = \frac{\bar{d}_r^j \mu_C(y_d) + \bar{d}_r^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_C(y_d) + \mu_{C^{j+1}}(y_d)} = \bar{d}_r^j \mu_C(y_d) + \bar{d}_r^{j+1} \mu_{C^{j+1}}(y_d) \\
&= d_r^j \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} \mu_C(y_d) + d_r^{j+1} \frac{\mu_{C^{j+1}}(y_d) - \mu_C(y_d) \mu_{x^{j+1}}(\mathbf{x})}{[\mu_{C^{j+1}}(y_d)]^2} \mu_{C^{j+1}}(y_d) \\
&= \frac{d_r^j \mu_{x^{j+1}}(\mathbf{x}) \mu_C(y_d) + d_r^{j+1} \mu_{C^{j+1}}(y_d) - d_r^{j+1} \mu_C(y_d) \mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)}
\end{aligned} \tag{4.15}$$

Thus, for the triangular MFs D_s^j and D_t^{j+1} ,

$$\begin{aligned}
\mu_{D_s^j}(u_s) &= \frac{d_s^{j+1} - u_s}{d_s^{j+1} - d_s^j} \\
&= \frac{d_s^{j+1} - \frac{d_s^j \mu_{x^{j+1}}(\mathbf{x}) \mu_C(y_d) + d_s^{j+1} \mu_{C^{j+1}}(y_d) - d_s^{j+1} \mu_C(y_d) \mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)}}{d_s^{j+1} - d_s^j} \\
&= \frac{d_s^{j+1} \mu_{C^{j+1}}(y_d) - d_s^j \mu_{x^{j+1}}(\mathbf{x}) \mu_C(y_d) - d_s^{j+1} \mu_{C^{j+1}}(y_d) + d_s^{j+1} \mu_C(y_d) \mu_{x^{j+1}}(\mathbf{x})}{(d_s^{j+1} - d_s^j) \mu_{C^{j+1}}(y_d)} \\
&= \frac{(d_s^{j+1} - d_s^j) \mu_{x^{j+1}}(\mathbf{x}) \mu_C(y_d)}{(d_s^{j+1} - d_s^j) \mu_{C^{j+1}}(y_d)} \\
&= \mu_C(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)}
\end{aligned} \tag{4.16}$$

$$\begin{aligned}
\mu_{D_t^{j+1}}(u_t) &= \frac{u_t - d_t^j}{d_t^{j+1} - d_t^j} \\
&= \frac{\frac{d_t^j \mu_{x^{j+1}}(\mathbf{x}) \mu_C(y_d) + d_t^{j+1} \mu_{C^{j+1}}(y_d) - d_t^{j+1} \mu_C(y_d) \mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} - d_t^j}{d_t^{j+1} - d_t^j} \\
&= \frac{d_t^j \mu_{x^{j+1}}(\mathbf{x}) \mu_C(y_d) + d_t^{j+1} \mu_{C^{j+1}}(y_d) - d_t^{j+1} \mu_C(y_d) \mu_{x^{j+1}}(\mathbf{x}) - d_t^j \mu_{C^{j+1}}(y_d)}{(d_t^{j+1} - d_t^j) \mu_{C^{j+1}}(y_d)} \\
&= \frac{\mu_{C^{j+1}}(y_d) (d_t^{j+1} - d_t^j) - \mu_{x^{j+1}}(\mathbf{x}) \mu_C(y_d) (d_t^{j+1} - d_t^j)}{(d_t^{j+1} - d_t^j) \mu_{C^{j+1}}(y_d)} \\
&= \frac{\mu_{C^{j+1}}(y_d) - \mu_{x^{j+1}}(\mathbf{x}) \mu_C(y_d)}{\mu_{C^{j+1}}(y_d)} \\
&= 1 - \mu_C(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)}
\end{aligned} \tag{4.17}$$

Since $\mu_x(\mathbf{x}) > \mu_C(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) < \mu_{C^{j+1}}(y_d)$, we can derive that

$$\mu_{D_s^j}(u_s) = \mu_C(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} < \mu_C(y_d) < \mu_x(\mathbf{x}) \tag{4.18}$$

$$\mu_{D_t^{j+1}}(u_t) = 1 - \mu_C(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} > 1 - \mu_C(y_d) = \mu_{C^{j+1}}(y_d) > \mu_{x^{j+1}}(\mathbf{x}) \tag{4.19}$$

Assume at the j th rule, $\min_{r=1,\dots,p} [\mu_{\bar{D}_r^j}(u_r)] = \mu_{\bar{D}_1^j}(u_1) \wedge \dots \wedge \mu_{\bar{D}_p^j}(u_p) = \mu_{\bar{D}_s^j}(u_s)$, and at the $(j+1)$ th rule, $\min_{r=1,\dots,p} [\mu_{\bar{D}_r^{j+1}}(u_r)] = \mu_{\bar{D}_1^{j+1}}(u_1) \wedge \dots \wedge \mu_{\bar{D}_p^{j+1}}(u_p) = \mu_{\bar{D}_t^{j+1}}(u_t)$. The output \hat{y} from the fuzzy model can be obtained through inferencing calculation as

$$\begin{aligned}
\hat{y} &= \frac{c^j \mu_{\bar{D}_s^j}(u_s) + c^{j+1} \mu_{\bar{D}_t^{j+1}}(u_t)}{\mu_{\bar{D}_s^j}(u_s) + \mu_{\bar{D}_t^{j+1}}(u_t)} = \frac{c^j [\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s)] + c^{j+1} [\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t)]}{[\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s)] + [\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t)]} \\
&= \frac{c^j \mu_{D_s^j}(u_s) + c^{j+1} \mu_{x^{j+1}}(\mathbf{x})}{\mu_{D_s^j}(u_s) + \mu_{x^{j+1}}(\mathbf{x})} = \frac{c^j \mu_C(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} + c^{j+1} \mu_{x^{j+1}}(\mathbf{x})}{\mu_C(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} + \mu_{x^{j+1}}(\mathbf{x})} \\
&= \frac{c^j \mu_C(y_d) + c^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_C(y_d) + \mu_{C^{j+1}}(y_d)} = \frac{c^j \cdot \frac{c^{j+1} - y_d}{c^{j+1} - c^j} + c^{j+1} \cdot \frac{y_d - c^j}{c^{j+1} - c^j}}{\frac{c^{j+1} - y_d}{c^{j+1} - c^j} + \frac{y_d - c^j}{c^{j+1} - c^j}}
\end{aligned} \tag{4.20}$$

3. When $\mu_x(\mathbf{x}) < \mu_C(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) > \mu_{C^{j+1}}(y_d)$

$$\Rightarrow \bar{d}_r^j = d_r^j \frac{1}{\mu_C(y_d)} \quad \text{AND} \quad \bar{d}_r^{j+1} = (d_r^{j+1} - d_r^j) \frac{\mu_{x^j}(\mathbf{x})}{\mu_C(y_d)}$$

From the fuzzy inverse model (Equation 4.7), in order to obtain the desired output $y_d(k)$, the individual system input can be calculated as

$$\begin{aligned}
u_r &= \frac{\sum_{i=1}^l \bar{d}_r^i \mu_{C^i}(y_d)}{\sum_{i=1}^l \mu_{C^i}(y_d)} = \frac{\bar{d}_r^j \mu_{C^j}(y_d) + \bar{d}_r^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d) + \mu_{C^{j+1}}(y_d)} = \bar{d}_r^j \mu_{C^j}(y_d) + \bar{d}_r^{j+1} \mu_{C^{j+1}}(y_d) \\
&= d_r^j \frac{1}{\mu_{C^j}(y_d)} \mu_{C^j}(y_d) + (d_r^{j+1} - d_r^j) \frac{\mu_{x^j}(\mathbf{x})}{\mu_{C^j}(y_d)} \mu_{C^{j+1}}(y_d) \\
&= \frac{d_r^j \mu_{C^j}(y_d) + d_r^{j+1} \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d) - d_r^j \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d)}
\end{aligned} \tag{4.21}$$

Thus,

$$\begin{aligned}
\mu_{D_s^j}(u_s) &= \frac{d_s^{j+1} - u_s}{d_s^{j+1} - d_s^j} \\
&= \frac{d_s^{j+1} - \frac{d_s^j \mu_{C^j}(y_d) + d_s^{j+1} \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d) - d_s^j \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d)}}{d_s^{j+1} - d_s^j} \\
&= \frac{d_s^{j+1} \mu_{C^j}(y_d) - d_s^j \mu_{C^j}(y_d) - d_s^{j+1} \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d) + d_s^j \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d)}{(d_s^{j+1} - d_s^j) \mu_{C^j}(y_d)} \\
&= \frac{(d_s^{j+1} - d_s^j) \mu_{C^j}(y_d) - (d_s^{j+1} - d_s^j) \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d)}{(d_s^{j+1} - d_s^j) \mu_{C^j}(y_d)} \\
&= 1 - \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(\mathbf{x})}{\mu_{C^j}(y_d)}
\end{aligned} \tag{4.22}$$

$$\begin{aligned}
\mu_{D_t^{j+1}}(u_t) &= \frac{u_t - d_t^j}{d_t^{j+1} - d_t^j} \\
&= \frac{d_t^j \mu_{C^j}(y_d) + d_t^{j+1} \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d) - d_t^j \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d)} - d_t^j \\
&= \frac{d_t^j \mu_{C^j}(y_d) + d_t^{j+1} \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d) - d_t^j \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d) - d_t^j \mu_{C^j}(y_d)}{(d_t^{j+1} - d_t^j) \mu_{C^j}(y_d)} \\
&= \frac{\mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d) (d_t^{j+1} - d_t^j)}{(d_t^{j+1} - d_t^j) \mu_{C^j}(y_d)} \\
&= \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(\mathbf{x})}{\mu_{C^j}(y_d)}
\end{aligned} \tag{4.23}$$

Since $\mu_{x^j}(\mathbf{x}) < \mu_{C^j}(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) > \mu_{C^{j+1}}(y_d)$, we can derive that

$$\mu_{D_s^j}(u_s) = 1 - \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(\mathbf{x})}{\mu_{C^j}(y_d)} > 1 - \mu_{C^{j+1}}(y_d) = \mu_{C^j}(y_d) > \mu_{x^j}(\mathbf{x}) \tag{4.24}$$

$$\mu_{D_t^{j+1}}(u_t) = \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(\mathbf{x})}{\mu_{C^j}(y_d)} < \mu_{C^{j+1}}(y_d) < \mu_{x^{j+1}}(\mathbf{x}) \tag{4.25}$$

Assume at the j th rule, $\min_{r=1,\dots,p} [\mu_{\bar{D}_r^j}(u_r)] = \mu_{\bar{D}_1^j}(u_1) \wedge \dots \wedge \mu_{\bar{D}_p^j}(u_p) = \mu_{\bar{D}_s^j}(u_s)$, and at the $(j+1)$ th rule, $\min_{r=1,\dots,p} [\mu_{\bar{D}_r^{j+1}}(u_r)] = \mu_{\bar{D}_1^{j+1}}(u_1) \wedge \dots \wedge \mu_{\bar{D}_p^{j+1}}(u_p) = \mu_{\bar{D}_t^{j+1}}(u_t)$. The output \hat{y} from the fuzzy model can be obtained through inferencing calculation as

$$\begin{aligned}\hat{y} &= \frac{c^j \mu_{\bar{D}_s^j}(u_s) + c^{j+1} \mu_{\bar{D}_t^{j+1}}(u_t)}{\mu_{\bar{D}_s^j}(u_s) + \mu_{\bar{D}_t^{j+1}}(u_t)} = \frac{c^j [\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s)] + c^{j+1} [\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t)]}{[\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s)] + [\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t)]} \\ &= \frac{c^j \mu_{x^j}(\mathbf{x}) + c^{j+1} \mu_{D_t^{j+1}}(u_t)}{\mu_{x^j}(\mathbf{x}) + \mu_{D_t^{j+1}}(u_t)} = \frac{c^j \mu_{x^j}(\mathbf{x}) + c^{j+1} \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(\mathbf{x})}{\mu_{C^j}(y_d)}}{\mu_{x^j}(\mathbf{x}) + \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(\mathbf{x})}{\mu_{C^j}(y_d)}} \\ &= \frac{c^j \mu_{C^j}(y_d) + c^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d) + \mu_{C^{j+1}}(y_d)} = \frac{c^j \cdot \frac{c^{j+1}-y_d}{c^{j+1}-c^j} + c^{j+1} \cdot \frac{y_d-c^j}{c^{j+1}-c^j}}{\frac{c^{j+1}-y_d}{c^{j+1}-c^j} + \frac{y_d-c^j}{c^{j+1}-c^j}} = y_d\end{aligned}\quad (4.26)$$

4. When $\mu_{x^j}(\mathbf{x}) < \mu_{C^j}(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) < \mu_{C^{j+1}}(y_d)$

\Rightarrow undetermined

In this case, the graphic illustration of fuzzy inversion process is shown in Figure 4.2.

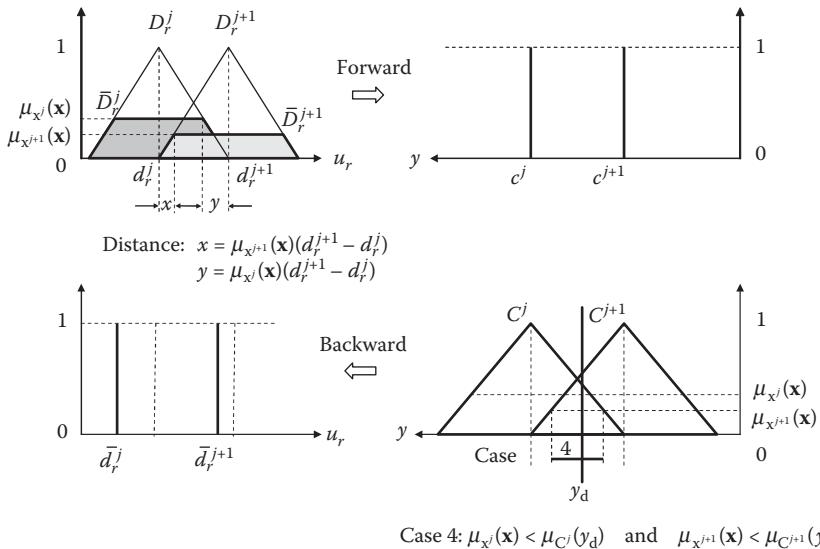


FIGURE 4.2 Graphical illustration for fuzzy inversion process in the case when $\mu_{x^j}(\mathbf{x}) < \mu_{C^j}(y_d)$ and $\mu_{x^{j+1}}(\mathbf{x}) < \mu_{C^{j+1}}(y_d)$.

In this case, the inverse solution depends on the ratio difference between $\frac{\mu_{C^j}(y_d)}{\mu_{C^{j+1}}(y_d)}$ and $\frac{\mu_{x^j}(\mathbf{x})}{\mu_{x^{j+1}}(\mathbf{x})}$ as listed below:

a. When $\frac{\mu_{C^j}(y_d)}{\mu_{C^{j+1}}(y_d)} < \frac{\mu_{x^j}(\mathbf{x})}{\mu_{x^{j+1}}(\mathbf{x})}$

$$\Rightarrow \bar{d}_r^j = d_r^j \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} \quad \text{AND} \quad \bar{d}_r^{j+1} = d_r^{j+1} \frac{\mu_{C^{j+1}}(y_d) - \mu_{C^j}(y_d)\mu_{x^{j+1}}(\mathbf{x})}{[\mu_{C^{j+1}}(y_d)]^2}$$

From the fuzzy inverse model (Equation 4.7), in order to obtain the desired output $y_d(k)$, the individual system input can be calculated as

$$\begin{aligned} u_r &= \frac{\sum_{i=1}^l \bar{d}_r^i \mu_{C^i}(y_d)}{\sum_{i=1}^l \mu_{C^i}(y_d)} = \frac{\bar{d}_r^j \mu_{C^j}(y_d) + \bar{d}_r^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d) + \mu_{C^{j+1}}(y_d)} = \bar{d}_r^j \mu_{C^j}(y_d) + \bar{d}_r^{j+1} \mu_{C^{j+1}}(y_d) \\ &= \frac{d_r^j \mu_{x^{j+1}}(\mathbf{x}) \mu_{C^j}(y_d) + d_r^{j+1} \mu_{C^{j+1}}(y_d) - d_r^{j+1} \mu_{C^j}(y_d) \mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} \end{aligned} \quad (4.27)$$

Thus,

$$\begin{aligned} \mu_{D_s^j}(u_s) &= \frac{d_s^{j+1} - u_s}{d_s^{j+1} - d_s^j} \\ &= \frac{d_s^{j+1} - \frac{d_s^j \mu_{x^{j+1}}(\mathbf{x}) \mu_{C^j}(y_d) + d_s^{j+1} \mu_{C^{j+1}}(y_d) - d_s^{j+1} \mu_{C^j}(y_d) \mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)}}{d_s^{j+1} - d_s^j} \\ &= \mu_{C^j}(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} \end{aligned} \quad (4.28)$$

$$\begin{aligned} \mu_{D_t^{j+1}}(u_t) &= \frac{u_t - d_t^j}{d_t^{j+1} - d_t^j} \\ &= \frac{d_t^j \mu_{x^{j+1}}(\mathbf{x}) \mu_{C^j}(y_d) + d_t^{j+1} \mu_{C^{j+1}}(y_d) - d_t^{j+1} \mu_{C^j}(y_d) \mu_{x^{j+1}}(\mathbf{x}) - d_t^j}{d_t^{j+1} - d_t^j} \\ &= 1 - \mu_{C^j}(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} \end{aligned} \quad (4.29)$$

Since $\mu_{x^j}(\mathbf{x}) < \mu_{C^j}(y_d)$, $\mu_{x^{j+1}}(\mathbf{x}) < \mu_{C^{j+1}}(y_d)$, and $\frac{\mu_{C^j}(y_d)}{\mu_{C^{j+1}}(y_d)} < \frac{\mu_{x^j}(\mathbf{x})}{\mu_{x^{j+1}}(\mathbf{x})}$, we can derive that

$$\mu_{D_s^j}(u_s) = \mu_{C^j}(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} < \mu_{x^j}(\mathbf{x}) \quad (4.30)$$

$$\mu_{D_t^{j+1}}(u_t) = 1 - \mu_{C^j}(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} > 1 - \mu_{C^j}(y_d) = \mu_{C^{j+1}}(y_d) > \mu_{x^{j+1}}(\mathbf{x}) \quad (4.31)$$

Assume at the j th rule, $\min_{r=1,\dots,p} [\mu_{\bar{D}_r^j}(u_r)] = \mu_{\bar{D}_1^j}(u_1) \wedge \dots \wedge \mu_{\bar{D}_p^j}(u_p) = \mu_{\bar{D}_s^j}(u_s)$, and at the $(j+1)$ th rule, $\min_{r=1,\dots,p} [\mu_{\bar{D}_r^{j+1}}(u_r)] = \mu_{\bar{D}_1^{j+1}}(u_1) \wedge \dots \wedge \mu_{\bar{D}_p^{j+1}}(u_p) = \mu_{\bar{D}_t^{j+1}}(u_t)$. The output \hat{y} from the fuzzy model can be obtained through inferencing calculation as

$$\begin{aligned} \hat{y} &= \frac{c^j \mu_{\bar{D}_s^j}(u_s) + c^{j+1} \mu_{\bar{D}_t^{j+1}}(u_t)}{\mu_{\bar{D}_s^j}(u_s) + \mu_{\bar{D}_t^{j+1}}(u_t)} = \frac{c^j [\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s)] + c^{j+1} [\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t)]}{[\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s)] + [\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t)]} \\ &= \frac{c^j \mu_{D_s^j}(u_s) + c^{j+1} \mu_{x^{j+1}}(\mathbf{x})}{\mu_{D_s^j}(u_s) + \mu_{x^{j+1}}(\mathbf{x})} = \frac{c^j \mu_{C^j}(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} + c^{j+1} \mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^j}(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} + \mu_{x^{j+1}}(\mathbf{x})} \\ &= \frac{c^j \mu_{C^j}(y_d) + c^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d) + \mu_{C^{j+1}}(y_d)} = y_d \end{aligned} \quad (4.32)$$

b. When $\frac{\mu_{C^j}(y_d)}{\mu_{C^{j+1}}(y_d)} > \frac{\mu_{x^j}(\mathbf{x})}{\mu_{x^{j+1}}(\mathbf{x})}$

$$\Rightarrow \bar{d}_r^j = d_r^j \frac{1}{\mu_{C^j}(y_d)} \quad \text{AND} \quad \bar{d}_r^{j+1} = (d_r^{j+1} - d_r^j) \frac{\mu_{x^j}(\mathbf{x})}{\mu_{C^j}(y_d)}$$

From the fuzzy inverse model (Equation 4.7), in order to obtain the desired output $y_d(k)$, the individual system input can be calculated as

$$\begin{aligned} u_r &= \frac{\sum_{i=1}^l \bar{d}_r^i \mu_{C^i}(y_d)}{\sum_{i=1}^l \mu_{C^i}(y_d)} = \frac{\bar{d}_r^j \mu_{C^j}(y_d) + \bar{d}_r^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d) + \mu_{C^{j+1}}(y_d)} = \bar{d}_r^j \mu_{C^j}(y_d) + \bar{d}_r^{j+1} \mu_{C^{j+1}}(y_d) \\ &= \frac{d_r^j \mu_{C^j}(y_d) + d_r^{j+1} \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d) - d_r^j \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d)} \end{aligned} \quad (4.33)$$

Thus,

$$\begin{aligned} \mu_{D_s^j}(u_s) &= \frac{d_s^{j+1} - u_s}{d_s^{j+1} - d_s^j} \\ &= \frac{d_s^{j+1} - \frac{d_r^j \mu_{C^j}(y_d) + d_r^{j+1} \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d) - d_r^j \mu_{x^j}(\mathbf{x}) \mu_{C^{j+1}}(y_d)}{\mu_{C^j}(y_d)}}{d_s^{j+1} - d_s^j} \\ &= 1 - \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(\mathbf{x})}{\mu_{C^j}(y_d)} \end{aligned} \quad (4.34)$$

$$\begin{aligned}
\mu_{D_t^{j+1}}(u_t) &= \frac{u_t - d_t^j}{d_t^{j+1} - d_t^j} \\
&= \frac{\frac{d_t^j \mu_C(y_d) + d_t^{j+1} \mu_{x^j}(x) \mu_{C^{j+1}}(y_d) - d_t^j \mu_{x^j}(x) \mu_{C^{j+1}}(y_d)}{\mu_C(y_d)}}{d_t^{j+1} - d_t^j} - d_t^j \\
&= \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(x)}{\mu_C(y_d)}
\end{aligned} \tag{4.35}$$

Since $\mu_{x^j}(x) < \mu_C(y_d)$, $\mu_{x^{j+1}}(x) < \mu_{C^{j+1}}(y_d)$, and $\frac{\mu_C(y_d)}{\mu_{C^{j+1}}(y_d)} > \frac{\mu_{x^j}(x)}{\mu_{x^{j+1}}(x)}$, we can derive that

$$\mu_{D_s^j}(u_s) = 1 - \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(x)}{\mu_C(y_d)} > 1 - \mu_{C^{j+1}}(y_d) = \mu_C(y_d) > \mu_{x^j}(x) \tag{4.36}$$

$$\mu_{D_t^{j+1}}(u_t) = \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(x)}{\mu_C(y_d)} < \mu_{x^{j+1}}(x) \tag{4.37}$$

Assume at the j th rule, $\min_{r=1,\dots,p} [\mu_{\overline{D}_r^j}(u_r)] = \mu_{\overline{D}_1^j}(u_1) \wedge \dots \wedge \mu_{\overline{D}_p^j}(u_p) = \mu_{\overline{D}_s^j}(u_s)$, and at the $(j+1)$ th rule, $\min_{r=1,\dots,p} [\mu_{\overline{D}_r^{j+1}}(u_r)] = \mu_{\overline{D}_1^{j+1}}(u_1) \wedge \dots \wedge \mu_{\overline{D}_p^{j+1}}(u_p) = \mu_{\overline{D}_t^{j+1}}(u_t)$. The output \hat{y} from the fuzzy model can be obtained through inferencing calculation as

$$\begin{aligned}
\hat{y} &= \frac{c^j \mu_{\overline{D}_s^j}(u_s) + c^{j+1} \mu_{\overline{D}_t^{j+1}}(u_t)}{\mu_{\overline{D}_s^j}(u_s) + \mu_{\overline{D}_t^{j+1}}(u_t)} = \frac{c^j [\mu_{x^j}(x) \wedge \mu_{D_s^j}(u_s)] + c^{j+1} [\mu_{x^{j+1}}(x) \wedge \mu_{D_t^{j+1}}(u_t)]}{[\mu_{x^j}(x) \wedge \mu_{D_s^j}(u_s)] + [\mu_{x^{j+1}}(x) \wedge \mu_{D_t^{j+1}}(u_t)]} \\
&= \frac{c^j \mu_{x^j}(x) + c^{j+1} \mu_{D_t^{j+1}}(u_t)}{\mu_{x^j}(x) + \mu_{D_t^{j+1}}(u_t)} = \frac{c^j \mu_{x^j}(x) + c^{j+1} \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(x)}{\mu_C(y_d)}}{\mu_{x^j}(x) + \mu_{C^{j+1}}(y_d) \frac{\mu_{x^j}(x)}{\mu_C(y_d)}} \\
&= \frac{c^j \mu_C(y_d) + c^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_C(y_d) + \mu_{C^{j+1}}(y_d)} = y_d
\end{aligned} \tag{4.38}$$

c. When $\frac{\mu_C(y_d)}{\mu_{C^{j+1}}(y_d)} = \frac{\mu_{x^j}(x)}{\mu_{x^{j+1}}(x)}$

\Rightarrow multiple solutions

In this situation, as long as the system input variable $u_r(k)$ satisfies

$$d_r^j [1 - \mu_{x^{j+1}}(x)] + d_r^{j+1} \mu_{x^{j+1}}(x) \leq u_r \leq d_r^j \mu_{x^j}(x) + d_r^{j+1} [1 - \mu_{x^j}(x)] \tag{4.39}$$

the output \hat{y} will be calculated as $\hat{y} = y_d$. From [Figure 4.2](#),

$$\frac{x}{d_r^{j+1} - d_r^j} = \frac{\mu_{x^{j+1}}(\mathbf{x})}{1} \Rightarrow x = \mu_{x^{j+1}}(\mathbf{x})(d_r^{j+1} - d_r^j) \quad (4.40)$$

$$\frac{y}{d_r^{j+1} - d_r^j} = \frac{\mu_{x^j}(\mathbf{x})}{1} \Rightarrow y = \mu_{x^j}(\mathbf{x})(d_r^{j+1} - d_r^j) \quad (4.41)$$

Within the range of

$$d_r^j + x \leq u_r \leq d_r^{j+1} - y \quad (4.42)$$

$$d_r^j + \mu_{x^{j+1}}(\mathbf{x})(d_r^{j+1} - d_r^j) \leq u_r \leq d_r^{j+1} - \mu_{x^j}(\mathbf{x})(d_r^{j+1} - d_r^j) \quad (4.43)$$

$$d_r^j[1 - \mu_{x^{j+1}}(\mathbf{x})] + d_r^{j+1}\mu_{x^{j+1}}(\mathbf{x}) \leq u_r \leq d_r^j\mu_{x^j}(\mathbf{x}) + d_r^{j+1}[1 - \mu_{x^j}(\mathbf{x})] \quad (4.44)$$

we have

$$\mu_{D_s^j}(u_s) > \mu_{x^j}(\mathbf{x}) \quad (4.45)$$

$$\mu_{D_t^{j+1}}(u_t) > \mu_{x^{j+1}}(\mathbf{x}) \quad (4.46)$$

Assume at the j th rule, $\min_{r=1,\dots,p} [\mu_{\bar{D}_r^j}(u_r)] = \mu_{\bar{D}_1^j}(u_1) \wedge \dots \wedge \mu_{\bar{D}_p^j}(u_p) = \mu_{\bar{D}_s^j}(u_s)$, and at the $(j+1)$ th rule, $\min_{r=1,\dots,p} [\mu_{\bar{D}_r^{j+1}}(u_r)] = \mu_{\bar{D}_1^{j+1}}(u_1) \wedge \dots \wedge \mu_{\bar{D}_p^{j+1}}(u_p) = \mu_{\bar{D}_t^{j+1}}(u_t)$. Since $\frac{\mu_C(y_d)}{\mu_{C^{j+1}}(y_d)} = \frac{\mu_{x^j}(\mathbf{x})}{\mu_{x^{j+1}}(\mathbf{x})}$, the output \hat{y} from the fuzzy model can be obtained through inferencing calculation as

$$\begin{aligned} \hat{y} &= \frac{c^j \mu_{\bar{D}_s^j}(u_s) + c^{j+1} \mu_{\bar{D}_t^{j+1}}(u_t)}{\mu_{\bar{D}_s^j}(u_s) + \mu_{\bar{D}_t^{j+1}}(u_t)} = \frac{c^j [\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s)] + c^{j+1} [\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t)]}{[\mu_{x^j}(\mathbf{x}) \wedge \mu_{D_s^j}(u_s)] + [\mu_{x^{j+1}}(\mathbf{x}) \wedge \mu_{D_t^{j+1}}(u_t)]} \\ &= \frac{c^j \mu_{x^j}(\mathbf{x}) + c^{j+1} \mu_{x^{j+1}}(\mathbf{x})}{\mu_{x^j}(\mathbf{x}) + \mu_{x^{j+1}}(\mathbf{x})} = \frac{c^j \mu_C(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} + c^{j+1} \mu_{x^{j+1}}(\mathbf{x})}{\mu_C(y_d) \frac{\mu_{x^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(y_d)} + \mu_{x^{j+1}}(\mathbf{x})} \\ &= \frac{c^j \mu_C(y_d) + c^{j+1} \mu_{C^{j+1}}(y_d)}{\mu_C(y_d) + \mu_{C^{j+1}}(y_d)} = y_d \end{aligned} \quad (4.47)$$

From the derivation, the fuzzy inverse model for each individual input can be automatically constructed based on the fuzzy forward model. Once the desired system output $y_d(k)$ is given and the current state variables $[x_1(k), \dots, x_n(k)]^T$ are measurable, the required plant inputs $[u_1(k), \dots, u_p(k)]^T$ can be obtained simultaneously by the fuzzy inferencing calculation.

4.2 SIMULATION EXAMPLES

In recent years, robot manipulators have been used extensively in industry, which are composed of a serial chain of rigid links connected through each revolute or prismatic joint. The desired path of the end tip is usually specified in Cartesian

coordinate space. However, in the trajectory control, it is often desired to perform control actions in joint space instead. Thus, solving the joint movement from the given end-tip trajectory is required in many cases. The problem of finding a suitable solution that transforms the desired tool movement (position and orientation) to joint movement (rotational or longitudinal) is called inverse kinematics calculation. By doing so, the joint movement can be appropriately determined to satisfy the desired trajectory of the end-effector. Analytical and numerical techniques are two most common approaches to solving the inverse kinematics problems (Kucuk and Bingul, 2005). Analytical techniques use geometric and algebraic identities to solve a set of nonlinear and coupled equations, which in turn makes the inverse kinematics problem solvable only for a very small class of simple manipulators (Alsina and Gehlot, 1995; Kucuk and Bingul, 2005). Numerical solutions rely on interactive procedures to solve those equations, which are slow for practical applications (Manocha and Canny, 1994). Thus, the real-time calculation of inverse kinematics of robot manipulators is widely recognized as a challenging problem. In the following examples, the proposed fuzzy inverse method is shown to be an effective way of finding the inverse kinematics solution. Unlike the traditional method in trying to solve the complex coupled nonlinear inverse kinematics equations, the required joint movements can be determined simultaneously by the fuzzy inverse model inferencing to realize the desired robot endpoint trajectory.

4.2.1 TWO-LINK ROBOT MANIPULATOR

Consider a two-link robot manipulator as shown in Figure 4.3. The first joint is assumed to be fixed at the origin in the Cartesian coordinate system. The free tip of the second link (x , y) describes the endpoint trajectory, and can be expressed as functions of the lengths of the two arms l_1 , l_2 and the two joint angles θ_1 , θ_2 as

$$\begin{aligned} x &= l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\ y &= l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \end{aligned} \quad (4.48)$$

where $l_1 = 5$ in. and $l_2 = 5$ in. Suppose the endpoint of the second link moves along a slot ($y = 6$ in.) reciprocally. The desired endpoint trajectory along x -direction is a

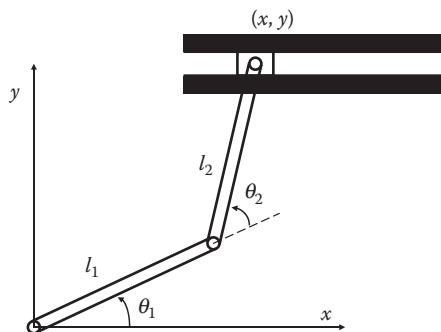


FIGURE 4.3 Two-link robot manipulator.

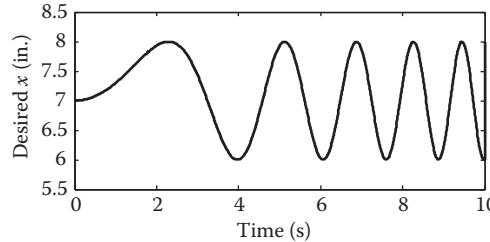


FIGURE 4.4 Desired endpoint movement along x -direction of a two-link robot manipulator.

sinusoid function of time with continuously increasing frequency and the amplitude of 1 in., as shown in Figure 4.4. Suppose the rotational angle of the first link θ_1 is limited within the range of $[15^\circ, 35^\circ]$. The objective is to compute the required joint angles θ_1, θ_2 in order to achieve the desired endpoint trajectory.

By using the proposed fuzzy inverse method, the robot manipulator movement can be modeled in fuzzy domain and the inverse model can be automatically constructed, from which the required joint angle values θ_1, θ_2 can be derived from fuzzy inferencing calculation. First, the robot arm movement is modeled as a fuzzy MISO system. The inputs are the joint angles θ_1, θ_2 and the output is the x -coordinate of the endpoint, where the y -coordinate is fixed at $y = 6$ in. The fuzzy forward model is constructed based on 300 pairs of input–output training data as

$$R^i: \text{IF } \theta_1(k) = D_r^i \text{ AND } \theta_2(k) = D_s^i \text{ THEN } x(k) = c^i \quad (4.49)$$

where the centers d_r^i of the triangular MFs D_r^i and the position of the output singleton MF c^i are listed in Table 4.1.

And then the fuzzy inverse model is automatically constructed for the two joint angles θ_1 and θ_2 based on the procedure described in Section 4.1:

$$\begin{aligned} IR^i: \text{IF } x(k) = E^i \text{ THEN } \theta_1(k) &= f_1^i \\ \text{IF } x(k) = E^i \text{ THEN } \theta_2(k) &= f_2^i \end{aligned} \quad (4.50)$$

where the center e^i of the input triangle MF E^i and the position of the output singleton MF f_r^i are listed in Table 4.2.

The invertibility of the system is checked and the required joint angles for the two joints θ_1 and θ_2 are calculated individually as shown in Figure 4.5. In Figure 4.6,

TABLE 4.1

Parameters for the Fuzzy Forward Model of a Two-Link Robot Manipulator

Rule Number	1	2	3	4	5	6
Center d_1^i ($^\circ$)	12.0205	16.6023	18.9005	23.4894	28.1041	34.9520
Center d_2^i ($^\circ$)	70.7879	57.3877	50.6876	37.2873	23.8871	3.7867
Position c^i (cm)	5.5184	6.1708	6.4743	7.0256	7.4898	7.9932

TABLE 4.2
**Parameters for the Obtained Fuzzy Inverse Model of a Two-Link
 Robot Manipulator**

Rule Number	1	2	3	4	5	6
Center e^i (cm)	5.5184	6.1708	6.4743	7.0256	7.4898	7.9932
Position f_1^i ($^\circ$)	12.0205	16.6023	18.9005	23.4894	28.1041	34.9520
Position f_2^i ($^\circ$)	70.7879	57.3877	50.6876	37.2873	23.8871	3.7867

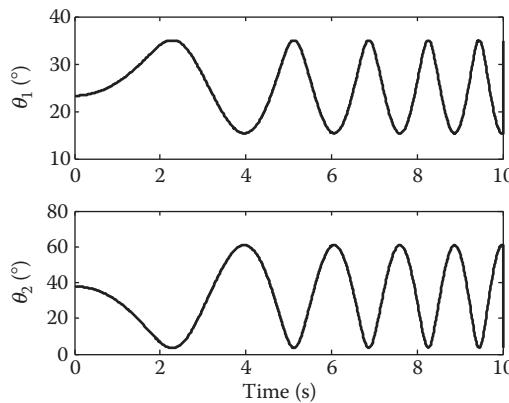


FIGURE 4.5 Required joint angles for the two joints θ_1 and θ_2 of a two-link robot manipulator.

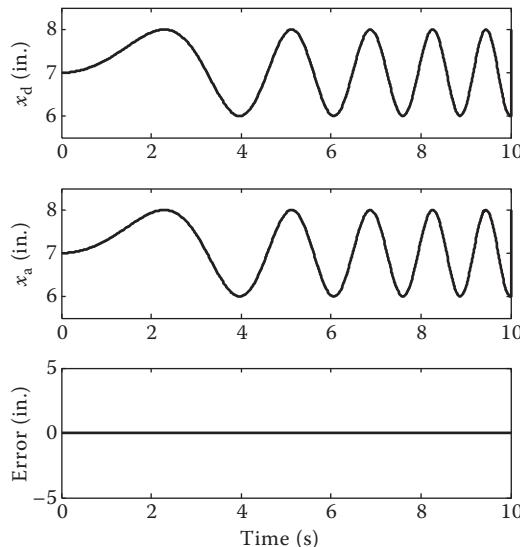


FIGURE 4.6 Desired trajectory, actual trajectory, and the error of a two-link robot manipulator.

the first plot is the desired endpoint movement along x -direction. The second plot is the actual endpoint movement along x -direction, which is calculated from the robot physical model using the obtained joint angles θ_1, θ_2 . The last plot shows the error between the desired endpoint trajectory and the actual endpoint trajectory. As illustrated in these figures, the magnitude of the error is negligible compared with the magnitude of the desired x -coordinate and the mean squared error (MSE) over the simulation is calculated to be 5.174×10^{-6} , thus illustrating the effectiveness of the fuzzy inverse model construction method in robot manipulator's trajectory planning.

4.2.2 FIVE-LINK ADEPTONE INDUSTRY ROBOT MANIPULATOR

In this example, a 5 degree-of-freedom AdeptOne industry robot manipulator is considered as shown in Figure 4.7, where the five joint variables are defined as $q = [\theta_1 \theta_2 d_3 \theta_4 \theta_5]^T$. The goal of the trajectory planning is to move the endpoint of the manipulator along a straight line in the three-dimensional space from the starting point (10, 10, 15) cm to the final point (20, 20, 25) cm. The kinematics parameter matrix of the five-link robot manipulator is provided in [Table 4.3](#), where $l_1 = 45$ cm, $l_2 = 35$ cm, $d_1 = 50$ cm, and $h = 10$ cm.

In order to move the manipulator endpoint from (10, 10, 15) to (20, 20, 25) cm in the three-dimensional space, the inverse kinematics solutions for the first three joints θ_1, θ_2, d_3 are required, where the first two joint variables θ_1 and θ_2 are rotational variables to determine the endpoint movement in the horizontal plane. The third joint

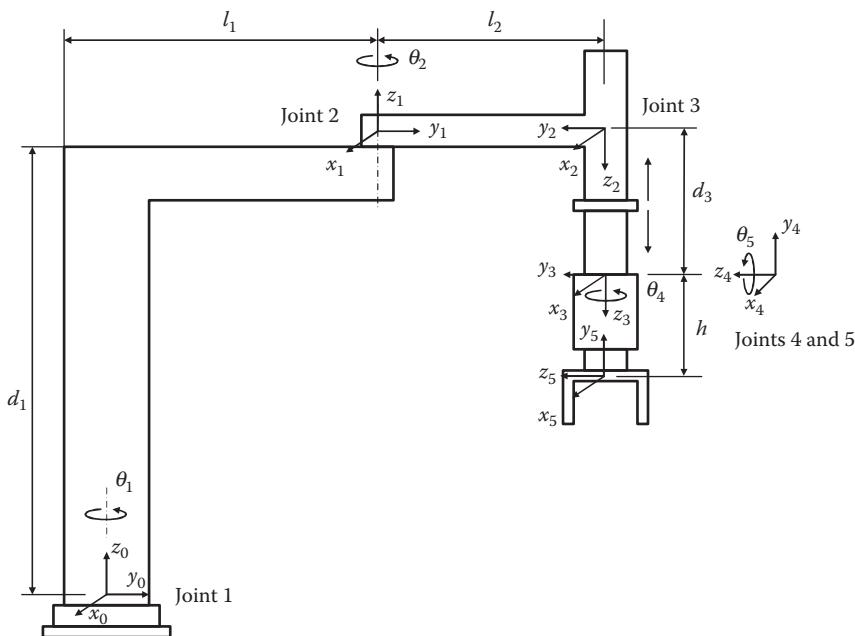


FIGURE 4.7 Five-link AdeptOne robot manipulator.

TABLE 4.3
**Denavit–Hartenberg Kinematics Parameter
Matrix of a Five-Link Robot Manipulator**

Link Number	α_i (°)	a_i (cm)	d_i (cm)	θ_i (°)
1	0	l_1	d_1	θ_1
2	-180	l_2	0	θ_2
3	0	0	d_3	0
4	-90	0	h	θ_4
5	0	0	0	θ_5

variable d_3 can be directly obtained from the desired vertical movement of the endpoint along z -direction. The last two joint variables θ_4 and θ_5 can be maintained at their initial values.

In the fuzzy forward model construction, the inputs are specified as the first two joint angle variables θ_1 and θ_2 , and the output is the x -coordinate (or y -coordinate) of the endpoint. And then the end-tip movement of the five-link robot manipulator is modeled using 300 input–output data pairs as the following fuzzy system:

$$R^i: \text{IF } \theta_1(k) = D_1^i \text{ AND } \theta_2(k) = D_2^i \text{ THEN } x(k) = c^i \quad (4.51)$$

where

the centers d_r^i of the triangular MFs D_r^i
the position of the output singleton MF c^i are listed in Table 4.4

The fuzzy inverse model is automatically constructed for the two joint angles θ_1 and θ_2 as

$$\begin{aligned} IR^i: \text{IF } x(k) = E^i \text{ THEN } \theta_1(k) &= f_1^i \\ \text{IF } x(k) = E^i \text{ THEN } \theta_2(k) &= f_2^i \end{aligned} \quad (4.52)$$

where

the center e^i of the input triangle MF E^i
the position of the output singleton MF f_r^i are listed in Table 4.5

TABLE 4.4
Parameters for the Fuzzy Forward Model of a Five-Link Robot Manipulator

Rule Number	1	2	3	4	5	6
Center d_1^i (°)	82.0026	84.8203	86.2117	89.0542	91.8711	96.1005
Center d_2^i (°)	-166.0004	-164.4223	-163.3021	-160.6819	-157.1330	-141.0007
Position c^i (cm)	9.9197	10.3863	10.7905	11.7803	13.1897	19.9687

TABLE 4.5**Parameters for the Obtained Fuzzy Inverse Model of a Five-Link Robot Manipulator**

Rule Number	1	2	3	4	5	6
Center e^i (cm)	9.9197	10.3863	10.7905	11.7803	13.1897	19.9687
Position f_1^i ($^{\circ}$)	82.0026	84.8203	86.2117	89.0542	91.8711	96.1005
Position f_2^i ($^{\circ}$)	-166.0004	-164.4223	-163.3021	-160.6819	-157.1330	-141.0007

The required joint angles/displacement for the three joints θ_1 , θ_2 , and d_3 are illustrated in Figure 4.8. The actual endpoint movement along x -, y -, and z - directions is shown in Figure 4.9, and the error between the desired endpoint trajectory and the actual endpoint trajectory along each direction is plotted in Figure 4.10.

For performance comparison, an algebraic approach (Goel, 1988) to obtain the inverse kinematics of the five-link AdeptOne robot manipulator is used and the required joint angles/displacement are calculated and simulated. This method utilizes a set of homogeneous transformation matrix $A_{i-1,i}(\theta_i)$ to transform the movement from the $(i-1)$ th joint space to the i th joint space, so the position and the orientation of the end-tip can be sequentially determined from the base segment

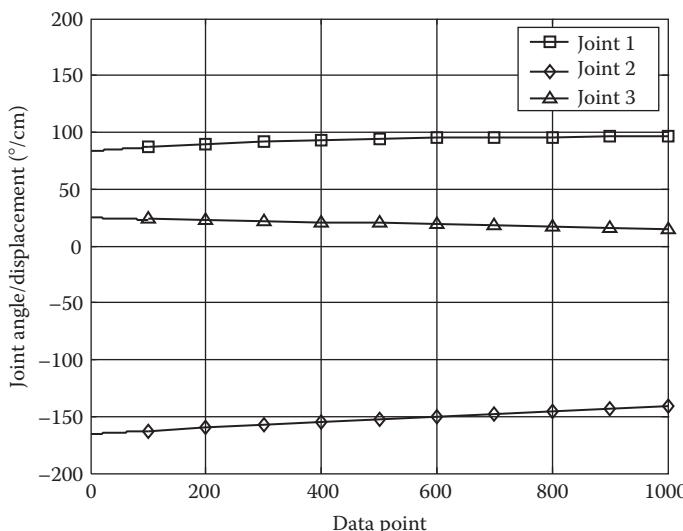


FIGURE 4.8 Required joint angles/displacement of the three joints θ_1 , θ_2 , and d_3 of a five-link robot manipulator based on fuzzy inverse method.

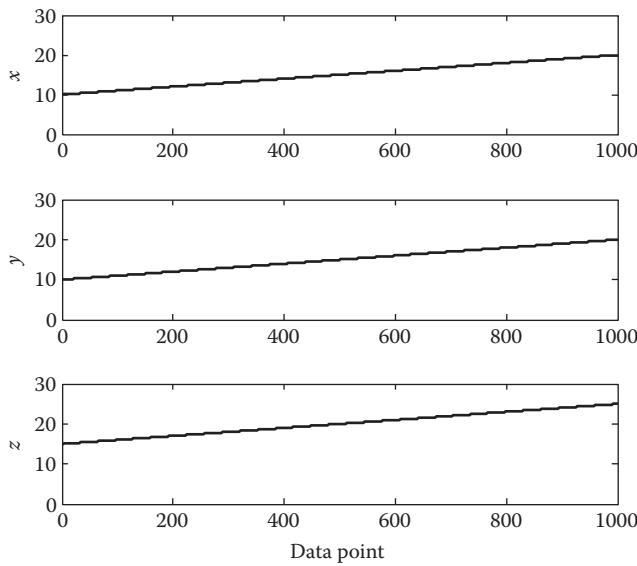


FIGURE 4.9 Actual endpoint movement along x -, y -, and z -directions of a five-link robot manipulator based on fuzzy inverse method.

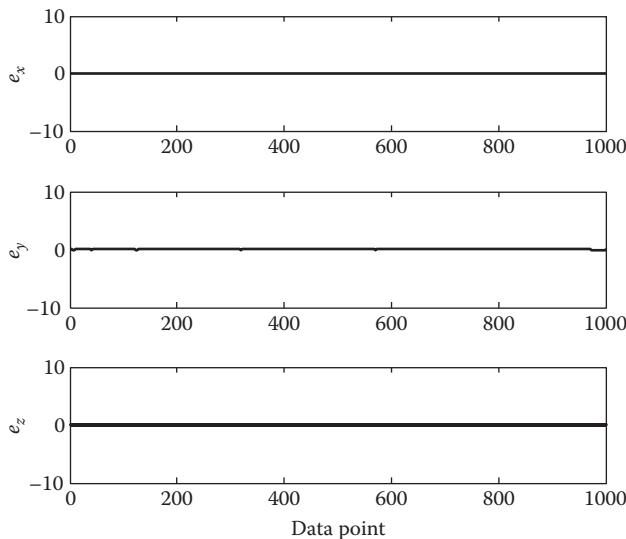


FIGURE 4.10 Error along x -, y -, and z -directions of a five-link robot manipulator based on fuzzy inverse method.

to the endpoint in the inertial coordinate frame. The transformation matrices are listed as follows:

$$\mathbf{A}_{0,1}(\theta_1) = \begin{pmatrix} \cos \theta_1 & -\sin \theta_1 & 0 & l_1 \cos \theta_1 \\ \sin \theta_1 & \cos \theta_1 & 0 & l_1 \sin \theta_1 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{A}_{1,2}(\theta_2) = \begin{pmatrix} \cos \theta_2 & \sin \theta_2 & 0 & l_2 \cos \theta_2 \\ \sin \theta_2 & -\cos \theta_2 & 0 & l_2 \sin \theta_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{A}_{2,3}(d_3) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{A}_{3,4}(\theta_4) = \begin{pmatrix} \cos \theta_4 & 0 & -\sin \theta_4 & 0 \\ \sin \theta_4 & 0 & \cos \theta_4 & 0 \\ 0 & -1 & 0 & h \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{A}_{4,5}(\theta_5) = \begin{pmatrix} \cos \theta_5 & -\sin \theta_5 & 0 & 0 \\ \sin \theta_5 & \cos \theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Furthermore, the multiplication of all transformation matrixes describes the movement of the end-tip in the base frame space as

$$\mathbf{A}_{0,5} = \mathbf{A}_{0,1}(\theta_1) \cdot \mathbf{A}_{1,2}(\theta_2) \cdot \mathbf{A}_{2,3}(d_3) \cdot \mathbf{A}_{3,4}(\theta_4) \cdot \mathbf{A}_{4,5}(\theta_5) \quad (4.53)$$

Define $\gamma = \sqrt{x^2 + y^2}$, $C_2 = \frac{1}{2l_1l_2}(\gamma^2 - l_1^2 - l_2^2)$, and $S_2 = \pm\sqrt{1 - C_2^2}$. The required joint angle θ_2 is derived as $\theta_2 = \text{atan } 2(S_2, C_2)$. Define $C_1 = \frac{1}{\gamma}[l_2S_2y + (l_1 + l_2C_2)x]$ and $S_1 = \frac{1}{\gamma}[(l_1 + l_2C_2)y - l_2S_2x]$. The required joint angle θ_1 is derived as $\theta_1 = \text{atan } 2(S_1, C_1)$. The joint displacement d_3 is derived as $d_3 = d_1 - z - h$.

The calculated joint angles/displacement for the three joints θ_1 , θ_2 , and d_3 are shown in [Figure 4.11](#). The actual endpoint movement along each direction is plotted in [Figure 4.12](#), and the error is plotted in [Figure 4.13](#).

The inverse kinematics performance with the proposed fuzzy inverse method and the analytical method (Goel, 1988) is compared in terms of the MSE over the simulation and the results are listed in [Table 4.6](#).

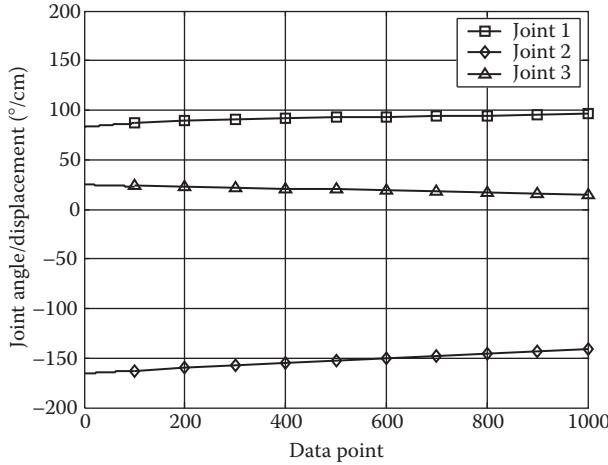


FIGURE 4.11 Required joint angles/displacement of the three joints θ_1 , θ_2 , and d_3 of a five-link robot manipulator based on algebraic inverse kinematics calculation.

Since the algebraic approach utilizes the physical model of the robot to derive the solution for the inverse kinematics, the error is only coming from the round-off in the computer calculation, which is obviously less than the one obtained from the proposed fuzzy inverse method. However, the algebraic approach method is directly calculated based on the physical model and is only suitable to solve the inverse kinematics problem of simple robot manipulators. As the complexity of the robot

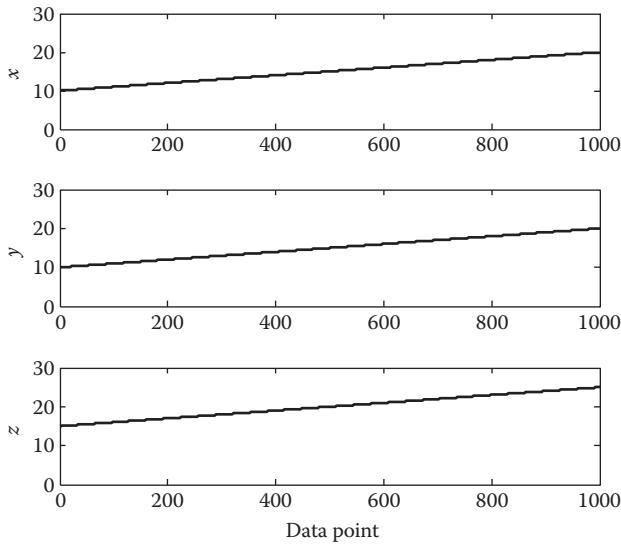


FIGURE 4.12 Actual endpoint movement along x -, y -, and z -directions of a five-link robot manipulator based on algebraic inverse kinematics calculation.

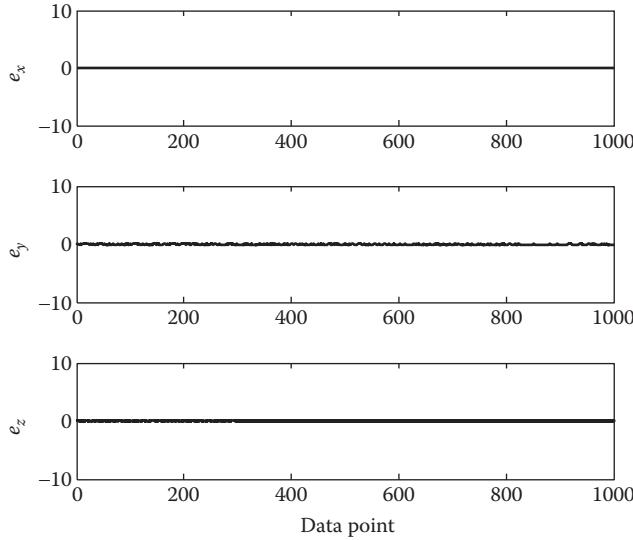


FIGURE 4.13 Error along x -, y -, and z -directions of a five-link robot manipulator based on algebraic inverse kinematics calculation.

TABLE 4.6
Comparison of the MSE

	MSE (x-Direction)	MSE (y-Direction)	MSE (z-Direction)
Fuzzy inverse model	2.700×10^{-3}	2.900×10^{-3}	4.995×10^{-5}
Algebraic approach (Goel, 1988)	3.215×10^{-29}	3.127×10^{-29}	4.995×10^{-5}

kinematics increases, an analytical solution is difficult to formulate in a straightforward way.

4.2.3 FOUR-LINK ADEPTONE INDUSTRY ROBOT MANIPULATOR

In this example, a four-link AdeptOne industry robot manipulator is chosen to evaluate the performance of the proposed fuzzy inverse method. The robot is a typical selective compliance assembly robot arm (SCARA) in industrial applications, whose link-coordinate is shown in Figure 4.14. The Denavit–Hartenberg kinematics parameter matrix of the 4 degree-of-freedom SCARA robot is given in Table 4.7, where $l_1 = 42.5$ mm, $l_2 = 37.5$ mm, $d_1 = 87.7$ mm, and $d_4 = 20$ mm. The endpoint movement can be defined by a vector of four joint variables as $q = [\theta_1 \ \theta_2 \ d_3 \ \theta_4]^T$. The first two joint variables θ_1 and θ_2 are rotational variables, which determine the endpoint horizontal movement, the third joint variable d_3 is a longitudinal variable affecting the vertical movement along z -direction, and the last joint variable θ_4 is a rotational variable to adjust the endpoint orientation.

Suppose the robot manipulator is required to move along a circular trajectory described by $(x - 43)^2 + (y - 43)^2 = 5^2$ in the horizontal plane. Only the first two

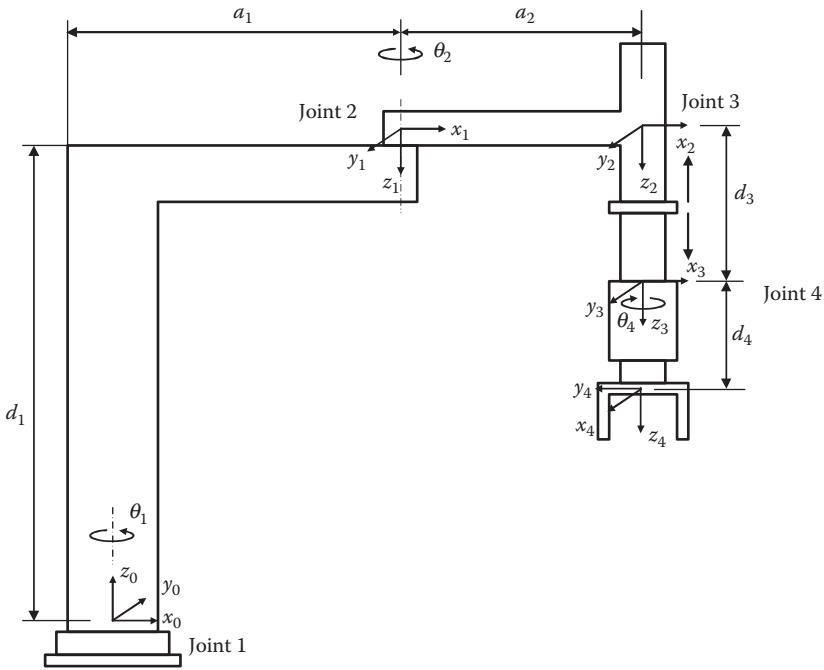


FIGURE 4.14 Four-link AdeptOne robot manipulator.

joints θ_1 and θ_2 are required to move the robot end-tip following the circular trajectory. The other two joint variables d_3 and θ_4 can be maintained at their initial values.

In the fuzzy inverse model construction, first, a fuzzy forward model is constructed to represent this MISO system. The inputs are the two joint angles θ_1 and θ_2 , and the output is the x -coordinate of the endpoint, where the endpoint follows the desired circular trajectory $(x - 43)^2 + (y - 43)^2 = 5^2$. The end-tip movement of this robot manipulator is modeled by 300 input-output data pairs as

$$R^i: \text{IF } \theta_1(k) = D_1^i \text{ AND } \theta_2(k) = D_2^i \text{ THEN } x(k) = c^i \quad (4.54)$$

TABLE 4.7
**Denavit–Hartenberg Kinematics Parameter
Matrix of a Four-Link Robot Manipulator**

Link	α_i (°)	a_i (mm)	d_i (mm)	θ_i (°)
1	180	l_1	d_1	θ_1
2	0	l_2	0	θ_2
3	0	0	d_3	0
4	0	0	d_4	θ_4

TABLE 4.8
Parameters for the Fuzzy Forward Model of a Four-Link Robot Manipulator

Rule Number	1	2	3	4	5	6
Center d_1^i (rad)	1.5567	1.5331	1.5213	1.4976	1.4742	1.4386
Center d_2^i (rad)	1.5439	1.4668	1.4409	1.3997	1.3662	1.3280
Position c^i (cm)	38.0960	39.0203	39.4831	40.4263	41.3882	42.8730

where the centers d_r^i of the input triangular MFs D_r^i and the position of the output singleton MF c^i are listed in Table 4.8.

The fuzzy inverse model is automatically constructed from the fuzzy forward model based on the proposed inverse method for each joint variable:

$$\begin{aligned} \text{IR}^i: \text{IF } x(k) = E^i &\text{ THEN } \theta_1(k) = f_1^i \\ \text{IF } x(k) = E^i &\text{ THEN } \theta_2(k) = f_2^i \end{aligned} \quad (4.55)$$

where

the center e^i of the input triangle MF E^i

the position of the output singleton MF f_r^i are listed in Table 4.9

The required system inputs θ_1 and θ_2 can be simultaneously obtained through fuzzy inferencing calculation as shown in Figure 4.15. The desired endpoint trajectory and the actual trajectory for the four-link robot arm are plotted in Figure 4.16, where the solid plot is the desired endpoint trajectory and the dashed plot is the actual endpoint trajectory. It is obvious that the actual endpoint trajectory follows the desired endpoint trajectory accurately. Figure 4.17 shows the error between the desired end-point trajectory and the actual value along x - and y -directions, respectively. The MSE of the proposed fuzzy inverse method over the simulation is calculated as listed in Table 4.10.

The performance of the proposed fuzzy inverse method is compared with the modular neural network approach (Alsina and Gehlot, 1995) for the four-link AdeptOne robot manipulator, where a three-layer sigmoid ADALINE neural module is assigned to each robot link to realize its own inverse kinematics. For example, the i th neural module is constructed for the i th link variable (angle or displacement) from the i th joint space to the $(i - 1)$ th joint space. Each neural module is designed to have

TABLE 4.9
Parameters for the Obtained Fuzzy Inverse Model of a Four-Link Robot Manipulator

Rule Number	1	2	3	4	5	6
Center e^i (cm)	38.0960	39.0203	39.4831	40.4263	41.3882	42.8730
Position f_1^i (rad)	1.5567	1.5331	1.5213	1.4976	1.4742	1.4386
Position f_2^i (rad)	1.5439	1.4668	1.4409	1.3997	1.3662	1.3280

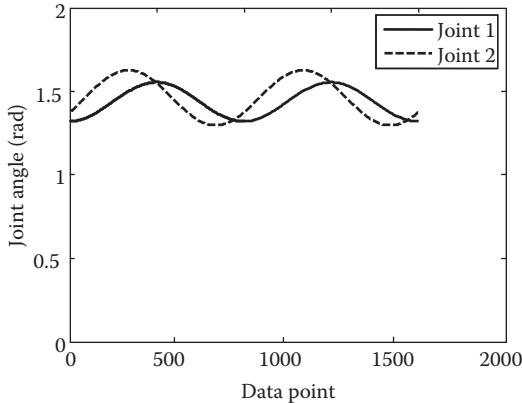


FIGURE 4.15 Required joint angles for the two joints θ_1 and θ_2 .

six neurons in the input layer, seven neurons in the hidden layer, and seven neurons in the output layer. The input is the i th link movement vector $\mathbf{X}_i = [x_i \ y_i \ z_i \ \phi_i \ \theta_i \ \psi_i]^T$, and the outputs are the $(i - 1)$ th link movement $\mathbf{X}_{i-1} = [x_{i-1} \ y_{i-1} \ z_{i-1} \ \phi_{i-1} \ \theta_{i-1} \ \psi_{i-1}]^T$ and the i th joint variable q_i (angle θ_i or displacement d_i). The four neural networks cascade each other and the global learning error is defined as $E = \sqrt{\sum_i (q_i^* - q_i)^2}$, and the learning coefficient is specified as $\mu = 0.3 + \frac{0.7}{\sqrt{3}}E$ in order to facilitate the error convergence. The backpropagation algorithm was used in the neural networks training process. The joint angles of the two joints θ_1 and θ_2 from the inverse kinematics calculation are shown in [Figure 4.18](#). The

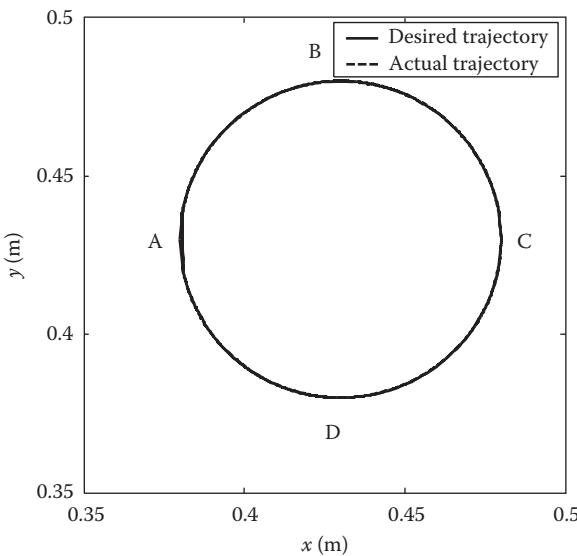


FIGURE 4.16 Desired endpoint trajectory and actual endpoint trajectory.

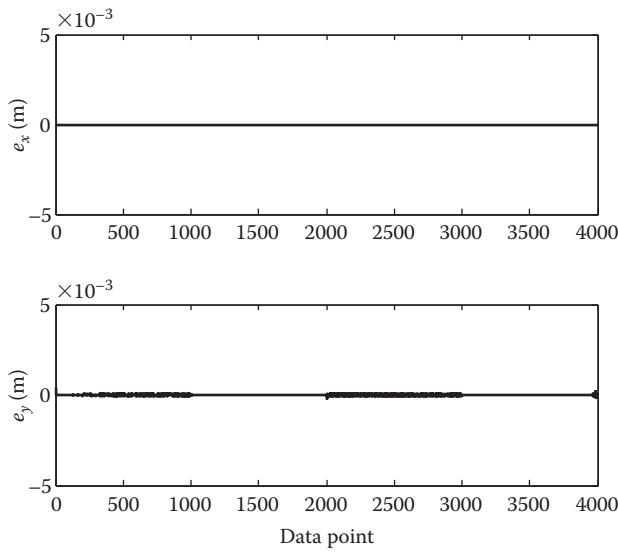


FIGURE 4.17 Error along x - and y -directions.

TABLE 4.10
MSE of the Proposed Fuzzy Inverse Method

	MSE (x-Direction)	MSE (y-Direction)
Fuzzy inverse model	2.4072×10^{-4}	6.5000×10^{-3}

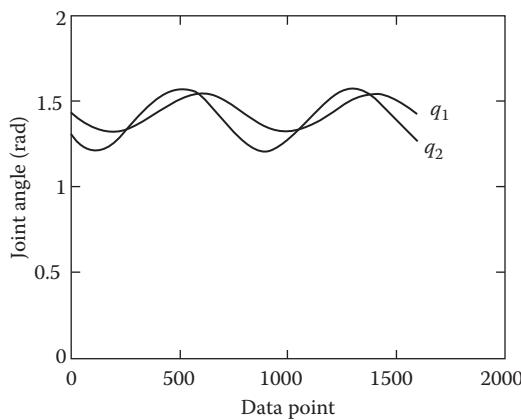


FIGURE 4.18 Required joint angles for the two joints θ_1 and θ_2 . (From Alsina, P.J. and Gehlot, N.S., *Proceedings of the 38th Midwest Symposium on Circuits and Systems*, IEEE, Piscataway, NJ, 1995. With permission.)

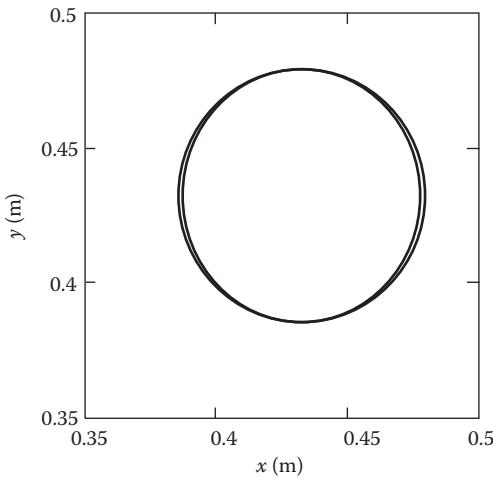


FIGURE 4.19 Desired endpoint trajectory and actual endpoint trajectory. (From Alsina, P.J. and Gehlot, N.S., *Proceedings of the 38th Midwest Symposium on Circuits and Systems*, 1995. With permission.)

desired endpoint circular trajectories and the actual endpoint circular trajectories are shown in Figure 4.19. Figures 4.18 and 4.19 are directly extracted from Alsina and Gehlot (1995). From visual comparison, it can be noticed that the proposed fuzzy inverse method shows better accuracy than the modular neural network approach.

It should also note that the precision and accuracy of the proposed fuzzy inverse method depends on the number of fuzzy rules in the original fuzzy forward model. With a larger number of fuzzy rules, the fuzzy forward model will model the system more accurately, and the inverse model will be more accurate as well.

4.3 CONCLUSION

This chapter presents a new method of deriving a fuzzy inverse model from a fuzzy forward model for a general MISO system. Unlike most existing fuzzy inverse methods, where only one input variable can be obtained and the remaining inputs are fixed as known values, this method can derive a fuzzy inverse model for all system input variables simultaneously in order to achieve the desired system output. The simulation examples demonstrate that the proposed fuzzy inverse models can calculate the inverse kinematics of complex industrial robot manipulators in a straightforward way without solving coupled nonlinear equations. This method is applicable to general MISO nonlinear systems in determining multiple system inputs to achieve the desired output value and is suitable for real-time control applications.

REFERENCES

- Alsina, P.J. and Gehlot, N.S., Robot inverse kinematics: A modular neural network approach, *Proceedings of the 38th Midwest Symposium on Circuits and Systems*, IEEE, Piscataway, NJ, Vol. 2, pp. 631–634, 1995.
- Babuška, R., *Fuzzy Modeling for Control*, Kluwer Academic Publishers, Dordrecht, Germany, 1998.
- Babuška, R., Sousa, J., and Verbruggen, H.B., Model-based design of fuzzy control systems, *Proceedings of the International Conference EUFIT'95*, Vol. 1, Aachen, Germany, pp. 837–841, 1995.
- Goel, P.K., The inverse kinematics solutions, closed-form dynamics and simulation of AdeptOne industrial robot, *IEEE International Conference on Robotics and Automation*, IEEE, New York, Vol. 3, pp. 1688–1693, 1988.
- Kucuk, S. and Bingul, Z., The inverse kinematics solutions of fundamental robot manipulators with offset wrist, *IEEE International Conference on Mechatronics*, IEEE, Piscataway, NJ, pp. 197–202, 2005.
- Manocha, D. and Canny, J.F., Efficient inverse kinematics for general 6R manipulators, *IEEE Transactions on Robotics and Automation*, 10(5): 648–657, 1994.
- Piegat, A., *Fuzzy Modeling and Control*, Physica-Verlag, New York, 2001.
- Xu, C. and Shin, Y.C., A fuzzy inverse model construction method for a MISO system, *IEEE Transaction on Fuzzy Systems*, in press, 2008.

5 Model-Based Optimization

Optimization of nonlinear systems has been an important issue for many problems across various disciplines. Many strategies based on gradient-based search techniques have been developed so far for optimization of nonlinear problems; however, in general, they have some drawbacks. First, the classical quantitative gradient-based methods are subject to numerical problems such as convergence to local minima or divergence when the problem to be optimized is highly nonlinear or ill-defined. Second, the accuracy of the optimization result will strongly depend on the fidelity of the models used, which often cannot warrant precise predictions due to the complexity and multimodality of the system or process. If a portion of the model is not precise, it will affect the entire optimization accuracy due to the inversion of Jacobian matrix. Third, only quantitative models can be used, thereby ruling out the possibility of using heuristic rules or empirical data, which, in some cases, are the only ways to describe the problem. Last, computing time for optimization is very long. Due to these reasons, there exists a need to develop an intelligent optimization scheme that is flexible such that different models can be easily incorporated into it and it is more efficient in optimal solution searching.

The approach we consider in this chapter is model-based optimization strategies based on soft computing techniques. After a knowledge base or a model base is established based on the intelligent modeling scheme discussed in the previous chapter, the next step is to perform inferencing to make decisions. The inferencing engine provides a reasoning strategy for searching the knowledge base to determine how to change the input conditions in order to provide appropriate decisions. Typically, the goal of inferencing in optimization is to determine the values of input (or design) parameters to minimize the given objective function within various constraints of both input and output variables. There are two types of inferencing mechanisms: forward chaining and backward chaining. Forward chaining begins with known facts and progresses with reasoning to determine whether the results are within acceptable regions. Backward chaining, on the other hand, begins with the prescribed output conditions and works backward to determine input conditions to satisfy the required consequent conditions.

In this chapter, a forward inferencing method is constructed based on evolutionary strategies. Due to their random, evolutionary nature of searching, evolutionary strategies provide the ability of finding global optimum points. Therefore, it is suitable for many optimization problems. In this chapter, the optimization strategies are presented along with some examples.

5.1 MODEL BUILDING

Consider a multivariable nonlinear system as shown in Figure 5.1. We assume that the knowledge exists in three different forms: analytical models, empirical data, and heuristic rules. The goal is to perform optimization utilizing all these forms of knowledge within a given set of constraints. As described in the previous chapter, empirical data can be used to construct an empirical model using either an artificial neural network (ANN) or a fuzzy basis function network (FBFN). Heuristic knowledge can be captured by an FBFN, as it can naturally incorporate if-then type rules. Since both ANN and FBFN can be constructed to model multi-input multi-output (MIMO) nonlinear relationships, the representations of knowledge can be carried out in compact form. For a known paradigm of ANN or FBFN, only the parameters of the networks need to be stored in the model base (Figure 5.2).

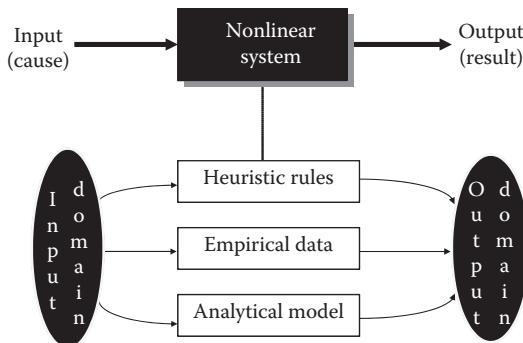


FIGURE 5.1 Heterogenous domains of knowledge representing the input–output relationships of a nonlinear system.

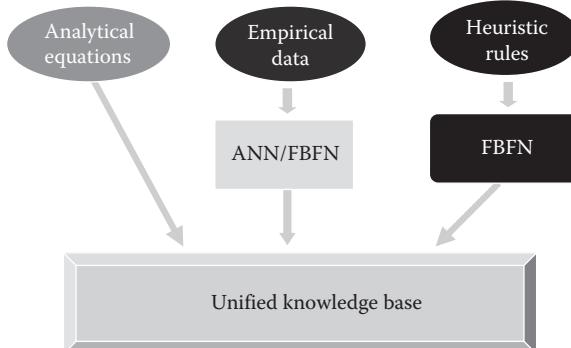


FIGURE 5.2 Building unified knowledge base from heterogeneous domains of knowledge.

5.2 MODEL-BASED FORWARD OPTIMIZATION

Various modeling schemes of nonlinear systems were described in Chapters 2 and 3. This chapter introduces an off-line optimization method as the next step after modeling. It is formulated as a constrained nonlinear optimization problem with mixed-discrete variables, and then the extended evolution strategy (ES) is proposed for the problem. The extended ES is first applied to popular numerical examples to assess its performance. In Section 5.4, the performance of the described model-based optimization scheme is demonstrated with two grinding optimization problems, which were chosen from the literature.

5.2.1 FORMULATION OF THE PROBLEM

The optimization strategy discussed in this chapter takes the form of model-based optimization proposed by Lee and Shin (2000), which requires system models for optimization. For the formulation of problem, the following variables are defined:

Input variables (\mathbf{x} , \mathbf{d}): Input values that one can directly manipulate

$$\mathbf{x} = (x_1, x_2, \dots, x_{n_x})^T \in \Re^{n_x} \text{ continuous variables}$$

$$\mathbf{d} = (d_1, d_2, \dots, d_{n_d})^T \in \mathbb{Z}^{n_d} \text{ integer (discrete) variables with valid metrics}$$

Output variables: Variables that one can directly measure or estimate

$$\mathbf{y} = (y_1, y_2, \dots, y_m)^T$$

It is assumed here that the relationships between input and output variables are determined by a priori modeling by neural network (NN) models, FBFN, or analytical models, if available, and can be represented by

$$y_j = y_j(\mathbf{x}, \mathbf{d}), \quad j = 1, 2, \dots, m \quad (5.1)$$

The optimization problem can be formulated as follows:

minimize

$$J = f[\mathbf{x}, \mathbf{d}; \mathbf{y}(\mathbf{x}, \mathbf{d})]$$

subject to

1. Inequality constraints regarding input variables

$$x_l(i) \leq x_i \leq x_u(i), \quad i = 1, \dots, n_x$$

$$d_j \in \{d_j(1), d_j(2), \dots, d_j(n_j)\}, \quad j = 1, \dots, n_d$$

2. Inequality constraints regarding output variables

$$g_i[\mathbf{y}(\mathbf{x}, \mathbf{d})] \geq 0, \quad i = 1, \dots, n_g$$

where n_g is the number of constraints regarding output variables.

It can be seen that the optimization problem that needs to be solved is a constrained nonlinear programming problem with mixed-discrete variables (Rao, 1996). For the sake of generality, it is assumed that the objective of cost function $f(\cdot)$ can be any function of input variables \mathbf{x} , \mathbf{d} and output variable \mathbf{y} . Therefore, this optimization can be applied to various problems.

Figure 5.3 shows the overall scheme of the model-based optimization described in this chapter. The training module trains FBFNs or RBFNs (radial basis function networks) in the model of hierarchical structure based on the input and output of the system to be optimized. In the case when analytical models or mathematical relationships are available, they will substitute the FBFNs or RBFNs. Whether the model is an FBFN or an analytic equation, it will be used to simulate system outputs for the input variable values specified by the optimization module. Hence, the performance of model-based optimization will be dictated by the effectiveness of both optimization and modeling modules.

Furthermore, this optimization formulation can be extended to the cases where some of the output variables (effects) are time-dependent functions of input variables (causes) if such relationships can be described by monotonically increasing or decreasing functions. In order to satisfy constraints for output variables, that is, certain output variables must not exceed the prescribed limits, the inequality constraints regarding output variables should be changed as follows:

$$g_i[\mathbf{y}(\mathbf{x}, \mathbf{d}, k)] \geq 0, \quad i = 1, \dots, n_g, \quad k = 1, \dots, N_d$$

where

k is discrete time interval

N_d is the total number of discrete steps

Incorporating the above constraints, however, results in an optimization problem that is too complex to solve even with the most advanced optimization algorithms. By considering the fact that output conditions are monotonously changing over time, the above constraints can be greatly simplified. For example, if the output variable

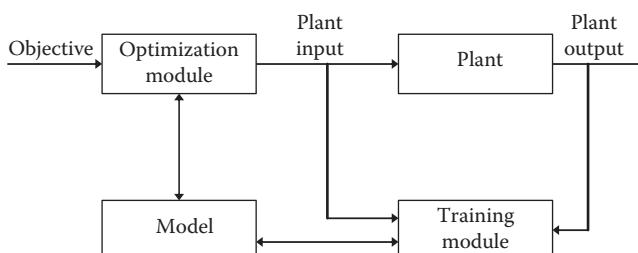


FIGURE 5.3 Overall architecture of optimization scheme.

P increases monotonously with the time, it can be shown that the following two constraints on the output variable P are equivalent:

$$\begin{aligned} P(k) &\leq P^*, \quad k = 1, \dots, N_d \\ \Leftrightarrow P(N_d) &\leq P^* \end{aligned}$$

5.2.2 OPTIMIZATION ALGORITHM

In order to provide robust solutions to the optimization problem described in Section 5.2.1, we introduce an extended evolution algorithm for a constrained optimization problem with mixed-discrete variables by combining the standard ES (Bäck, 1996) and the specialized ES for integer variables (Rudolph, 1994).

5.2.2.1 Standard Evolutionary Strategies for Continuous Variables

The original ES was developed for the following optimization problem with continuous variables:

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to } g_1(\mathbf{x}) \geq 0, \dots, g_r(\mathbf{x}) \geq 0 \end{aligned}$$

where $\mathbf{x} \in \Re^n$. Among several forms of ES, the simplest one is the single-membered ES, whose population at each generation consists of a single individual, which is the design vector \mathbf{x} . At generation $t=0$, the individual is usually initialized with a random number in the feasible region. In the single-membered ES, only the mutation operator exists to generate offspring and is accomplished by adding a random vector to the parent. The mutation operator enables the algorithm to perform stochastic search around the previous design vector in a controlled manner. Specifically, at each generation t , one offspring $\mathbf{x}'(t)$ is generated by adding a random vector to the parent $\mathbf{x}(t)$ as follows:

$$\mathbf{x}'(t) = \mathbf{x}(t) + \xi \cdot \mathbf{N}(0, 1) \quad (5.2)$$

where $\mathbf{N}(0, 1) \in \Re^n$ is the normally distributed random vector with expectation 0 and standard deviation 1. The standard deviation $\xi \in \Re_+$, which can be considered as a step size, is the only control parameter in the single-membered ES and accounts for the effectiveness of algorithm since a too large value of ξ will result in divergence of the search while a too small value will cause a stagnation problem. Hence, the following Rechenberg's 1/5 success rule has been commonly employed for the adjustment of step size ξ (Bäck, 1996):

The ratio of successful mutations to all mutation should be 1/5. If it is greater, increase the step size ξ , if it is less, decrease it.

where successful mutation indicates an instance when an offspring generated by mutation is better than the parent. After the mutation operator is applied, the parent

for the next generation is selected by simply choosing the better one between the two individuals:

$$\mathbf{x}(t+1) = \begin{cases} \mathbf{x}'(t) & \text{if } f[\mathbf{x}'(t)] \leq f[\mathbf{x}(t)] \text{ and } g_r[\mathbf{x}'(t)] \geq 0 \text{ for } \forall r \\ \mathbf{x}(t) & \text{otherwise} \end{cases} \quad (5.3)$$

The above process is repeated until a certain termination criterion is met. The above single-membered ES, no matter how simple it might look in its original form, showed promising results in numerical optimization problems (Schwefel, 1995). It was also shown that the solution of the single-membered ES converges to the global minimum as $t \rightarrow \infty$ for a general class of regular optimization problems whose global minimums are not isolated, and are greater than the negative infinity (Schwefel and Bäck, 1995).

The performance of ES can be further improved by introducing multimembers and other features into the single-membered ES (Bäck and Schütz, 1995). In (μ, λ) -ES adopted in this study, the μ individuals best out of λ offsprings are selected to form the parents of the next generation.

The multimembered ES also introduces recombination and self-adaptation of control parameters into the algorithm. The individual \mathbf{a} is used in this study to denote the member of parent population or offspring population. Each individual consists not only of the design variable vector \mathbf{x} , but also the control vector $\xi \in \Re_+^n$, which controls the step size of mutation. Hence each individual \mathbf{a} can be represented as $\mathbf{a} = (\mathbf{x}, \xi) \in I = \Re^n \times \Re_+^n$, where I is the population space. The control vector ξ also undergoes recombination and mutation to adapt to the environment. Figure 5.4 shows the outline of (μ, λ) -ES.

The recombination operator generates offspring population by combining the individuals in parent population, and provides a way of simulating inheritance and mating of nature. The recombination operator can be categorized into two branches, that is, the discrete recombination and the intermediate recombination. With the

```

 $t := 0;$ 
initialize the initial population  $\{\mathbf{a}_1(0), \dots, \mathbf{a}_\mu(0)\} \in I^\mu$ 
where  $I = \Re^n \times \Re_+^n$ 
and  $\mathbf{a}_i(t) = (\mathbf{x}_i(t), \sigma_i(t))$ 
 $\forall i \in \{1, \dots, \mu\};$ 
evaluate the initial population:  $\{f(\mathbf{x}_1(0)), \dots, f(\mathbf{x}_\mu(0))\}$ 
while stopping condition is not met, do
    recombine  $\mu$  parents to obtain  $\lambda$  offspring;
    mutate the  $\lambda$  offspring;
    evaluate the offspring:  $\{f(\mathbf{x}_1(t)), \dots, f(\mathbf{x}_\lambda(t))\};$ 
    select the best  $\mu$  individuals out of  $\lambda$  offspring;
     $t := t + 1;$ 
end

```

FIGURE 5.4 Standard (μ, λ) -ESs.

discrete recombination, for each variable of the offspring individual, the variable is copied from either the first or the second randomly chosen parent with probability of 1/2. On the other hand, the value is calculated as the average of the corresponding variables of the randomly chosen parent individuals with the intermediate recombination. These operators can also be applied in their global form where one randomly chosen parent is fixed and the second parent is randomly chosen anew for each single variable. Furthermore, different recombination operators are usually applied to the component vectors of an individual. For $\forall c \in \{x, \xi\}$ and $\forall i \in \{1, \dots, n\}$, the recombination operators used most frequently in the ES can be described as follows (Bäck, 1996):

$$c'_i = \begin{cases} c_{S,i} \text{ or } c_{T,i} & (\text{discrete}) \\ c_{S,i} \text{ or } c_{T,i} & (\text{global discrete}) \\ c_{S,i} + (c_{T,i} - c_{S,i})/2 & (\text{intermediate}) \\ c_{S,i} + (c_{T,i} - c_{S,i})/2 & (\text{global intermediate}) \end{cases}$$

where c'_i denotes the result of applying the recombination operator to c_i . The indices S and T denote two parents selected at random from the population (the index i in T_i indicates T to be sampled anew for each value of i). “or” denotes a decision with probability of 1/2. Here, we use discrete recombination operators for x , while using global intermediate operators for ξ , according to Bäck (1996) and Schwefel (1995).

The mutation operator provides a way of accomplishing stochastic search. The mutation operator operates on design vector \mathbf{x} and the corresponding standard deviation, ξ , and is performed by the following component-wise operations (Bäck, 1996):

$$\xi'_i = \xi_i \cdot \exp[\tau' \cdot N(0, 1) + \tau \cdot N_i(0, 1)] \quad (5.4)$$

$$x'_i = x_i + \xi'_i \cdot N_i(0, 1) \quad (5.5)$$

Here, $N(0, 1)$ denotes a normally distributed random variable with expectation 0 and standard deviation 1. The index i in $N_i(0, 1)$ indicates that the random variable is sampled anew for each possible value of i . It has been suggested that the following values be used for exogenous parameters τ and τ' (Bäck, 1996):

$$\tau = \frac{1}{\sqrt{2\sqrt{n_x}}}, \quad \tau' = \frac{1}{\sqrt{2n_x}} \quad (5.6)$$

5.2.2.2 Handling of Discrete Variables

The development of genetic algorithm (GA) for integer (discrete) variables has a longer history than other evolutionary algorithms, probably due to its inherent nature of representing variables in binary strings. The main focus of GA in dealing with integer (discrete) variables lies in the representation scheme (Lin and Hajela, 1992; Lin et al., 1995; Wu and Chow, 1995). Many researchers have used the same

mutation and reproduction operators as those for continuous variables once the integer variables are properly encoded.

Unlike GA, the main focus of evolutionary strategies on treating integer (discrete) variables lies on the development of a specialized mutation operator (Rudolph, 1994; Bäck and Schütz, 1995; Cai and Thierauf, 1997) for integer (discrete) variables. We can generalize a standard ES to include Rudolph's (1994) approach in handling integer variables due to its ability to handle variables with meaningful metrics and self-adaptability. Rudolph (1994) proposed a way of designing mutation distributions for integer variables based on the concept of maximum entropy. A discrete probability distribution was derived from a geometrical distribution for integer variables in which natural metrics are valid. The step size of the distribution was also self-adapted as in the case of continuous variables.

Since his original scheme can handle only integer variables not including discrete variables, we can modify it to handle the case of discrete variables and then combine it with the standard ES for continuous variables. It is assumed that discrete values of the j th discrete variable have been mapped to the integer values $\{1, 2, \dots, n_j\}$, where n_j is the number of discrete values the j th discrete variable can take, so that only the positive integer values can be found in the individual of the population. For example, suppose there are a set of five discrete values a discrete variable can take, then the following corresponding set of integer values can be defined:

$$\begin{array}{c} \{0.5, 3, 4.5, 9.2, 10.5\} \\ \Downarrow \\ \{1, 2, 3, 4, 5\} \end{array}$$

For the mutation of integer variables d_j 's, the following is suggested by Rudolph (1994) based on a geometrical distribution:

$$P\{z_j = k\} = \frac{p}{2-p}(1-p)^{|k|}, \quad k \in Z \quad (5.7)$$

$$d'_j = d_j + z_j, \quad j = 1, \dots, n_d \quad (5.8)$$

where

$P\{z_j = k\}$ is the probability for z_j to take the value of $k \in Z$
 d'_j is the mutated integer variable

The parameter p of geometrical distribution is determined by the mean step size s that controls the mutation:

$$s = n_d \cdot \frac{2(1-p)}{p(2-p)} \Leftrightarrow p = 1 - \frac{s/n_d}{[1 + (s/n_d)^2]^{1/2} + 1}$$

Rudolph (1994) proposed a way of self-adapting the mean step size s , similar to that just as with the control variables ξ_i 's for the continuous variables. The mean step size

undergoes the mutation by multiplication with a lognormally distributed random variable as follows:

$$s' = s \cdot \exp[\tau'' \cdot N(0, 1)] \quad (5.9)$$

where the exogenous parameter τ'' is given by

$$\tau'' = \frac{1}{\sqrt{n_d}} \quad (5.10)$$

Since a mean step size below 1 is meaningless for integer problems, the mean step size is set to 1 if the value is less than 1 after mutation.

The recombination operators of integer variables, on the other hand, are identical to those of the continuous variables. In the following, discrete recombination operators are used for \mathbf{d} , while global intermediate operators are used for s .

5.2.2.3 Handling of Constraints

The most popular and simple method to deal with constraints in the ES is to get rid of the offspring that violates the constraints. After an individual is generated by applying various genetic operators, the individual is tested against the constraints. If all the constraints are satisfied, the new individual is placed in the offspring pool for next operation. Otherwise, the individual is discarded and previous steps are repeated until an acceptable offspring is generated. Although this discarding method sometimes requires longer computational time due to the wasted steps in producing unacceptable individuals, it has the advantage of simplicity over other methods such as the penalty function method, which requires a judicious choice of penalty functions to ensure convergence to correct answers in many applications. Therefore, we adopt the discarding method mainly due to its simplicity of implementation.

5.2.2.4 Algorithm

The algorithm of extended ES is shown in [Figure 5.5](#). At generation $t = 0$, the parent population of μ individuals are generated randomly and their objective function values are evaluated. At each generation t , recombination and mutation operators are applied repeatedly until the number of offsprings that satisfy the constraints reaches λ . The offspring population of λ is evaluated by the objective function $f(\mathbf{x}, \mathbf{d})$ and the best μ individuals among λ offsprings are selected for the parent population of next generation. The modified ES is terminated when the stopping condition is met, which is given by the predefined maximum number of generations.

In the modified ES, an individual \mathbf{a} can be represented as

$$\mathbf{a} = (\mathbf{x}, \mathbf{d}, \xi, s) \in I = \Re^{n_x} \times \mathbb{Z}_+^{n_d} \times \Re^{n_x} \times \mathbb{Z}_+$$

where I is the population space. The control parameter space $\Re^{n_x} \times \mathbb{Z}_+$ contains n_x standard deviations $(\xi_1, \dots, \xi_{n_x})$ for the normally distributed mutation of continuous design variables x_i 's and a step size s for the geometric distribution of integer design

```

 $t := 0;$ 
initialize the initial population  $\{\mathbf{a}_1(0), \dots, \mathbf{a}_\mu(0)\} \in I^\mu$ 
where  $I = \mathbb{R}^{n_x} \times \mathbb{Z}^{n_d} \times \mathbb{R}_+^{n_x} \times \mathbb{Z}_+$ 
and  $\mathbf{a}_i(t) = (x_i(t), \mathbf{d}_i(t), \xi_i(t), s_i(t))$ 
 $\forall i \in \{1, \dots, \mu\};$ 
evaluate the initial population:  $\{f(\mathbf{x}_1(0), \mathbf{d}_1(0)), \dots, f(\mathbf{x}_\mu(0), \mathbf{d}_\mu(0))\}$ 
while stopping condition is not met, do
do until  $\lambda$  offsprings are obtained:
    recombine:
    mutate:
    check for constraints:
end
evaluate the offspring:  $\{f(\mathbf{x}_1(t), \mathbf{d}_1(t)), \dots, f(\mathbf{x}_\lambda(t), \mathbf{d}_\lambda(t))\}$ 
select the best  $\mu$  individuals out of  $\lambda$  offspring:
 $t := t + 1;$ 
end

```

FIGURE 5.5 Modified (μ, λ) -ESs.

variable d_j 's. While the continuous variables and integer variables coupled with their own control variables are embedded in an individual \mathbf{a} , they undergo the mutation and recombination separately at the same step. The continuous variables are recombedined and mutated following the procedures described in Section 5.2.2.1, while discrete variables are handled by the procedure described in Section 5.2.2.2.

When the ranges of continuous design variables are bounded, as in the case when machine limitation constraints are present, it has been found from simulation that normalization of continuous variables improves the performance of modified ES. When the design vectors are near the constraint boundary, the discarding mechanism of handling constraints tends to kill the individuals of large step sizes since the mutation of each step size of a design vector shares the same factor of $\exp[\tau' \cdot N(0, 1)]$ as shown in Equation 5.4. This can reduce the step size of the whole population inadvertently, thereby causing early stagnation of the algorithm. As a remedy, the range of each bounded continuous variable was linearly mapped to $[0, 1]$ before the ES was executed. Figure 5.6 shows the effect of normalization. It can be seen that the search space of the design variable x_2 before normalization is limited by the small search space of the design variable x_1 , which is close to constraint boundary A_1-A_2 , although the design variable x_2 is still far away from its constraint boundaries. By normalizing the ranges of design variables x_1 and x_2 , the design variable x_2 can have a larger search space, thereby increasing the probabilities for individual design vectors to reach the global optimum.

5.3 APPLICATION OF ES TO NUMERICAL EXAMPLES

Before applying the modified ES to the real optimization problems of grinding processes, its pure performance as an optimization tool is measured by applying it

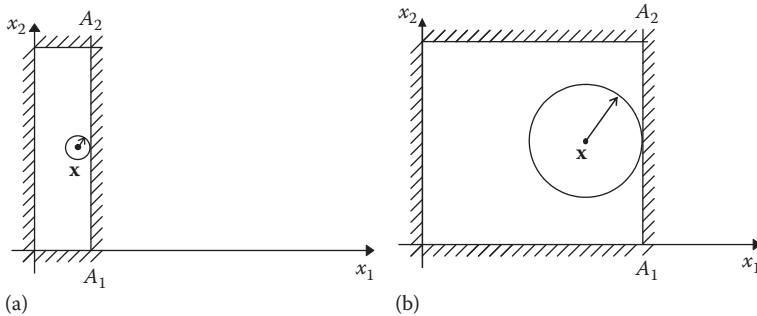


FIGURE 5.6 Effect of normalization on the search space of design vector: (a) step size of design vector \mathbf{x} before normalization, and (b) step size of design vector \mathbf{x} after normalization.

to conventional numerical examples. The following three examples are adopted from Wu and Chow (1995). For the implementation of the modified ES, the size of parent population μ and that of offspring population are set to 15 and 100, respectively, while the control parameters τ , τ' , and τ'' are chosen following Equations 5.6 and 5.10 with the constants of proportionality equal to 1.

Example 5.1 Design of a Gear Train

$$\text{Find } \mathbf{d} = \{d_1, d_2, d_3, d_4\}^T$$

$$\text{to minimize } f(\mathbf{d}) = \left(\frac{1}{6.931} - \frac{d_1 d_2}{d_3 d_4} \right)^2$$

$$\text{subject to } 12 \leq d_i \leq 60, \quad i = 1, \dots, 4$$

where $d_j, j = 1, \dots, 4$ is the number of teeth on the j th gear. The purpose of this problem is to achieve a gear ratio as close as possible to $1/6.931$, while the numbers of teeth are integer variables and have to be between 12 and 60.

Example 5.2 Design of a Pressure Vessel

$$\text{Find } (\mathbf{x}, \mathbf{d}) = \{x_1, x_2, d_1, d_2\}^T$$

$$\text{to minimize } f(\mathbf{x}, \mathbf{d}) = 0.6224d_1x_1x_2 + 1.7781d_2x_1^2 + 3.1611d_1^2x_2 + 19.84d_1^2x_1$$

$$\text{subject to } g_1(\mathbf{x}, \mathbf{d}) = d_1 - 0.0193x_1 \geq 0$$

$$g_2(\mathbf{x}, \mathbf{d}) = d_2 - 0.00954x_1 \geq 0$$

$$g_3(\mathbf{x}, \mathbf{d}) = \pi x_1^2 x_2 + \frac{4}{3} \pi x_1^3 - 750 \times 1728 \geq 0$$

$$g_4(\mathbf{x}, \mathbf{d}) = 240.0 - x_2 \geq 0$$

$$g_5(\mathbf{x}, \mathbf{d}) = d_1 - 1.1 \geq 0$$

$$g_6(\mathbf{x}, \mathbf{d}) = d_2 - 0.6 \geq 0$$

In this example, design variables x_1 and x_2 representing dimensions are continuous variables and the side constraints are $40 \leq x_1 \leq 80$ and $20 \leq x_2 \leq 60$, respectively. Design variables d_1 and d_2 are discrete values, integer multiples of 0.0625 in., in accordance with the available thickness of rolled steel plates. The purpose of this problem is to minimize the total manufacturing cost for the pressure vessel.

Example 5.3 Design of a Coil Compression Spring

$$\begin{aligned}
 & \text{Find } (\mathbf{x}, \mathbf{d}) = \{x_1, d_1, d_2\}^\top \\
 & \text{to minimize } f(\mathbf{x}, \mathbf{d}) = \pi^2 x_1 d_2^2 (d_1 + 2) / 4 \\
 & \text{subject to } g_1(\mathbf{x}, \mathbf{d}) = S - \frac{8C_f F_{\max} x_1}{\pi d_2^3} \geq 0 \\
 & \quad g_2(\mathbf{x}, \mathbf{d}) = l_{\max} - l_f \geq 0 \\
 & \quad g_3(\mathbf{x}, \mathbf{d}) = d_2 - d_{\min} \geq 0 \\
 & \quad g_4(\mathbf{x}, \mathbf{d}) = D_{\max} - x_1 - d_2 \geq 0 \\
 & \quad g_5(\mathbf{x}, \mathbf{d}) = \frac{x_1}{d_2} - 3.0 \geq 0 \\
 & \quad g_6(\mathbf{x}, \mathbf{d}) = \delta_{\text{pm}} - \delta_p \geq 0 \\
 & \quad g_7(\mathbf{x}, \mathbf{d}) = l_f - \delta_p - \frac{F_{\max} - F_p}{K} - 1.05(d_1 + 2)d_2 \geq 0 \\
 & \quad g_8(\mathbf{x}, \mathbf{d}) = -\delta_w - \frac{F_{\max} - F_p}{K} \geq 0
 \end{aligned}$$

where

$$\begin{aligned}
 C_f &= \frac{4(x_1/d_2) - 1}{4(x_1/d_2) - 4} - \frac{0.165d_2}{x_1} \\
 K &= \frac{Gd_2^4}{8d_1 x_1^3} \\
 l_f &= \frac{F_{\max}}{K} + 1.05(d_1 + 2)d_2 \\
 \delta_p &= \frac{F_p}{K}
 \end{aligned}$$

Design variable d_1 is an integer variable that denotes the number of spring coils and x_1 is a continuous variable that represents the winding diameter. Design variable d_2 is a discrete variable that can take 42 discrete values and denotes the wire diameter. The constraints are given in terms of the maximum working load F_{\max} (=1000.0 lb), the allowable maximum shear strength S (=189,000.0 psi), the maximum free length l_{\max} (=14.0 in.), the minimum wire diameter d_{\min} (=0.2 in.), the maximum outside diameter of spring D_{\max} (=3.0 in.), the preload compression force F_p (=300.0 lb), the allowable maximum deflection under preload δ_{pm} (=6.0 in.), the deflection from preload position to maximum load position δ_w (=1.25 in.). In the constraint inequalities, the parameter G denotes the shear modulus of the material whose value corresponds to 11.5×10^6 psi. Refer to Wu and Chow (1995) for the discrete values and the parameters used in this simulation. The objective of this problem is to minimize the volume of wire used to manufacture the spring.

TABLE 5.1
Optimum Solution of the Gear Train Design (Example 5.1)

Items (Number of Teeth)	Sandgren (1990)	Fu et al. (1991)	Meta-GA (Wu and Chow, 1995)	ES	Type of Variables
d_1	18	14	19	19	Integer
d_2	22	29	16	16	Integer
d_3	45	47	43	43	Integer
d_4	60	59	49	49	Integer
$f(\mathbf{d})$	5.7×10^{-6}	4.5×10^{-6}	2.7×10^{-12}	2.7×10^{-12}	

The optimization results from various optimization algorithms for the above three examples are listed in Tables 5.1 through 5.3. Sandgren (1990) employed a refined enumeration method coupled with either an exterior penalty function or a quadratic programming method to solve the above examples, while a generalized penalty function method that imposes a penalty on noninteger values was adopted by Fu et al. (1991). The results of meta-GA that employs another GA to optimize the GA parameters are also listed in Tables 5.1 through 5.3 (Wu and Chow, 1995). In the last example, optimization results of a conventional GA are also shown (Chen and Tsao, 1993). It can be seen that the modified ES achieved superior performance for all the three examples. In Example 5.1, the modified ES achieved an objective function value of 2.7×10^{-12} , which is as low as that of the meta-GA with Example 5.1, but better than those obtained from conventional optimization algorithms. With Examples 5.2 and 5.3, the modified ES outperformed all the conventional optimization algorithms and the meta-GA. Considering the fact that the meta-GA is one of the most advanced evolutionary algorithms, the performance of the modified ES is excellent.

TABLE 5.2
Optimum Solution of the Pressure Vessel Design (Example 5.2)

Items	Sandgren (1990)	Fu et al. (1991)	Meta-GA (Wu and Chow, 1995)	ES	Type of Variables
x_1 (in.)	48.97	48.3807	58.1978	58.2902	Continuous
x_2 (in.)	106.72	111.7449	44.2930	43.6927	Continuous
d_1 (in.)	1.125	1.125	1.125	1.125	Discrete
d_2 (in.)	0.625	0.625	0.625	0.625	Discrete
g_1	0.179	0.191250	0.001782	0.0000	
g_2	0.1578	0.163449	0.069793	0.06891	
g_3	3.0	75.8750	974.5829	3.0828	
g_4	133.284	128.2551	195.7070	196.3073	
$f(\mathbf{x}, \mathbf{d})$ (\$)	7982.5	8048.619	7207.497	7197.87	

TABLE 5.3
Optimum Solution of the Spring Design (Example 5.3)

Items	Sandgren (1990)	Chen and Tsao (1993)	Meta-GA (Wu and Chow, 1995)	ES	Type of Variables
x_1 (in.)	1.180701	1.2287	1.227411	1.10918	Continuous
d_1 (number of spring coils)	10	9	9	9	Integer
d_2 (in.)	0.283	0.283	0.283	0.263	Discrete
g_1 (psi)	54.309	415.969	550.993	3615.633	
g_2 (in.)	8.8187	8.9207	8.9264	9.176612	
g_3 (in.)	0.08298	0.08300	0.08300	0.063000	
g_4 (in.)	1.8193	1.7713	1.7726	1.627820	
g_5	1.1723	1.3417	1.3371	1.217414	
g_6 (in.)	5.4643	5.4568	5.4585	5.464279	
g_7 (in.)	0.0	0.0	0.0	0.0	
g_8 (in.)	0.0004	0.0174	0.0134	0.000017	
$f(\mathbf{x}, \mathbf{d})$ (in. ³)	2.7995	2.6709	2.6681	2.0823	

Another advantage of the ES over the meta-GA should be found in the computational efficiency and ease of use. The meta-GA tries to find an optimal set of control parameters of GA by using another main GA. Hence, each individual in the main GA requires a separate GA for the optimization problem. For example, if the main GA has a population size of 20, then 20 optimization problems need to be solved simultaneously. Considering the fact that those 20 optimization problems should be solved for each number of generation that is larger than 100 in general, the heavy computational load makes the application of meta-GA to a larger scale of problems almost prohibitive even with parallel computation schemes. On the other hand, the self-adaptability of ES enables one to obtain an optimal set of control parameters and design variables at the same time with just one run of the algorithm.

The computational efficiency of the modified ES can be assessed from its computation time. For the Sun E4504 machine, the actual CPU times taken for Examples 5.1 through 5.3 were 1.5, 12.1, and 4.7 s, respectively. Although direct comparison of computation time among four algorithms is not available since the results from three algorithms other than the ES are adopted from literature, it can be deduced from the above computation time of ES that the modified ES possesses competent computational efficiency.

Performance of the modified ES is also compared with that of another ES proposed by Cai and Thierauf (1997), who employed a parallel network of ES. For the example of designing a pressure vessel that is similar to Example 5.2 except different constraints, it has been found that both algorithms produce identical results. The advantage of the proposed ES can be found in the simplicity of the proposed ES over the complex parallel computing structure required by Cai and Thierauf's ES.

5.4 APPLICATION OF MODEL-BASED OPTIMIZATION SCHEME TO GRINDING PROCESSES

The model-based optimization scheme in Section 5.2 is applied to grinding optimization problems, which are chosen from literature (Wen et al., 1992; Liao and Chen, 1994). The first example shows a case when the proposed model scheme is applied to experimental data, while the second example uses analytic models. For the implementation of the ES, the size of parent population μ and that of offspring population are again set to 15 and 100, respectively, while the control parameters τ , τ' , and τ'' are chosen following Equations 5.6 and 5.10 with the constants of proportionality equal to 1.

5.4.1 APPLICATION TO CREEP FEED GRINDING EXAMPLE

In order to see how the developed scheme works with the real optimization problem of grinding processes, the scheme is first applied to Liao and Chen's (1994) example. They employed the backpropagation NN and gradient-descent algorithm to modeling and optimization of creep feed grinding. For this example, the grinding process is modeled by the FBFN using the extended adaptive least-squares (ALS) algorithm in Section 2.3.3 with the experimental data, and then the ES is applied to optimize the grinding process. The results are compared with those of Liao and Chen's scheme.

Liao and Chen used 32 sets of experimental data to model the creep feed grinding process. The five input variables were bond type B , mesh size m , concentration c that, when divided by four, gives the volumetric percentage of grit, workpiece speed v_w (in./min), and depth of cut d (10^{-3} in.). Surface roughness R_a ($\mu\text{in.}$), normal grinding force per unit width F_n (lb/in.), and grinding power per unit width P_w (hp/in.) were the three output variables. Since they performed two experiments for the same set of input variables, they used the average of the two sets of output variables to train the NN. Therefore, the actual number of data sets used for training was 16 instead of 32. Hence, the same set of training data was used for the learning of MIMO FBFNs. Table 5.4 lists the experimental data used for training.

An MIMO FBFN with five input and three output variables was constructed using the extended ALS algorithm (Lee and Shin, 2000). Table 5.5 shows the comparison between the trained FBFN using the extended ALS algorithm and the NN using the backpropagation algorithm. It can be seen that the MIMO FBFN trained by the extended ALS algorithm can model the grinding process more accurately with a smaller number of parameters than the backpropagation NN used by Liao and Chen (1994). This is a very important advantage of the new modeling scheme since an excessive number of parameters in network can always cause overfitting problems such as oscillations of the network output between the desired output of training data.

TABLE 5.4
Experimental Data Used for Training

Number	B	m	c ^a	v_w in./min (mm/s)	d, 10 ⁻³ in. (mm)	R _a , μ in. (μ m)	F _n , lb/in. (N/mm)	P _w , hp/in. (kW/m)
1	Resin	80	50	6.8 (2.9)	58 (1.47)	34.28 (0.870)	253.5 (44.35)	7.385 (216.5)
2	Resin	180	50	6.8 (2.9)	102 (2.59)	32.92 (0.835)	378.5 (66.25)	11.465 (336.5)
3	Resin	80	100	6.8 (2.9)	102 (2.59)	33.82 (0.87)	370 (64.75)	10.7 (314)
4	Resin	180	100	6.8 (2.9)	58 (1.47)	26.70 (0.675)	320 (56.0)	7.895 (231.5)
5	Resin	80	50	16.2 (6.9)	102 (2.59)	35.60 (0.905)	559.5 (97.95)	20.125 (591.5)
6	Resin	180	50	16.2 (6.9)	58 (1.47)	33.40 (0.85)	441 (77.2)	12.985 (381.5)
7	Resin	80	100	16.2 (6.9)	58 (1.47)	35.16 (0.89)	404 (70.7)	11.465 (336.5)
8	Resin	180	100	16.2 (6.9)	102 (2.59)	27.74 (0.705)	640 (112.05)	22.155 (650.5)
9	Vitrified	80	50	6.8 (2.9)	102 (2.59)	34.185 (0.87)	212 (37.15)	5.345 (157.5)
10	Vitrified	180	50	6.8 (2.9)	58 (1.47)	22.445 (0.57)	179.5 (31.45)	3.67 (108)
11	Vitrified	80	100	6.8 (2.9)	58 (1.47)	31.625 (0.80)	157 (27.5)	3.565 (105)
12	Vitrified	180	100	6.8 (2.9)	102 (2.59)	32.97 (0.835)	159 (27.85)	4.99 (146.5)
13	Vitrified	80	50	16.2 (6.9)	58 (1.47)	35.63 (0.905)	228.5 (40.0)	6.365 (186.5)
14	Vitrified	180	50	16.2 (6.9)	102 (2.59)	24.08 (0.61)	434 (75.95)	12.48 (366.5)
15	Vitrified	80	100	16.2 (6.9)	102 (2.59)	34.68 (0.885)	362.5 (63.45)	9.27 (272)
16	Vitrified	180	100	16.2 (6.9)	58 (1.47)	33.725 (0.855)	184 (32.25)	5.91 (173.5)

Source: Regenerated from Liao, C.W., and Chen, L.J., *Int. J. Mach. Tool. Manu.*, 34, 919, 1994.

^a 4 × volumetric %.

After the NN is trained using the 16 sets of experimental data, Liao and Chen (1994) applied their optimization methodology to the following problem:

minimize

$$f = -v_w d + 3R_a + 6F_n + 6P_w$$

subject to

$$6.8 \leq v_w \leq 16.2$$

TABLE 5.5
Comparison between the FBFN and the Backpropagation Algorithm

	Lee and Shin (2000)	Liao and Chen (1994)
Network structure	5 input–3 output-FBFN with 6 fuzzy rules	5-5-4-3 backpropagation NN
Number of parameters in the network	78	93
Sum of squared error	212.2	541.0

$$\begin{aligned}
58 &\leq d \leq 102 \\
B &\in \{\text{resinoid(1), vitrified(2)}\} \\
m &\in \{80, 100, 120, 140, 160, 180\} \\
c &\in \{50, 75, 100\} \\
R_a &\leq 33 \\
P_w &\leq 53.6
\end{aligned}$$

The objective of optimization is to maximize the metal removal rate given by $v_w d$ while minimizing the grinding force F_n and the grinding power P_w with the smallest surface roughness R_a obtainable, subject to various constraints. It can be seen that the above problem is a constrained nonlinear mixed-integer optimization problem with multiobjectives. Some of the objectives such as maximizing metal removal rate and minimizing surface roughness can be conflicting each other; however, those objectives are balanced by the weighting factors such as 1, 3, 6, and 6 for each objective. By adopting such objective functions, the optimization algorithm can find operating conditions that compromise among process objectives that conflict each other. Liao and Chen applied a gradient-descent algorithm with Boltzmann factor in order to avoid being trapped in a local minimum. However, it was not shown whether the input variable values obtained from numerical optimization would really achieve the optimal objective function values in experiment while satisfying all the constraints.

The modified ES has been applied to the same optimization problem using the FBFN model (Lee and Shin, 2000). The modified ES was run until 100 generations were evaluated. [Table 5.6](#) compares the results with those of Liao and Chen's work. It may first appear that Liao and Chen achieved a slightly lower objective function value of 197.73 than 205.27 from the FBFN with the modified ES. However, it should be noted that the same input obtained by Liao and Chen produces an objective function value of 705.27 when applied to the FBFN, which is much larger than the optimal objective function value from Liao and Chen's NN.

Considering the fact that only 16 sets of data pairs were used for the training of networks, it is not surprising to see discrepancies between the NN and the FBFN although they were trained by the same experimental data. As the optimization was performed based on two different models, the results are bound to be different. Since no experimental validation data are available, we cannot tell whose operating parameters will provide lower objective function values with the real process. However, more credits should be given to the FBFN with the modified ES since it achieved lower training errors with a smaller number of parameters in the network.

Furthermore, the following advantages of the proposed scheme over the Liao and Chen's methodology can be claimed:

1. Liao and Chen's gradient-descent technique requires derivatives of the objective function with respect to input variables. Hence, whenever the process objectives are changed, the derivatives need to be derived manually again and optimization routines should be changed. Since the ES does not require any form of derivatives, the ES will provide more flexibility and portability than the gradient-descent methods.

TABLE 5.6
Comparison between the Optimization Results

Source of Input and Output	Input Variables					Output Variables			Objective <i>f</i>
	<i>B</i>	<i>M</i>	<i>c</i> ^a	<i>v_w</i> , in./min (mm/s)	<i>d</i> , 10 ⁻³ in. (mm)	<i>R_a</i> , μ in. (μ m)	<i>F_n</i> , lb/in. (N/mm)	<i>P_w</i> , Hp/in. (kw/m)	
Liao and Chen's input and NN	Vitrified	180	50	16.2 (6.85)	63.17 (1.605)	31.72 (0.8057)	182 (31.87)	5.65 (166)	197.73
Liao and Chen's input and FBFN	Vitrified	180	50	16.2 (6.85)	63.17 (1.605)	29.91 (0.7597)	265.7 (46.52)	7.46 (219)	705.27
Lee and Shin's input and FBFN	Vitrified	180	75	10.5 (4.44)	58.00 (1.473)	32.76 (0.8321)	117.3 (20.54)	2.44 (71.6)	205.27

^a 4 × volumetric %.

2. Modeling scheme shows better performance than Liao and Chen's modeling scheme. Considering the fact that the training data are expensive to obtain from the grinding process, this also is an important advantage.
3. FBFN enables one to analyze the process in clear physical terms after training since they are based on linguistic fuzzy rules, while the back-propagation NN always remains as a black box. In addition, the FBFN scheme provides a way of incorporating heuristic knowledge from the experts into the model, while the NN does not have that capability.

It is not easy to assess the quality of solutions given by the optimization method since it was not evaluated experimentally. However, it seems very obvious that the optimization scheme proposed in this study can find a set of process input parameters to meet the demand given by the objective function and constraints. It should be noted that complex optimization problems such as the one given in this example simply cannot be solved by a rough guess given by human operators especially when they have multiobjectives and are constrained. For the purpose of reference, the experimental conditions in [Table 5.4](#) were tested against the optimization problem. It was found that only 7 sets of process conditions out of 16 satisfied the constraint requirements. Among the objective function values of the process conditions that satisfied all the constraints, the minimum value was 389.25 while the maximum value reached as high as 2403.8 (Lee and Shin, 2000).

5.4.2 APPLICATION TO SURFACE GRINDING EXAMPLE

Wen et al. (1992) developed an off-line optimization scheme for surface grinding processes. They adopted analytic models from various sources and formulated two nonlinear constrained optimization problems. The objective of the first problem is to maximize the production rate while minimizing the production cost in rough grinding, and the objective of the second problem is to obtain the finest surface finish while minimizing the production cost. The two optimization problems are constrained by the burning constraint, the wheel wear constraint, machine stiffness constraints, etc. They can be formulated in a formal way as follows:

1. Rough grinding

minimize

$$f(v_s, v_w, C, L) = 0.5 \frac{C_T}{C_T^*} - 0.5 \frac{\text{WRP}}{\text{WRP}^*}$$

subject to

burning constraint: $u \leq u^*$

wheel wear constraint: $\text{WRP}/\text{WWP} \geq G$

machine stiffness constraint: $\frac{1}{2K_c} \left(1 + \frac{v_w}{v_s G} \right) + \frac{1}{K_s} \geq \frac{|R_{\text{rem}}|}{K_m}$

surface finish constraint: $R_a \leq R_a^*$

2. Finish grinding

minimize

$$f(v_s, v_w, C, L) = 0.3 \frac{C_T}{C_T^*} + 0.7 \frac{R_a}{R_a^*}$$

subject to

burning constraint: $u \leq u^*$

wheel wear constraint: $\text{WRP}/\text{WWP} \geq G$

machine stiffness constraint: $\frac{1}{2K_c} \left(1 + \frac{v_w}{v_s G} \right) + \frac{1}{K_s} \geq \frac{|R_{em}|}{K_m}$

production rate constraint: $\text{WRP} \geq \text{WRP}^*$

where

C_T is the total production cost (\$/pc)

WRP is the workpiece removal parameter ($\text{mm}^3/\text{min N}$)

WWP is the wheel-wear parameter ($\text{mm}^3/\text{min N}$)

u is the specific grinding energy (J/mm^3)

G is the grinding ratio

K_c is the cutting stiffness (N/mm)

K_s is the wheel-wear stiffness (N/mm)

K_m is the static machine stiffness (N/mm)

R_{em} is the dynamic machine-characteristics

R_a is the arithmetic average value of surface roughness (μm)

C_T^* denotes the expected value of C_T

WRP^* , u^* , and R_a^* represent the constraint boundaries of WRP, u and R_a , respectively

It should be noted that the workpiece removal parameter instead of the workpiece removal rate was adopted by Wen et al. (1992) to represent the production rate since the workpiece removal parameter WRP is directly proportional to the volumetric removal rate Z_w (mm^3/min) for a constant depth of cut in the surface grinding process (King and Hahn, 1986). Wheel speed v_s (m/s), workpiece speed v_w (m/s), dressing depth C (mm), and dressing lead L (mm/rev) were chosen as the design variables and eight models based on nonlinear analytical equations were adopted for formulating objective functions and constraint boundaries. Refer to King and Hahn (1986) for detailed description of the various parameters used in this example. Wen et al. (1992) applied a successive quadratic programming algorithm to solve the above problems. The ES has been applied to the same problems with the following additional constraints on the design variables in order to ensure reasonable bounds in the design variables (Lee and Shin, 2000):

$$25 \leq v_s \leq 41.67$$

$$0.25 \leq v_w \leq 0.4167$$

TABLE 5.7
Comparison between the Optimization Results for Rough Grinding

Optimization Algorithm	Input Variables				Output Variables			Objective f
	v_s (m/s)	v_w (m/s)	C (mm)	L (mm/rev)	C_T (\$/pc)	WRP (mm ³ /min N)	R_a (μm)	
Successive quadratic programming	33.333	0.3327	0.0547	0.0440	6.203	17.467	1.744	-0.127
ES	41.67	0.25	0.0985	0.0709	7.016	27.958	1.800	-0.348

$$0.01 \leq C \leq 0.1$$

$$0.01 \leq L \leq 0.1$$

The results of the optimization are compared in Tables 5.7 and 5.8. The modified ES again showed excellent performance when compared to the successive quadratic programming method. In the example of rough grinding, it can be seen that the ES achieved the objective function value of -0.348, which is much lower than -0.127 of the successive quadratic programming. It is also interesting to see that the roughness is on the boundary of the constraint with the ES, while it is not with the successive quadratic programming. It can also be observed that v_s and v_w are on the boundary of design variable's constraints with the ES.

In the case of finish grinding, the objective is to obtain the finest surface roughness while minimizing the production cost. It can be seen that the modified ES achieved finer surface roughness with a lower production cost than the conventional technique. It should also be noted that v_s and v_w are on the boundary of design variable's constraints with the ES, while they are not with the successive quadratic programming.

The advantage of ES can be found in another aspect other than the lower objective function values achieved. The gradient-based algorithms such as the successive quadratic programming require a good initial starting point in the design

TABLE 5.8
Comparison between the Optimization Results for Finish Grinding

Optimization Algorithm	Input Variables				Output Variables			Objective f
	v_s (m/s)	v_w (m/s)	C (mm)	L (mm/rev)	C_T (\$/pc)	WRP (mm ³ /min N)	R_a (μm)	
Successive quadratic programming	33.333	0.3332	0.0515	0.0905	7.716	20.000	0.828	0.553
ES	41.67	0.4167	0.0100	0.0954	6.844	20.000	0.658	0.461

space. For example, Wen et al. (1992) suggested using the values recommended by experts for the initial values of optimization. It seems that the performance of the gradient-based algorithms largely depends on the choice of the initial point. This can be a severe drawback in optimization of grinding processes whose relationships between design variables and process outputs are highly nonlinear. Unless a good starting point is used, the algorithm is very likely to come up with suboptimal values instead of the global optimum point. In the ES, on the other hand, since the initial points are selected randomly, a judicious choice of initial point is not required to achieve optimal performance.

The computational efficiency of the modified ES was also excellent. When run on a Sun E4504 server machine, it took 4.6 s for the modified ES to produce a solution.

The model-based optimization described in this chapter is not process-specific, that is, it is not confined to any specific objective function or process. Hence, the described modified ES scheme offers applicability to a broad range of other complex manufacturing processes such as milling processes, turning processes, or injection-molding processes.

REFERENCES

- Bäck, T., *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, Reading, NY, 1996.
- Bäck, T. and Schütz, M., Evolution strategies for mixed-integer optimization of optical multilayer systems, *Evolutionary Programming IV: Proceedings of the 4th Annual Conference on Evolutionary Programming*, San Diego, CA, pp. 33–51, 1995.
- Cai, J. and Thierauf, G., Evolution strategies in engineering optimization, *Engineering Optimization*, 29: 177–199, 1997.
- Chen, J.L. and Tsao, Y.C., Optimal design of machine elements using genetic algorithms, *Journal of the Chinese Society of Mechanical Engineers*, 14(2): 193–199, 1993.
- Fu, J.F., Fenton, R.G., and Cleghorn, W.L., A mixed integer-discrete-continuous programming method and its application to engineering design optimization, *Engineering Optimization*, 17: 263–280, 1991.
- King, R.I. and Hahn, R.S., *Modern Grinding Technology*, Chapman and Hall, Reading, NY, 1986.
- Lee, C.W. and Shin, Y.C., Evolutionary modeling and optimization of grinding processes, *International Journal of Production Research*, 38(12): 2787–2813, 2000.
- Lee, C.W., Choi, T.J., and Shin, Y.C., Intelligent model-based optimization of the surface grinding process for heat-treated 4140 steel alloys with aluminum oxide grinding wheels, *Transactions of the ASME, Journal of Manufacturing Science and Engineering*, 125: 65–76, February 2003.
- Liao, T.W. and Chen, L.J., A neural network approach for grinding processes: Modeling and optimization, *International Journal of Machine Tools and Manufacture*, 34(7): 919–937, 1994.
- Lin, C.-Y. and Hajela, P., Genetic algorithms in optimization problems with discrete and integer design variables, *Engineering Optimization*, 19: 309–327, 1992.
- Lin, S.-S., Zhang, C., and Wang, H.-P., On mixed-discrete nonlinear optimization problems: A comparative study, *Engineering Optimization*, 23: 287–300, 1995.
- Rao, S.S., *Engineering Optimization: Theory and Practice*, 3rd edn., John Wiley & Sons, Reading, NY, 1996.

- Rudolph, G., An evolutionary algorithm for integer programming, *Parallel Problem Solving from Nature—PPSN III: International Conference on Evolutionary Computation*, Lecture Notes in Computer Science, Vol. 866, Springer-Verlag, London, United Kingdom, pp. 139–148, 1994.
- Sandgren, E., Nonlinear integer and discrete programming in mechanical design optimization, *Transactions of the ASME, Journal of Mechanical Design*, 112: 223–229, 1990.
- Schwefel, H.-P., *Evolution and Optimum Seeking*, Wiley, Reading, NY, 1995.
- Schwefel, H.-P. and Bäck, T., Evolution strategies II: Theoretical aspects, *Genetic Algorithms in Engineering and Computer Science*, John Wiley & Sons, New York, 1995.
- Wen, X.M., Tay, A.A.O., and Nee, A.Y.C., Micro-computer-based optimization of the surface grinding process, *Journal of Materials Processing Technology*, 29: 75–90, 1992.
- Wu, S.-J. and Chow, P.-T., Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization, *Engineering Optimization*, 24: 137–159, 1995.

6 Neural Control

Neural control is an attractive tool when the model of the plant to be controlled is not well known and input–output data are available in the form of measurements. In such cases, neural networks (NNs) can be used to approximate the plant dynamics as a black box model and a suitable control law can be designed to regulate the system response. Both static and dynamic NNs have been used for controller design and their general approximation and learning abilities make NNs an attractive tool when a mathematical model of the plant is not known. In this chapter, we review several of the commonly used neural control techniques.

6.1 SUPERVISED CONTROL

Artificial neural networks (ANNs) possess general approximation capabilities so that they can be used to mimic complex input–output relationships. While it is often difficult to capture the human’s control action based on sensory input by a mathematical model, ANNs can be trained to mimic human action based on measurement data along with desired response as input and can be used as a nonlinear controller to replace the human worker. This is schematically shown in [Figure 6.1](#). The neural controller can be first trained by the human operator’s action. During this training, the human operator must generate his actions solely based on the sensor input. Once it is fully trained, then the dynamics of the plant and the time response of the operator can be emulated by the neural controller.

6.2 DIRECT INVERSE CONTROL

If the inverse relationship can be established between the input and output of a nonlinear system, then it would be possible to construct a controller directly based on the inverse model such that the control system follows the command. ANN can be used to learn the inverse relationship of a nonlinear system, either by off-line training or online training, then it would be relatively straightforward to construct a controller. [Figure 6.2](#) shows a schematic of an inverse controller.

An approximate inverse model of the plant by NNs can be obtained as long as the plant dynamics is invertible. Consider, for example, the time-invariant nonlinear system given by

$$y(k) = f(y(k-1), \dots, y(k-n), u(k-1), u(k-2), \dots, u(k-m)) \quad (6.1)$$

Then, the goal is to obtain an inverse model such that

$$u(k) = \phi(y(k+1), y(k), \dots, y(k-n+1), u(k-1), \dots, u(k-m+1)) \quad (6.2)$$

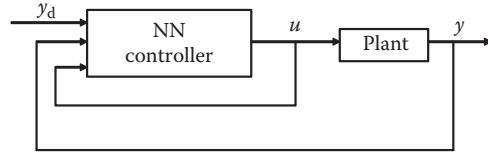


FIGURE 6.1 Supervised control.

Once the NN is trained, then $y(k+1)$ can be replaced by $r(k+1)$. In order to make this inverse controller effective, the training data must cover a sufficiently large domain of input–output mapping space and the command signal must be limited to the output domain space used during training.

The inverse controller can also be designed based on online learning schemes. The indirect learning scheme provides a means of generating feedforward control law with the inverse model of the plant during operation. The error backpropagation algorithm discussed in Sections 3.3 and 3.4 can be used for the training of the inverse model. A schematic of this learning scheme is shown in Figure 6.3. If a correct inverse model is established, then the output of the overall system can track input commands closely. The training is carried out to minimize the error between u and v . However, the feedforward neural controller designed by this scheme has a tendency of converging to a single value of u when the input and output values of the plant are used for NN training. Therefore, this scheme alone may not lead to satisfactory controller design since the NN model might not represent the true inverse model of the plant.

Another way of designing a feedforward neural controller is by direct learning of an inverse model as shown in Figure 6.4. In this case, an adaptive learning scheme needs to be employed to train the neural inverse controller such that the error between the command and the output becomes minimized and hence can be used for time varying system. This scheme, however, requires that the Jacobian of the plant be known. In addition, the stability of the system cannot be guaranteed. The total error $e = y_d - y$ can be propagated back through the plant using the partial derivatives of the plant at its operating point:

$$\Delta w_{ji}^{(k)} = \mu \delta_j u_i^{(k)} \quad (6.3)$$

$$\delta_j^{(k-1)} = f'(\text{net}) \sum_i \delta_i^p \frac{\partial P_i}{\partial u_j} \quad (6.4)$$

$$\delta_i^p = y_{d_i} - y_i \quad (6.5)$$

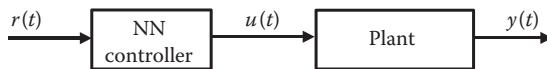


FIGURE 6.2 Inverse NN controller.

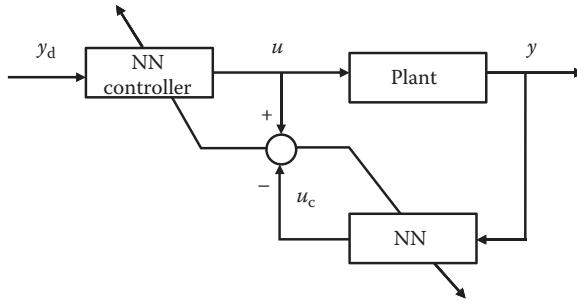


FIGURE 6.3 Indirect learning scheme.

where $P_i(\mathbf{u})$ denotes the i th element of the plant output for plant input \mathbf{u} . Since the plant is unknown, its partial derivatives (Jacobian) can be determined by

$$\frac{\partial P_i}{\partial u_j} \approx \frac{P_i(\mathbf{u} + \delta u_j) - P_i(\mathbf{u})}{\delta u_j} \quad (6.6)$$

The approximate derivative can be determined by varying each input slightly and measuring corresponding output change at each operating point (Psaltis et al., 1988).

In general, this type of inverse controllers has some drawbacks:

- They lack the capability of rejecting disturbances.
- Learning may not be efficient because the network must learn the responses of the plant over a wide range of operating conditions.
- Input must be determined a priori. In other words, its effectiveness is limited to the input range used for training.

For this reason, a feedback controller is often used in conjunction with the inverse neural model as shown in [Figure 6.5](#). This will reduce the sensitivity of the entire control system against imperfect modeling and provide some disturbance rejection capability.

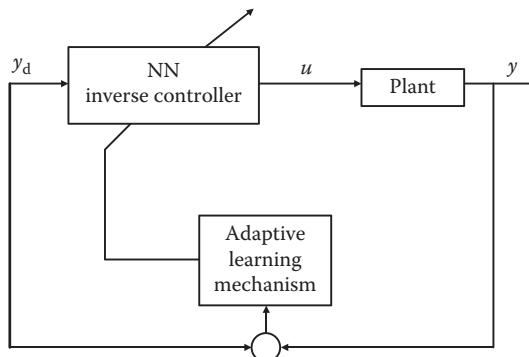


FIGURE 6.4 Direct learning of inverse model.

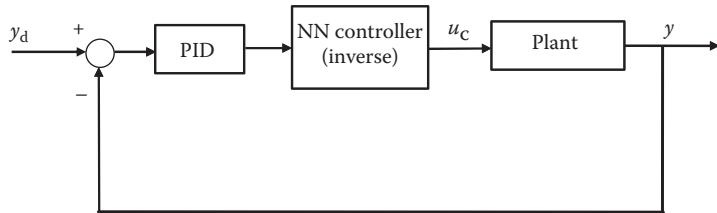


FIGURE 6.5 Variant of direct inverse control.

6.3 MODEL REFERENCE ADAPTIVE CONTROL

In model reference adaptive control, the goal is to make the control system behave like the reference model, which specifies the desired response of the system. The controller parameters are adjusted so as to minimize the error between the model and the actual system outputs. When the system is highly nonlinear, then linear model reference adaptive controllers might not produce satisfactory performance. In such cases, similar to the model reference adaptive control for linear systems, NNs can be used to design model reference adaptive control for nonlinear systems.

Feasibility of adapting ANNs to adaptive control of nonlinear systems has been actively explored. There are two different approaches to the model reference adaptive controller design. Figure 6.6 shows a direct model reference adaptive control (MRAC). In the direct MRAC, the parameters of the controllers can be adjusted in a similar way as in the direct control. Figure 6.7 shows the indirect MRAC. In this case, the unknown plant dynamics is modeled by an NN, which is subsequently incorporated into the control law synthesis. While both direct and indirect approaches have been used for linear systems only the indirect method with NNs has been implemented so far (Narendra and Parthasarathy, 1990; Narendra, 1992; Narendra and Mukhopadhyay, 1994).

Let us consider a system described by the following dynamics:

$$y_p(k+1) = f(y_p(k), y_p(k-1), \dots, y_p(k-n+1)) + u(k) \quad (6.7)$$

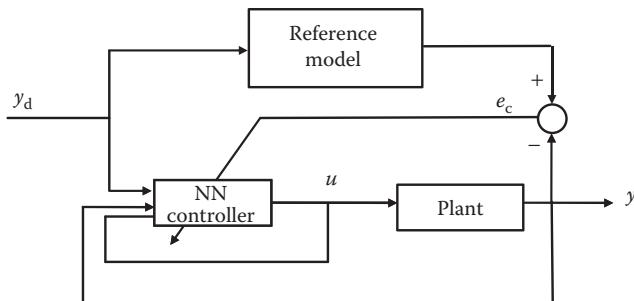


FIGURE 6.6 Direct model reference neural adaptive control.

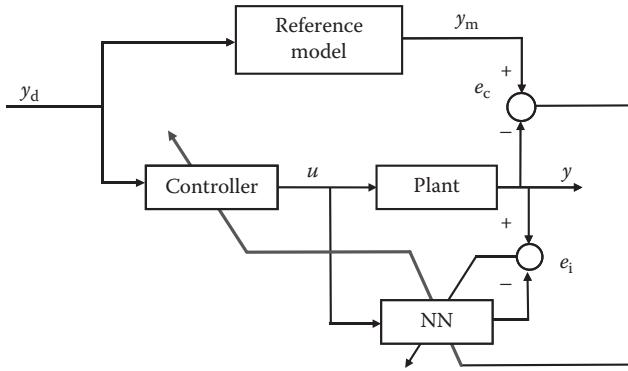


FIGURE 6.7 Indirect model reference neural adaptive control.

This system dynamics can be approximated by an NN as follows:

$$\hat{y}_p(k+1) = \phi(y_p(k), y_p(k-1), \dots, y_p(k-n+1)) + u(k) \quad (6.8)$$

Assume that the reference model is described by

$$y_m(k+1) = a_1 y_m(k) + a_2 y_m(k-1) + \dots + a_l y_m(k-l+1) + r(k) \quad (6.9)$$

If the output error $e_c(k)$ is defined as $e_c(k) = y_p(k) - y_m(k)$, then the goal of the control is to determine the bounded input $u(k)$ such that $\lim_{k \rightarrow \infty} e_c(k) \simeq 0$.

If the control input can be computed at each instant k as follows:

$$\begin{aligned} u(k) = & -\phi(y_p(k), y_p(k-1), \dots, y_p(k-n+1)) + a_1 y_m(k) \\ & + a_2 y_m(k-1) + \dots + a_l y_m(k-l+1) + r(k) \end{aligned} \quad (6.10)$$

then the resulting dynamics becomes

$$\begin{aligned} y_p(k+1) = & f(\bullet) - \phi(\bullet) + a_1 y_m(k) + a_2 y_m(k-1) \\ & + \dots + a_l y_m(k-l+1) + r(k) \end{aligned} \quad (6.11)$$

6.4 INTERNAL MODEL CONTROL

In internal model control (IMC), plant dynamics is modeled by an NN model and a feedback similar to conventional feedback controllers, except that error between y and \hat{y} , is used (Figure 6.8) (Carlos and Morari, 1982). In conventional IMC, the controller can be designed based on the internal model to improve the robustness to disturbance. In NN-based IMC, NN can also be used for the controller based on the forward NN model being trained during operation.

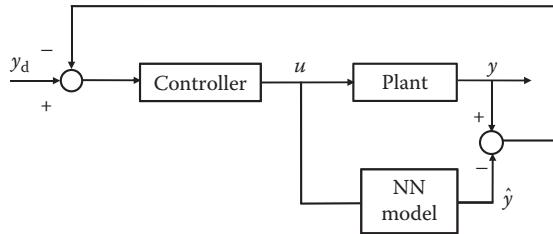


FIGURE 6.8 Internal model control.

6.5 MODEL PREDICTIVE CONTROL

Model predictive control has been widely used in industry and there are a wide variety of algorithms. Model predictive control utilizes a separately identifiable model and uses multiple-step predictions. For nonlinear systems, NN models can be obtained by forward modeling and used as the internal model as shown in Figure 6.9. This structure was proposed by Psichogios and Ungar (1991) and Saint-Donat et al. (1991) with sequential quadratic programming for the optimization. Other work involving NNs in predictive control includes Hunt et al. (1992), Liu et al. (1998), and Liu and Daley (2001). In most of these algorithms, nonlinear programming techniques are used to minimize the performance index based on future predicted values.

In this section, an approach based on affine nonlinear predictors using NNs that is easier for online implementation is briefly discussed. Consider discrete-time affine nonlinear control systems whose input–output relationship can be described by

$$y_t = F(\mathbf{y}_t) + G(\mathbf{y}_t)u_{t-d}$$

where

$F(\cdot)$ and $G(\cdot)$ are nonlinear functions

y is the output

u is the control input

d is the embedded delay of the system

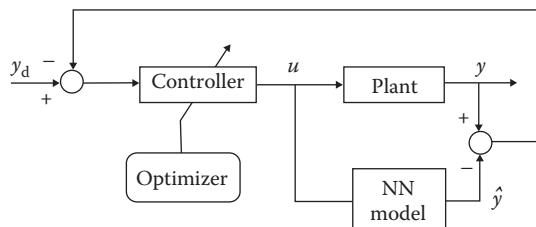


FIGURE 6.9 Model predictive control with NNs.

The vector \mathbf{y}_t is defined by $\mathbf{y}_t = [y_{t-1}, y_{t-2}, \dots, y_{t-n}]$. For such systems, the following $i+d$ step ahead of nonrecursive nonlinear predictor can be employed (Liu et al. 1998), $y_{t+d+i} = H(y_{t-1}, y_{t-2}, \dots, y_{t-n+1}, u_{t+i}, u_{t+i-1}, \dots, u_{t-d})$

$$\hat{y}_{t+d+i} = \hat{F}_i(\mathbf{x}_t) + \sum_{j=0}^i \hat{G}_{ij}(\mathbf{x}_t)u_{t+j}$$

For $i = 0, 1, \dots, L$, where $\hat{F}_i(\mathbf{x}_t)$ and $\hat{G}_{ij}(\mathbf{x}_t)$ are nonlinear functions of the vector $\mathbf{x}_t = [y_t, y_{t-1}, y_{t-2}, \dots, y_{t-n+1}, u_{t-1}, u_{t-2}, \dots, u_{t-d}]$

Neural networks can be used to approximate both $\hat{F}_i(x_t)$ and $\hat{G}_{ij}(x_t)$ as follows, using radial basis function networks:

$$\begin{aligned}\hat{F}_i(\mathbf{x}_t) &= \sum_{k=1}^{N_i} f_{i,k} \phi_{i,k}(\mathbf{x}_t) \\ \hat{G}_{ij}(\mathbf{x}_t) &= \sum_{k=1}^{N_{ij}} g_{ij,k} \phi_{ij,k}(\mathbf{x}_t)\end{aligned}$$

For $j < i$ and i, j, \dots, L , and N_i and N_{ij} represent the size of networks.

Predictive control based on the NNs can be carried out by minimizing a cost function given by

$$J_p = \frac{1}{2} \|R_{t+d+L} - Y_{t+d+L}\| + \frac{1}{2} \alpha \|\Delta U_{t+L}\|$$

where

$$\begin{aligned}R_{t+d+L} &= [r_{t+d} \quad r_{t+d+1} \quad \cdots \quad r_{t+d+L}] \\ Y_{t+d+L} &= [\hat{y}_{t+d} \quad \hat{y}_{t+d+1} \quad \cdots \quad \hat{y}_{t+d+L}] \\ U_{t+L} &= [u_t \quad u_{t+1} \quad \cdots \quad u_{t+L}]\end{aligned}$$

6.6 FEEDFORWARD CONTROL

Feedforward control has been used for improving tracking performance of control systems. Consider a linear system with command feedforward control as shown in Figure 6.10 in discrete frequency domain. In tracking control, the goal is $y_n = r_n$. If a

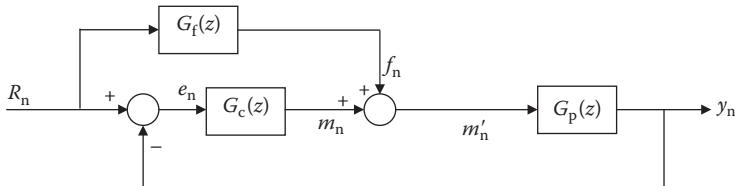


FIGURE 6.10 Feedforward control.

feedforward controller G_f is added as shown in Figure 6.10, then we can obtain the output as follows:

$$m'_n = G_f(z)r_n + G_c(z)\underbrace{(r_n - y_n)}_{e_n}$$

$$y_n = G_p(z)m'_n$$

$$\frac{y_n}{G_p(z)} = [G_f(z) + G_c(z)]r_n - G_c(z) \cdot y_n$$

$$[1 + G_p G_c(z)]y_n = G_p(z)[G_f(z) + G_c(z)]r_n$$

$$\frac{y_n}{r_n} = \left[1 + \frac{G_f(z)}{G_c(z)}\right] \frac{G_c(z)G_p(z)}{1 + G_c(z)G_p(z)}$$

If we choose the feedforward controller as

$$G_f(z) = \frac{1}{G_p(z)} = G_p(z)^{-1}$$

then perfect tracking can be achieved as follows:

$$\frac{y_n}{r_n} = \left[1 + \frac{1}{G_p(z)G_c(z)}\right] \frac{G_c(z)G_p(z)}{1 + G_c(z)G_p(z)} = 1$$

For unknown nonlinear systems, an NN model can be obtained for the inverse dynamics of the plant and can be added to the feedback control system as shown in Figure 6.11.

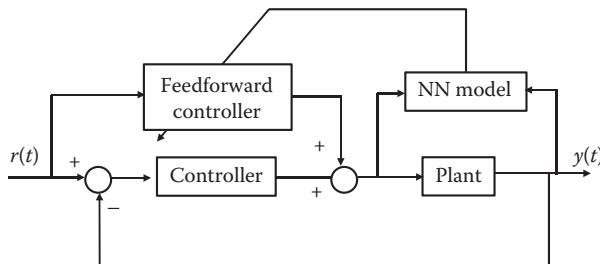


FIGURE 6.11 Feedforward control with NNs.

REFERENCES

- Carlos, E.G. and Morari, M., Internal model control 1: A unifying review and some new results. *Industrial and Engineering Chemistry Process Design and Development*, 21: 308–323, 1982.
- Hunt, K.J., Sbararo, D., Zbikowski, R., and Gawthrop, P.J., Neural networks for control systems—A survey, *Automatica*, 28(6): 1083–1112, 1992.
- Liu, G.P. and Daley, S., Adaptive predictive control of combustor NO_x emissions, *Control Engineering Practice*, 9: 631–638, 2001.
- Liu, G.P., Kadirkamanathan, V., and Billings, S.A., Predictive control for non-linear systems using neural networks, *International Journal of Control*, 71(6): 1119–1132, 1998.
- Narendra, K.S., Adaptive control of dynamical systems using neural networks, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, White, D.A. and Sofge, D.A. (Eds.), Van Nostrand-Reinhold, New York, pp. 141–183, 1992.
- Narendra, K.S. and Mukhopadhyay, S., Adaptive control of multivariable systems using neural networks, *Neural Networks*, 7: 737–752, 1994.
- Narendra, K.S. and Parthasarathy, K., Identification and control of dynamical systems using neural networks, *IEEE Transaction on Neural Networks*, 1: 4–27, 1990.
- Psaltis, D., Sideris, A., and Yamamura, A.A., A multilayer neural network controller, *IEEE Control Systems Magazine*, 9: 17–25, 1988.
- Psichogios, D.C. and Ungar, L.H., Direct and indirect model based control using artificial neural networks, *Industrial and Engineering Chemical Research*, 30: 2564–2573, 1991.
- Saint-Donat, J., Bhat, N., and McAvoy, T.J., Neural net model predictive control, *International Journal of Control*, 54: 1453–1468, 1991.

7 Fuzzy Control

The fuzzy control technique was first initiated by the pioneering research of Mamdani (Mamdani, 1974, 1977; Mamdani and Assilian, 1975) for systems structurally difficult to model, which was motivated by Zadeh's paper on system analysis with fuzzy sets theory (Zadeh, 1965, 1968, 1973, 1975). Since then, fuzzy control theory has become one of the most extensively studied areas in both academia and industry. Many theoretical developments and practical applications have been reported. It is an appealing alternative to conventional control methods since it provides a systematic and efficient framework to deal with uncertainties and nonlinearities in complex systems, when an accurate system analytical model is not available, not possible to obtain, or too complicated to use for control purposes. In this chapter, different fuzzy control schemes are explained in each section with individual example to give readers an easy understanding of the fuzzy controllers' working principle.

7.1 KNOWLEDGE-BASED FUZZY CONTROL

The knowledge-based fuzzy control is typically used when the explicit system analytical model is not available. It is intuitive to understand and easy to design for engineers who are unfamiliar with control theory. When the controlled system follows some general operating characteristics, a fuzzy controller can be constructed based on the knowledge of the controlled system's behavior and human operator's experience. This method enables one to capture the system dynamics qualitatively and is simple to execute for real-time applications, which has been demonstrated in various research literatures and commercial products. In this type of fuzzy control design, each fuzzy control rule represents one local system behavior corresponding to certain control input values and then the global fuzzy controller is built up based on the aggregation of individual local fuzzy rules.

7.1.1 FUZZY PID CONTROL

The simplest version of fuzzy controller is the fuzzy proportional-integral-derivative (PID) controller. It can be viewed as a heuristic control approach, which is designed without any explicit system analytical model. It has been applied extensively to various industrial problems due to its simple structure and robust performance over a wide range of operating conditions. The control action depends on the error $e(t)$, the change of the error $c(t)$, and the integral of the error $b(t)$ as

$$e(t) = y_d(t) - y(t) \quad (7.1)$$

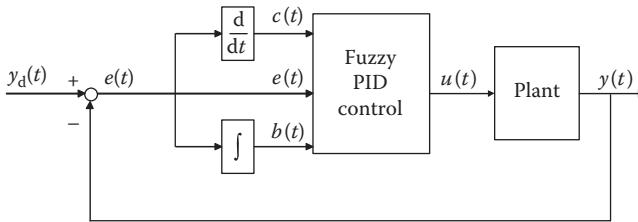


FIGURE 7.1 Fuzzy PID control system in continuous domain.

$$c(t) = \frac{d}{dt} e(t) \quad (7.2)$$

$$b(t) = \int e(\tau) d\tau \quad (7.3)$$

where

- $y_d(t)$ is the desired system output
- $y(t)$ is the actual output

The closed-loop fuzzy PID control system is schematically illustrated in Figure 7.1.

The PID fuzzy control rule base is given in the form of three-input single-output as

$$R: \text{IF } e_1 \text{ is } E_k \text{ AND } e_2 \text{ is } E_1 \text{ AND } e_3 \text{ is } E_q, \text{ THEN } u \text{ is } U \quad (7.4)$$

where

- e_1 is the error $e(t)$
- e_2 is the change of the error $c(t)$
- e_3 is the integral of the error $b(t)$ represents the control action $u(t)$
- E_k , E_1 , and E_q are the linguistic variables of e_1 , e_2 , and e_3 , respectively
- U is the linguistic variable of u

Similar to a conventional PID controller, the derivative term e_2 is mainly to dampen the oscillatory response in transient time and the integral term e_3 is to eliminate the system steady-state error. The defuzzified control action $u(t)$ is calculated based on the three inputs by the fuzzy inferencing calculation as

$$u(t) = f(e(t), c(t), b(t)) \quad (7.5)$$

where $f(\cdot)$ is a nonlinear fuzzy input–output mapping determined by the fuzzy control rules.

In many applications, the three control inputs and the control output are normalized by multiplying appropriate scaling factors, GE, GC, GI, and GU, to expand or shrink the corresponding variables within the normalized range $[-1,1]$. These scaling factors have a strong influence on the dynamic response of the closed-loop

system in terms of rise time, oscillation, and overshoot level. Generally, increasing the input scaling factors (GE and GC) makes the system performance more sensitive around the set point and induces response oscillation; decreasing output scaling factor GU causes a slow rising time and thus extends the system stabilization period (Procyk and Mamdani, 1979; Passino and Yurkovich, 1998).

Since a fuzzy PID controller has three inputs, it inevitably expands the fuzzy rule base into multidimension and imposes high computational load for real-time implementation. For example, if a number of N fuzzy sets are defined for each control input $e(t)$, $c(t)$, and $b(t)$, then a total N^3 fuzzy rules will be required to cover the complete three-dimensional control space in order to derive the appropriate control action. Thus, in some cases, other types of fuzzy PID control derivations are adopted in real applications to reduce the online calculation load, such as fuzzy P control, fuzzy PI control, fuzzy PD control, or any combination of these control schemes:

$$\begin{aligned} \text{Fuzzy P control: } u(t) &= f(e(t)) \\ \text{Fuzzy PI control: } u(t) &= f(e(t), b(t)) \\ \text{Fuzzy PD control: } u(t) &= f(e(t), c(t)) \end{aligned} \quad (7.6)$$

In general, a fuzzy PI controller gives a poor performance during transient time for higher-order systems due to the internal integration operation, while a fuzzy PD controller has a difficulty in removing the system steady-state error.

In actual implementation, the fuzzy control inferencing is usually performed in a personal computer (PC) or a microprocessor and executed in discrete domain, where the control inputs are the error $e(k)$, the difference of the error $c(k)$, and the summation of the error $b(k)$:

$$e(k) = y_d(k) - y(k) \quad (7.7)$$

$$c(k) = e(k) - e(k-1) \quad (7.8)$$

$$b(k) = \sum_{i=1}^k e(i) \quad (7.9)$$

where

- $y_d(k)$ is the desired system output at the sampling time instant k
- $y(k)$ is the actual output

The fuzzy PID controller in discrete-time domain can be illustrated in [Figure 7.2](#) and the corresponding defuzzified control action $u(k)$ is calculated as

$$u(k) = f(e(k), c(k), b(k)) \quad (7.10)$$

The fuzzy control calculation is implemented in discrete-time domain and the control action $u(k)$ passes through a zero-order hold (ZOH) to obtain a continuous signal entering into the plant.

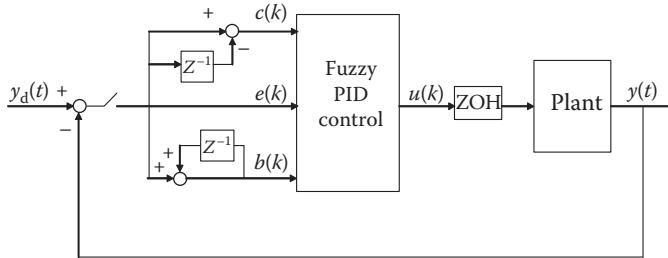


FIGURE 7.2 Fuzzy PID control system in discrete domain.

This fuzzy control systems explained above is in absolute form. Sometime, the fuzzy PID controller can also be designed in incremental form, where the control output is the change of the control action as follows:

$$R: \text{IF } e_1 \text{ is } E_k \text{ AND } e_2 \text{ is } E_1 \text{ AND } e_3 \text{ is } E_q, \text{ THEN } \Delta u \text{ is } \Delta U \quad (7.11)$$

The defuzzified control action $\Delta u(k)$ is calculated as

$$\Delta u(k) = f(e(k), c(k), b(k)) \quad (7.12)$$

And the final control action $u(k)$ is obtained in incremental form as

$$u(k) = u(k - 1) + \Delta u(k) \quad (7.13)$$

A fuzzy PID controller is inherently a piecewise linear PID controller by the fuzzy rule base establishment and fuzzy inferencing mechanism. It has a better control capability than a conventional linear PID controller within the overall operating range, especially for nonlinear systems, since the nonlinear terms are initially included in the fuzzy rule table construction (Li et al., 2001a). In the region close to the origin, a fuzzy PID controller functionally behaves approximately as a linear PID controller. When the system cannot be represented by an explicit analytical model, a fuzzy PID control will provide its superior, effective performance and the ease of implementation for real-time application. Compared to other complicated fuzzy control strategies, a fuzzy PID controller has a simpler structural configuration and hence is more practically useful.

Example 7.1 (Golob, 1999)

Consider a simple electromagnetic suspension system shown in Figure 7.3, where the electromagnetic force is opposite to the gravity of the suspended steel ball and intends to maintain the ball levitated in the mid air.

The magnetic force F depends on the electromagnet current I , electromagnet characteristics, and air gap X between the steel ball and the electromagnet. The dynamics of the steel ball in the magnetic field is derived based on Newton's second law of motion as

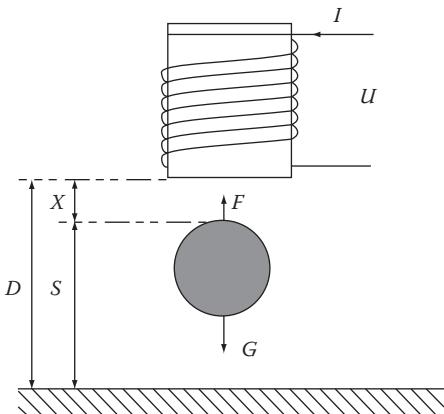


FIGURE 7.3 Simple electromagnetic suspension system. (From Golob, M., *Advances in Soft Computing: Engineering Design and Manufacturing*, Springer, NY, 1999. With permission.)

$$F - G = m \cdot \frac{d^2 X}{dt^2} \quad (7.14)$$

where

X is the air gap

m is the mass of the suspended steel ball

$G = mg$ is the gravity force

The voltage equation of the electromagnetic coil is expressed as

$$U = L \cdot \frac{dI}{dt} + I \cdot R \quad (7.15)$$

where

U is the controlled voltage applied to the electromagnet

L is the inductance of the electromagnet

R is the coil resistance

The nominal system parameters are shown in [Table 7.1](#).

This system is unstable and is controlled by a fuzzy PID controller through real-time implementation. The fuzzy PID controller is implemented in ANSI-C language on a PC with the sampling frequency of 1 kHz. The ball position is measured by an optical contact-free position sensor and the position error is calculated as the difference between the set point and the current measured ball position. The control inputs are the position error, the derivative of the error, and the integral of the error. The control output is the voltage level applied to the electromagnet. The physical magnetic system communicates with the PC by the A/D and D/A converters with 12-bit resolution, which maps the signal to integers from 0 to 4095. For all three control inputs, five triangular membership functions (MFs) are defined over each operating range with 50% overlap as shown in [Figure 7.4a](#). The names of the MFs are defined as negative big (NB), negative small (NS),

TABLE 7.1
Nominal System Parameters

Parameters	Values
Number of coil n	1200
Maximum air gap D	0.025 m
Mass of the steel ball m	0.147 kg
Electromagnet inductance L	520 mH
Coil resistance R	2.8 Ω

Source: Golob, M., *Advances in Soft Computing: Engineering Design and Manufacturing*, Springer, NY, 1999, pp. 215–227. With permission.

zero (ZE), positive small (PS), and positive big (PB). For the control output, five singleton MFs are defined as shown in Figure 7.4b.

The fuzzy PID controller with three inputs and one output can be linguistically expressed as

$$\begin{aligned}
 R^1: & \text{IF } X_{1(1)} \text{ is } E_{1(1)} \text{ AND } X_{2(1)} \text{ is } E_{2(1)} \text{ AND } X_{3(1)} \text{ is } E_{3(1)}, \text{ THEN } Y_{(1)} \text{ is } U_{(1)} \\
 R^2: & \text{IF } X_{1(2)} \text{ is } E_{1(2)} \text{ AND } X_{2(2)} \text{ is } E_{2(2)} \text{ AND } X_{3(2)} \text{ is } E_{3(2)}, \text{ THEN } Y_{(2)} \text{ is } U_{(2)} \\
 & \dots \quad \dots \quad \dots \\
 R^m: & \text{IF } X_{1(m)} \text{ is } E_{1(m)} \text{ AND } X_{2(m)} \text{ is } E_{2(m)} \text{ AND } X_{3(m)} \text{ is } E_{3(m)}, \text{ THEN } Y_{(m)} \text{ is } U_{(m)}
 \end{aligned} \tag{7.16}$$

where

$i = 1, 2, \dots, m$ is the number of fuzzy rules

$X_{k(i)}$ is the fuzzy set of the k th input variable defined in the universe of discourse, $k = 1, 2, 3$

$Y_{(i)}$ is the output fuzzy set defined in the universe of discourse

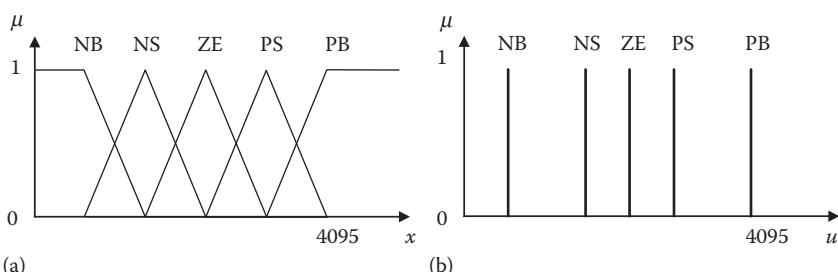


FIGURE 7.4 Input and output MFs: (a) input MFs and (b) output MFs. (From Golob, M., *Advances in Soft Computing: Engineering Design and Manufacturing*, Springer, NY, 1999. With permission.)

The overall fuzzy relation R is an aggregation of m fuzzy rule relations as

$$R = \bigcup_{i=1}^m R^i = \bigcup_{i=1}^m \{X_{1(i)} \wedge X_{2(i)} \wedge X_{3(i)} \wedge Y_{(i)}\} \quad (7.17)$$

Given the current fuzzy inputs X'_1 , X'_2 , and X'_3 , the fuzzy output Y' is calculated based on the compositional rule of inference as

$$Y' = (X'_1 \wedge X'_2 \wedge X'_3) \circ R \quad (7.18)$$

Due to the multidimensionality of the fuzzy PID relation (Equation 7.17), the number of rules increases as the third power of the number of MFs for each input variable and the final rule base may contain maximum $5^3 = 125$ rules. The composition rule of inference (Equation 7.18) will be difficult to perform for real-time operation. In order to minimize the number of rules and reduce the calculation load, a multivariable fuzzy control rule decomposition method is proposed, where the complex rule base (R) with multiple inputs is reduced to a number of simple rule bases (R_1^i , R_2^i , and R_3^i) with only one variable in the antecedent. For example, consider a complex PID fuzzy control rule base with two rules R^1 and R^2 :

$$\begin{aligned} Y' &= (X'_1 \wedge X'_2 \wedge X'_3) \circ R \\ &= (X'_1 \wedge X'_2 \wedge X'_3) \circ \{R^1 \vee R^2\} \\ &= (X'_1 \wedge X'_2 \wedge X'_3) \circ \{(X_{1(1)} \wedge X_{2(1)} \wedge X_{3(1)} \wedge Y_{(1)}) \vee (X_{1(2)} \wedge X_{2(2)} \wedge X_{3(2)} \wedge Y_{(2)})\} \end{aligned} \quad (7.19)$$

With max–min composition, Mamdani min implication, and min operation for the antecedent intersection, the multidimensional fuzzy rule with multiple independent variables in the antecedent can be simplified into a set of simple rule bases and the fuzzy output will be formulated as

$$\begin{aligned} Y'_s &= X'_1 \circ \{R_1^1 \vee R_1^2\} \wedge X'_2 \circ \{R_2^1 \vee R_2^2\} \wedge X'_3 \circ \{R_3^1 \vee R_3^2\} \\ &= X'_1 \circ \{(X_{1(1)} \wedge Y_{(1)}) \vee (X_{1(2)} \wedge Y_{(2)})\} \\ &\quad \wedge X'_2 \circ \{(X_{2(1)} \wedge Y_{(1)}) \vee (X_{2(2)} \wedge Y_{(2)})\} \\ &\quad \wedge X'_3 \circ \{(X_{3(1)} \wedge Y_{(1)}) \vee (X_{3(2)} \wedge Y_{(2)})\} \end{aligned} \quad (7.20)$$

Considering the interaction effect in the antecedent variables, the relationship of these two fuzzy outputs is $Y' \subseteq Y'_s$. Through this multivariable control rule simplification mechanism, the inference of the three-dimensional rule base is decomposed into three individual rule bases: the proportional rule base R_1^i , the differential rule base R_2^i , and the integral rule base R_3^i . Each rule base contains five rules as shown in [Table 7.2](#).

In this way, the multidimensional PID fuzzy control rule base is decomposed into three sets of one-dimensional rule bases for each input and fuzzy inferencing calculation is easier to execute for real-time implementation. The center of area (COA) defuzzification method is utilized in the actual experiment. The fuzzy

TABLE 7.2
Decomposed Fuzzy Control Rule Base

Fuzzy Rule Number	1	2	3	4	5
Control input	NB	NS	ZE	PS	PB
Control output	NB	NS	ZE	PS	PB

Source: Golob, M., *Advances in Soft Computing: Engineering Design and Manufacturing*, Springer, NY, 1999, pp. 215–227. With permission.

control output $u(k)$ is the average result of the proportional, differential, and integral actions as

$$u(k) = \frac{f_P[e(k)] + f_D[c(k)] + f_I[b(k)]}{3} \quad (7.21)$$

where $f_P(\cdot)$, $f_D(\cdot)$, and $f_I(\cdot)$ are nonlinear fuzzy input–output mappings determined by the three simplified rule bases.

In the experiments, the control performance of the fuzzy PID controller is compared with a conventional linear discrete PID controller. First, the magnetic ball position is controlled by a conventional linear PID controller, where optimal gain parameters are used when the nominal air gap set point is $h=16$ mm. The resultant performance is illustrated in Figure 7.5a. For this nonlinear magnetic suspension system, the linear PID controller works successfully for the local linearized system around the set point at 14–18 mm. However, when the operating condition is changed to 4–8 mm, the output response has inevitable large overshoot and oscillation. Comparably, the fuzzy PID control performance is shown in Figure 7.5b. The system performance near the set point at 14–18 mm is similar to the case with the conventional linear PID controller. When the system is operated around the set point at 4–8 mm, the oscillation is noticeably eliminated and the overshoot is drastically reduced compared to the result with the conventional linear PID controller. Furthermore, it should be emphasized that the fuzzy control results are obtained by using triangular MFs with five rules. The control performance can be improved by adopting nonlinear fuzzy MFs and by adding more fuzzy rules in the control system.

The control performance indices are summarized and presented in Table 7.3. By comparison, the fuzzy PID controller gives better performance than a traditional linear PID controller over the entire operating range. This fuzzy inference decomposition mechanism can be extended to a complex multivariable rule base with more input variables in the antecedent.

7.1.2 HYBRID FUZZY CONTROL

For a complex dynamic system with apparent nonlinear characteristics, a fuzzy controller can work with a conventional controller in parallel as a hybrid fuzzy control system, which was introduced by Pedrycz (1993, 1995) as shown in Figure 7.6. In this structure, the system nonlinearity far away from the set point can

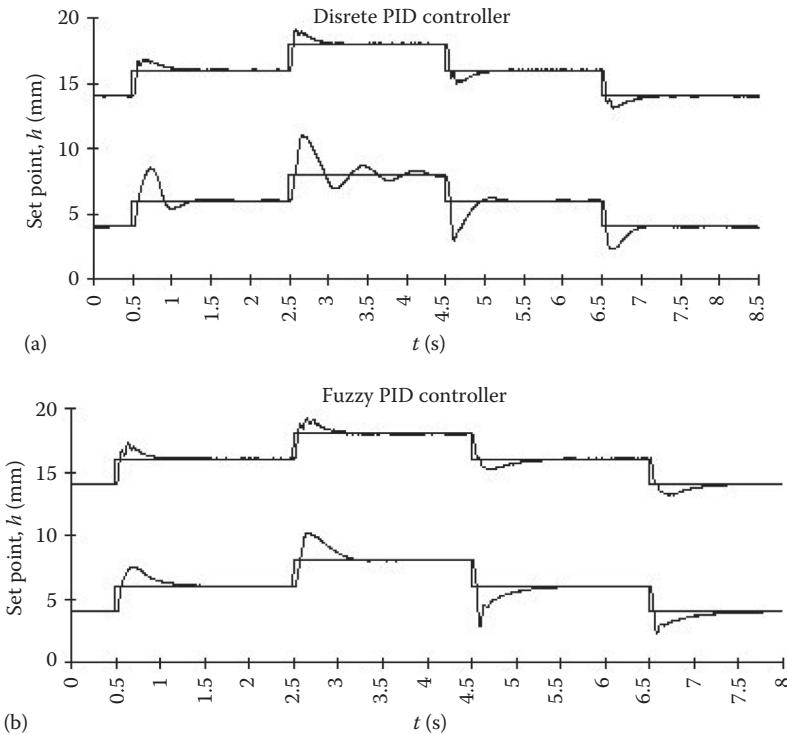


FIGURE 7.5 Control performance comparison: (a) conventional linear PID control performance and (b) fuzzy PID control performance. (From Golob, M., *Advances in Soft Computing: Engineering Design and Manufacturing*, Springer, NY, 1999. With permission.)

TABLE 7.3
Performance Indices

Performance Index	Equation	Integrated Absolute Error (IAE)	Integrated Squared Error (ISE)
		$\frac{1}{N} \sum_{k=1}^N y_{sp}(k) - y(k) $	$\frac{1}{N} \sum_{k=1}^N [y_{sp}(k) - y(k)]^2$
Set point 4–8 mm	Linear PID controller	0.392	0.613
	Fuzzy PID controller	0.319	0.400
Set point 14–18 mm	Linear PID controller	0.163	0.136
	Fuzzy PID controller	0.159	0.127

Source: Golob, M., *Advances in Soft Computing: Engineering Design and Manufacturing*, Springer, NY, 1999, pp. 215–227. With permission.

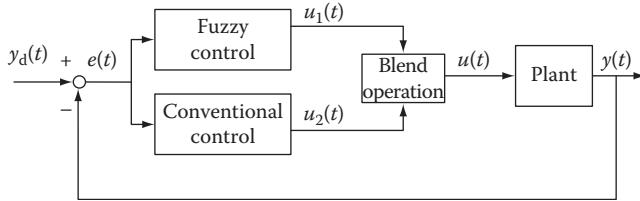


FIGURE 7.6 Hybrid fuzzy control system.

be quickly compensated by the fuzzy controller and the conventional controller is designed to guarantee the accurate performance near the set point.

The essential part of this control structure is the blend operation to provide the overall control signal $u(t)$, which is a composition of the signal produced by the fuzzy controller $u_1(t)$ and the one obtained from the conventional controller $u_2(t)$. It provides two distinct control modes when the state is far away or close by from the set point as shown in Figure 7.7.

When the state is far away from the set point, the fuzzy controller is predominant to generate a quick response and compensate for the system nonlinearity. As the state approaches the set point, the role of the fuzzy controller diminishes and the conventional controller takes over the control responsibility to guarantee the performance accuracy. The blend operation from the far-away state to the close-by state can be performed by another level of fuzzy controller (Pedrycz, 1993), which designates appropriate composition rate between the fuzzy control action $u_1(t)$ and the conventional control action $u_2(t)$. For example, the far-away degree of the current state away from the setpoint can be represented as

$$R: \text{IF } e_1 \text{ is } E_k \text{ AND } e_2 \text{ is } E_1, \text{ THEN } d_1 \text{ is } D_1 \quad (7.22)$$

where

e_1 is the error $e(t)$

e_2 is the change of the error $c(t)$

d_1 is the degree of the current state away from the set point

E_k , E_1 , and D_1 are the linguistic variables of e_1 , e_2 , and d_1 , respectively

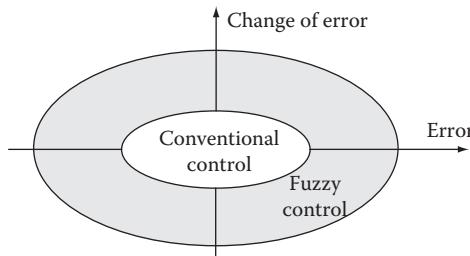


FIGURE 7.7 Close-by and faraway states. (From Pedrycz, W., *Fuzzy Sets Engineering*, CRC Press, Boca Raton, FL, 1995. With permission.)

The defuzzified far-away degree d_1 is calculated based on fuzzy inferencing calculation as

$$d_1 = f_1(e(t), c(t)) \quad (7.23)$$

where $f_1(\cdot)$ is a nonlinear fuzzy mapping determined by the fuzzy system (Equation 7.22). Similarly, the close-by degree of the current state away from the set point can be represented as

$$R: \text{IF } e_1 \text{ is } E_k \text{ AND } e_2 \text{ is } E_1, \text{ THEN } d_2 \text{ is } D_2 \quad (7.24)$$

And the defuzzified close-by degree d_2 is obtained as

$$d_2 = f_2(e(t), c(t)) \quad (7.25)$$

The overall control action is computed as weighted summation of the fuzzy control action $u_1(t)$ and the conventional control action $u_2(t)$ as

$$u(t) = \frac{u_1(t) \cdot d_1 + u_2(t) \cdot d_2}{d_1 + d_2} \quad (7.26)$$

With this designation, the hybrid fuzzy control structure will exhibit a dominant feature of the fuzzy controller when the state is far away from the set point. When the state is close to the set point, the conventional controller will take over the control responsibility and precisely guide the system to the desired output value.

Example 7.2 (Li et al., 2001b)

Currently, most commercial robot systems are equipped with a conventional linear PID controller due to its simple structure and ease of design. However, in practice, it is usually difficult to achieve the desired tracking performance in a broad operating range by a linear PID controller since the manipulator dynamic equations are tightly coupled and highly nonlinear. In this example, a hybrid fuzzy controller is implemented on a direct drive two-link manipulator, whose joints are driven by the electromagnetic forces of the motors. The dynamic equations are expressed as

$$\begin{aligned} \tau_1 &= x_1 \ddot{\theta}_1 + 2l_1 x_4 \cos(\theta_2) \ddot{\theta}_1 + x_3 \ddot{\theta}_2 + l_2 x_4 \cos(q_2) \ddot{\theta}_2 + x_5 \operatorname{sgn}(\dot{\theta}_1) + x_7 \dot{\theta}_1 \\ &\quad + g x_2 \cos(\theta_1) + g x_4 \cos(\theta_1 + \theta_2) \\ \tau_2 &= x_3 \ddot{\theta}_1 + l_1 x_4 \cos(\theta_2) \ddot{\theta}_1 + x_3 \ddot{\theta}_2 + \operatorname{sgn}(\dot{\theta}_2) x_6 + x_8 \dot{\theta}_2 + g x_2 \cos(\theta_1) \\ &\quad + g x_4 \cos(\theta_1 + \theta_2) \end{aligned} \quad (7.27)$$

where

τ_1 and τ_2 are the applied joint torques

θ_1 and θ_2 are the joint angles

l_1 and l_2 are the lengths of the two links
 x_1, \dots, x_4 and x_5, \dots, x_8 are the inertial parameters and the Coulomb/viscous friction coefficients, which are identified as

$$\begin{aligned} x_1 &= I_{zz1} + I_{zz2} + l_1^2 m_2 = 3.1 \text{ kg m}^2 \\ x_2 &= l_1 m_2 + m_1 p_{x1} = 7.5 \text{ kg m} \\ x_3 &= I_{zz2} = 0.53 \text{ kg m}^2 \\ x_4 &= m_2 p_{x2} = 1.06 \text{ kg m} \\ x_5 &= F_{s1} = 6 \text{ N m} \\ x_6 &= F_{s2} = 3 \text{ N m} \\ x_7 &= F_{v1} = 1 \text{ N m s} \\ x_8 &= F_{v2} = 0.5 \text{ N m s} \end{aligned} \quad (7.28)$$

The reference trajectory in the experiment is specified as

$$\begin{aligned} \theta_{\text{ref1}} &= 1.07 + 0.5 \cos(5.58t) \\ \theta_{\text{ref2}} &= -0.2727 + 0.2727 \cos(7.678t) \end{aligned} \quad (7.29)$$

In this example, three types of controllers are implemented on this coupled non-linear manipulator: conventional linear PID controller, conventional linear PI + D controller, and hybrid fuzzy PD + conventional I + D controller, which are expressed in the following equations as

- Conventional linear PID controller:

$$\tau_i(t) = K_{Pi} e_i(t) + K_{Ii} \int e_i(\sigma) d\sigma + K_{Di} \dot{\theta}_i(t) \quad (7.30)$$

- Conventional linear PI + D controller:

$$\tau_i(t) = K_{Pi} e_i(t) + K_{Ii} \int e_i(\sigma) d\sigma - K_{Di} \dot{\theta}_i(t) \quad (7.31)$$

where $-K_{Di} \dot{\theta}_i(t)$ performs as the system damping term and is helpful for the stability of the controlled system. The incremental form of this linear PI + D controller can be expressed in discrete domain as

$$\begin{aligned} \Delta \tau_i(k) &= K_{Pi} [e_i(k) - e_i(k-1)] + K_{Ii} T e_i(k) - K_{Di} \\ &\times \frac{\theta_i(k) - 2\theta_i(k-1) + \theta_i(k-2)}{T^2} \end{aligned} \quad (7.32)$$

- Hybrid fuzzy PD + conventional I + D controller:

$$\Delta \tau_i(k) = K_{Pi} \Delta u_i(k) + K_{Ii} T e_i(k) - K_{Di} \frac{\theta_i(k) - 2\theta_i(k-1) + \theta_i(k-2)}{T^2} \quad (7.33)$$

where $\Delta u_i(k)$ is the fuzzy PD control output.

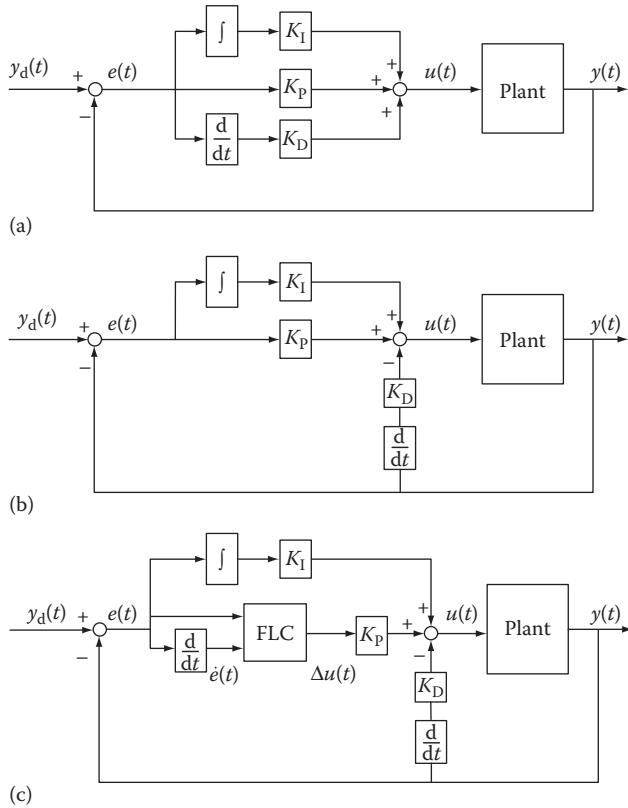


FIGURE 7.8 Three control schematic diagrams for the two-link manipulator: (a) conventional linear PID controller, (b) conventional linear PI + D controller, and (c) hybrid fuzzy PD + conventional I + D controller. (From Li et al., *Fuzzy Sets Syst.*, 122L, 125, 2001b. With permission.)

The three closed-loop control systems are shown in Figure 7.8. As can be observed in Figure 7.8c, the hybrid fuzzy control scheme consists of a fuzzy PD controller and a conventional integral (I) + derivative (D) controller. It is constructed by using a fuzzy PD controller in place of the proportional term in the conventional PI + D controller. This hybrid fuzzy PD + conventional I + D controller combines the advantages of a fuzzy controller and a conventional controller, where the fuzzy PD control term plays an important role in reducing the overshoot and shortening the rise time, the conventional integral (I) control part eliminates the steady-state error, and the conventional derivative (D) control part is responsible for reducing the output oscillation. The fuzzy control inputs are the error $e_i(t)$ and the change of error $\dot{e}_i(t)$. Three MFs are defined for each input/output variable as negative (N), zero (Z), and positive (P). The fuzzy rule base is defined in Table 7.4. Max-min composition and COA defuzzification are used in the fuzzy inferencing calculation.

TABLE 7.4
Fuzzy Rule Base of the Fuzzy PD
Control Part

$\Delta u_i(t)$	$e_i(t)$		
	<i>N</i>	<i>Z</i>	<i>P</i>
$\dot{e}_i(t)$	<i>N</i>	<i>N</i>	<i>N</i>
	<i>Z</i>	<i>Z</i>	<i>Z</i>
	<i>P</i>	<i>Z</i>	<i>P</i>

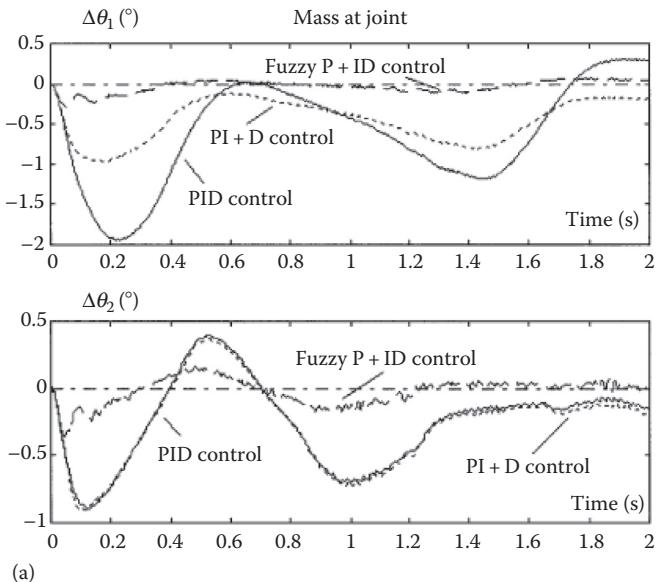
Source: Li, W. et al., *Fuzzy Sets Syst.*, 122L, 125, 2001b. With permission.

In this manipulator, a mass is mounted on the second link, which is movable from the joint to the tip. As the mass position changes, the inertial parameters will change and thus affect the manipulator's dynamics. The Coulomb and viscous friction coefficients of the manipulator are also uncertain as the manipulator rotates, which add on the overall control difficulty. The sampling time is specified as 2 ms in the experiment and the results are shown in Figure 7.9. Figure 7.9a is the tracking errors when the mass is placed at the joint, while Figure 7.9b shows the tracking errors when the mass is at the tip. It can be observed that in both cases, the accuracy of the hybrid fuzzy PD + conventional I + D controller is much better than that of the conventional linear PID controller. The statistical results are listed in Table 7.5 with the maximum deviation (MD) and the standard deviation (SD) comparisons in each case. The tracking precision is improved and the friction variations are compensated successfully with the hybrid fuzzy PD + conventional I + D controller, which demonstrates its effectiveness and robustness in practical operations.

7.1.3 SUPERVISORY FUZZY CONTROL

For complex practical systems, a single-level control system may not effectively achieve the control objective over a wide operating range, which makes a supervisory control structure particularly necessary when a precise plant model is not available. In a typical supervisory fuzzy control system, the lower level controller usually consists of a conventional controller to execute the fast direct control, while the higher level controller is designed as a fuzzy system to perform supervisory operation (Pedrycz, 1993, 1995). The diagram for a generalized supervisory fuzzy control system is shown in Figure 7.10, where the fuzzy system adjusts the conventional control parameters online according to certain heuristic rules. This structure is especially necessary when the plant is subject to unpredictable plant variations or unknown disturbances. It is generally capable of achieving better performance than conventional linear PID controller with fixed control parameters.

The basic idea behind the supervisory fuzzy control system is to decompose a complex control strategy into a group of local control algorithms, which are designed as a set of linear cases within their specific valid ranges. The design of supervisory fuzzy rules is straightforward, which is often not related to deriving certain control



(a)

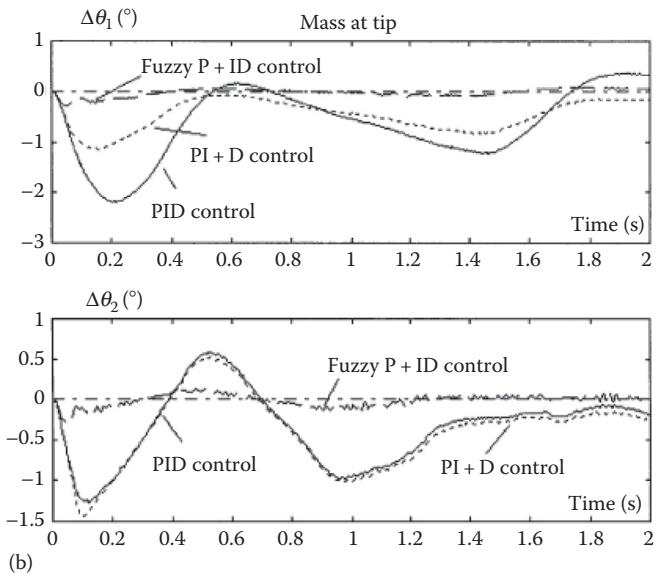


FIGURE 7.9 Tracking errors of the manipulator with mass position changes: (a) mass at joint and (b) mass at tip. (From Li et al., *Fuzzy Set Syst.*, 122L, 125, 2001b. With permission.)

action, but rather invoking the tuning of the lower level conventional controller with respect to the system status recognized by the fuzzy system. The criterion often relates to the satisfactory degree of the system performance (e.g., the output error) to the required conventional control parameter value changes. For example, in case

TABLE 7.5**Tracking Control Performance with Mass Position Changes**

	Performance Index	Maximum Deviation		Standard Deviation	
		Joint 1 (°)	Joint 2 (°)	Joint 1 (°)	Joint 2 (°)
Mass at joint	Conventional linear PID	1.9411	0.8946	0.6574	0.3315
	Conventional linear PI + D	0.9771	0.9077	0.2642	0.3127
	Hybrid fuzzy PD + conventional I + D	0.4707	0.2377	0.1434	0.0568
Mass at tip	Conventional linear PID	2.1831	1.2649	0.7580	0.4713
	Conventional linear PI + D	1.1586	1.4432	0.2998	0.4468
	Hybrid fuzzy PD + conventional I + D	0.5433	0.6116	0.1603	0.1078

Source: Li, W. et al., *Fuzzy Set Syst.*, 122L, 125, 2001b. With permission.

when the conventional controller is a linear PID controller, the supervisory fuzzy rules can be specified as

$$R^i: \text{IF } e_1 \text{ is } E_k^i \text{ AND } e_2 \text{ is } E_1^i \text{ AND } e_3 \text{ is } E_q^i, \text{ THEN } k_p \text{ is } K_p^i \text{ AND } k_I \text{ is } K_I^i \text{ AND } k_D \text{ is } K_D^i \quad (7.34)$$

where

$i = 1, 2, \dots, n$ is the number of fuzzy rules

e_1 is the error $e(t)$

e_2 is the change of the error $c(t)$

e_3 is the integral of the error $b(t)$

k_p , k_I , and k_D are the conventional PID control gains

E_k^i , E_1^i , E_q^i , K_p^i , K_I^i , and K_D^i are the corresponding linguistic variables

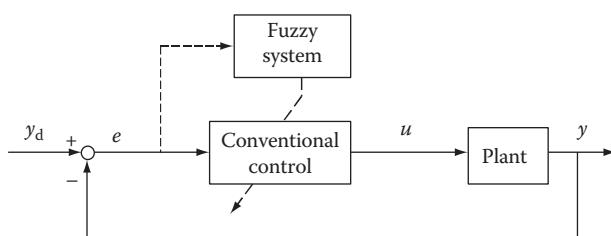


FIGURE 7.10 Supervisory fuzzy control system.

The associated fuzzy inferencing procedure can be completed in two steps as follows:

1. In the current status, the error $e(t)$, the change of error $\dot{e}(t)$, and the integral of the error $b(t)$ activate the fuzzy rule base, where the firing strength α_i of the i th rule is calculated as

$$\alpha_i = \mu_{E_k^i}(e_1) \wedge \mu_{E_l^i}(e_2) \wedge \mu_{E_q^i}(e_3)$$

2. The conventional control parameters, k_p , k_I , and k_D , are adjusted based on the defuzzified value from the higher level fuzzy system as

$$k_p = \frac{\sum_{i=1}^n K_p^i \cdot \alpha_i}{\sum_{i=1}^n \alpha_i}, \quad k_I = \frac{\sum_{i=1}^n K_I^i \cdot \alpha_i}{\sum_{i=1}^n \alpha_i}, \quad k_D = \frac{\sum_{i=1}^n K_D^i \cdot \alpha_i}{\sum_{i=1}^n \alpha_i} \quad (7.35)$$

Example 7.3 (Lee and Pang, 1994)

This example demonstrates a practical implementation of a DC servomotor speed control system operated by a supervisory fuzzy control system to compensate for large system variations and disturbances. The overall schematic diagram of the closed-loop control system is shown in Figure 7.11, where the parameters of the conventional PI controller are tuned online by the fuzzy system and thus the system performance is guaranteed for practical implementation in various operating conditions.

The overall control system consists of a higher level fuzzy system and a lower level conventional PI controller. The fuzzy system tunes the PI control gains, k_p

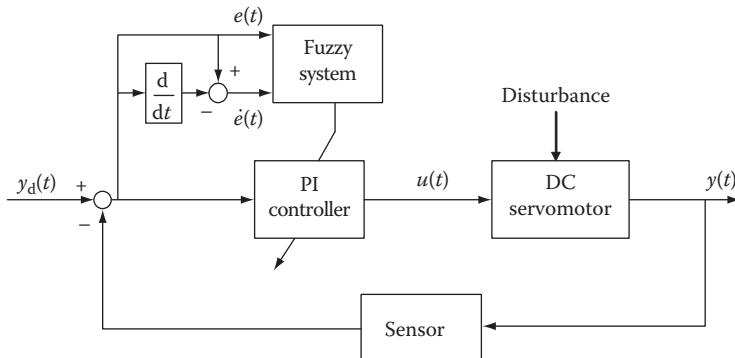


FIGURE 7.11 Supervisory fuzzy control system for a DC servomotor. (Modified from Lee, C.K. and Pang, W.H., *IEEE Conf. Publ., Michael Faraday House*, 2, 1088, 1994. With permission.)

TABLE 7.6
Fuzzy Rule Table for Tuning k_P

		Change of Error						
		LN	MN	SN	ZE	SP	MP	LP
Error	LN	ZE	SN	SN	MN	MN	LN	LN
	MN	SP	ZE	SN	SN	MN	MN	LN
	SN	SP	SP	ZE	SN	SN	MN	MN
	ZE	MP	SP	SP	ZE	SN	SN	MN
	SP	MP	MP	SP	SP	ZE	SN	SN
	MP	LP	MP	MP	SP	SP	ZE	SN
	LP	LP	LP	MP	MP	SP	SP	ZE

Source: Lee, C.K. and Pang, W.H., *IEEE Conf. Publ., Michael Faraday House*, 2, 1088, 1994. With permission.

and k_I , to compensate for system variations and external disturbances. An optical encoder sensor monitors the motor speed in real time. The error $e(t)$ and the change of error $\dot{e}(t)$ are obtained and enter into the fuzzy system for supervisory control decision.

The transfer function of the conventional PI controller is given by $G_C(s) = k_P + \frac{k_I}{s}$, where the proportional gain k_P is intended to minimize the output overshoot and oscillation within the shortest transient time, and the integral gain k_I is to eliminate the system steady-state error. The two inputs to the fuzzy system are the error $e(t)$ and the change of error $\dot{e}(t)$. The two outputs are the linear gain parameters k_P and k_I . Seven triangular MFs are defined for both input and output variables in the fuzzy system: large negative (LN), medium negative (MN), small negative (SN), zero (ZE), small positive (SP), medium positive (MP), and large positive (LP). The fuzzy rules for tuning these two gain parameters are generated based on human knowledge and are listed in Tables 7.6 and 7.7, respectively.

TABLE 7.7
Fuzzy Rule Table for Tuning k_I

		Change of Error						
		LN	MN	SN	ZE	SP	MP	LP
Error	LN	—	—	MP	LP	—	—	—
	MN	—	—	SP	MP	LP	—	—
	SN	MN	SN	ZE	SP	MP	LP	—
	ZE	LN	MN	SN	ZE	SP	MP	LP
	SP	—	LN	MN	SN	ZE	SP	MP
	MP	—	—	LN	MN	SN	—	—
	LP	—	—		LN	MN	—	—

Source: Lee, C.K. and Pang, W.H., *IEEE Conf. Publ., Michael Faraday House*, 2, 1088, 1994. With permission.

By examining the system error $e(t)$ and the change of error $\dot{e}(t)$, the proportional gain k_P and the integral gain k_I can be suitably adjusted to improve the system output response. For example, in [Table 7.6](#), consider the shallow element, as the error is LP (large positive) and the change of error is LN (large negative), which means the system output is smaller than the desired output while the error is decreasing. In such a condition, the large increase of the proportional gain k_P is desirable to accelerate the response. In [Table 7.7](#), at the start-up period when the system error and the change of error are large, the integral gain k_I is kept unchanged, which are shown as “—” in Table 7.7. As the system is approaching to stable, for example, consider the shallow element, as the error is SP (small positive) and the change of error is LP (large positive), which means the system steady-state error tends to increase. In such a condition, the integral gain k_I needs to be moderately increased to minimize the system steady-state error.

The system responses of the conventional linear PI controller and the supervisory fuzzy controller are compared in three different operating conditions. Figure 7.12a shows the servomotor system response in nominal operating condition with no system parameter variations, where the transfer function is specified as $G(s) = \frac{K}{s^2 + as + b}$. It can be observed that the supervisory fuzzy controller has similar system response as the conventional linear PI controller with short transient time and no output oscillation. Figure 7.12b and c is the simulation results with parameter variations, where the system output responses with the conventional linear PI controller has a large overshoot, severe oscillation, and long rising time. In comparison, the fuzzy system in the supervisory control structure can adjust the PI control gains online based on the observed control performance and the closed-loop system response is maintained to be similar as the response in the nominal working condition. The system transient period is short, and the overshoot and oscillation are eliminated.

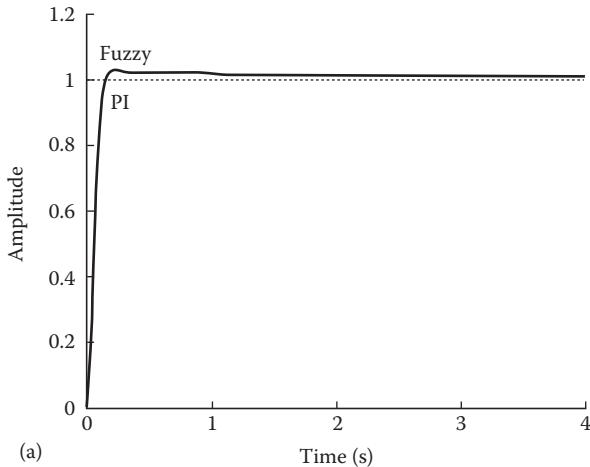


FIGURE 7.12 System response comparisons: (a) without system parameter variations ($\ddot{A}a = \ddot{A}b = 0$).

(continued)

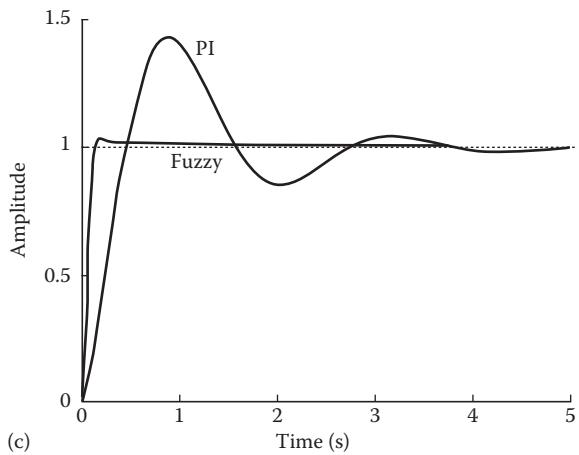
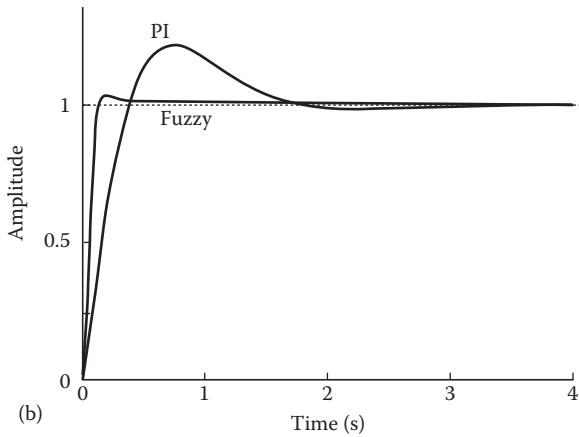


FIGURE 7.12 (continued) (b) With moderate parameter variations ($\ddot{A}a = 50\%$, $\ddot{A}b = 20\%$) and (c) with large parameter variations ($\ddot{A}a = 100\%$, $\ddot{A}b = 50\%$). (From Lee, C.K. and Pang, W.H., *IEEE Conf. Publ., Michael Faraday House*, 2, 1088, 1994. With permission.)

7.1.4 SELF-ORGANIZING FUZZY CONTROL

Static fuzzy controllers with fixed parameters are usually designed based on heuristic information from human experts. It is often difficult to properly choose the control parameters for complex or poorly defined dynamic systems. In most cases, the controllers are constructed for the nominal plant and the system performance cannot be guaranteed under various operating conditions due to the lack of online adaptation capability to compensate for unpredicted plant variation, noise, or disturbance. Therefore, different adaptive fuzzy control strategies have been studied as the requirement of the system performance increases. One attractive approach is the

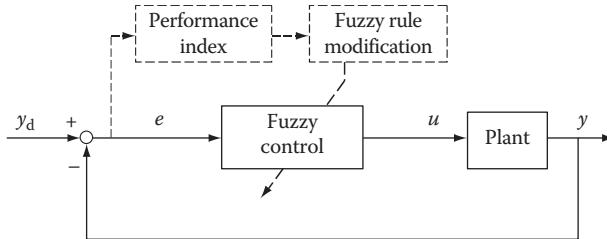


FIGURE 7.13 Self-organizing fuzzy control system.

linguistic self-organizing fuzzy controller, which was first proposed by Procyk and Mamdani (1979). This control structure has a learning algorithm (e.g., genetic algorithm, least-square estimation, gradient method, etc.) embedded in a classic fuzzy control structure. Based on the value of the performance index or objective function, the system performance is evaluated and an appropriate adaptation signal is generated to modify the control parameters online without *a priori* knowledge of plant dynamics. Several sets of control parameters can be possibly optimized online (Vishnupad, 1994):

- Input/output MFs
- Fuzzy control rules
- Input/output scaling factors

The functional block diagram of a general self-organizing fuzzy control system is shown in Figure 7.13, where the controller is composed of a basic level and a self-organizing level. The lower basic level is normally a classic fuzzy controller with fixed control parameters. The self-organizing level is designed to evaluate system response based on the value of the performance index and generate necessary modification instruction for the lower level controller without any plant mathematical model. In this way, a better system performance can be achieved and the system uncertainties can be effectively compensated.

Example 7.4 (Gao et al., 2000)

Consider a generic temperature control application shown in Figure 7.14. The temperature is measured by a suitable sensor, such as a thermocouple or a resistive thermal device (RTD), and converted to a signal readable by the controller. The controller compares the temperature to the desired set point, calculates the required control action, and then starts the control actuator, which alters the amount of heat being added to or taken from the process. The control objective is to regulate the temperature as close as possible to the set point within the shortest transient time. Currently, a conventional linear PID controller is broadly used in industry due to its easy implementation. However, different gain values are often required at the lower and higher level temperature ranges to avoid overshoot and

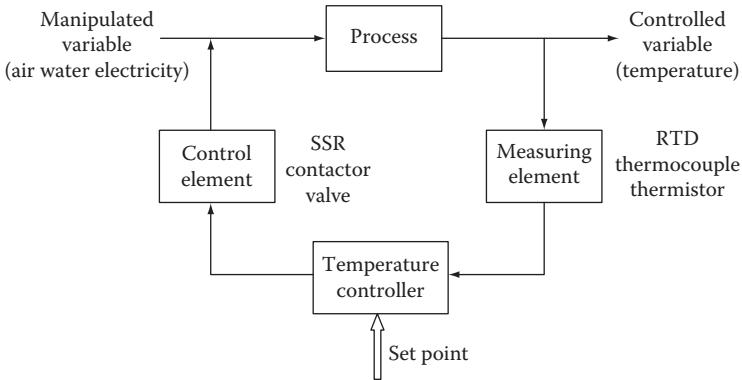


FIGURE 7.14 Typical industrial temperature control problem. (From Gao, Z., Trautzsch, T.A., and Dawson, J.G., *35th IAS Annu. Meet., IEEE Ind. Appl. Soc.*, 2, 1232, 2000. With permission.)

oscillation. It is usually costly and time consuming to determine the appropriate control gains over the entire operating range, and becomes even more problematic with varying time delay and unknown system nonlinearity. These practical difficulties motivate the investigation of intelligent adaptive control techniques as a substitute in improving the overall system performance.

In this example, a self-organizing fuzzy control structure is designed to regulate the industrial tank temperature with unknown process delays. As shown in Figure 7.15, in order to compensate for varying time delay and process dynamics changes, an adaptive fuzzy logic controller (FLC) is embedded in the first layer FLC and the overall control scheme is implemented in discrete-time domain using a ZOH scheme. The first layer fuzzy controller mimics what a simple linear PID controller would do when the fixed time delay is known and the control rules are determined based on the understanding of a conventional controller's working

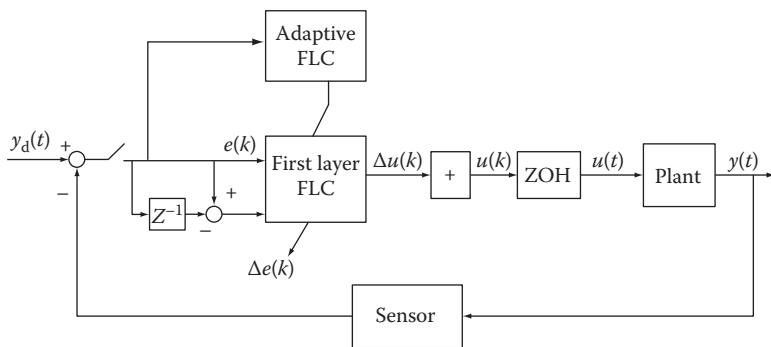


FIGURE 7.15 Self-organizing fuzzy controller for industrial temperature regulation. (Modified from Gao, Z., Trautzsch, T.A., and Dawson, J.G., *35th IAS Annu. Meet., IEEE Ind. Appl. Soc.*, 2, 1232, 2000. With permission.)

principle. The adaptive fuzzy controller is designed to compensate for the unknown and varying time delay.

In this example, the first layer fuzzy controller is a two-input single-output controller. The two inputs are the error $e(k)$ and the change of the error $\Delta e(k)$. Eight triangular MFs are defined over the operating range for each input variable and the linguistic labels are specified within the normalized universe of discourse as negative large (NL), negative medium (NM), negative small (NS), negative zero (NZ), positive zero (PZ), positive small (PS), positive medium (PM), and positive large (PL) with their individual peak position at $-0.6, -0.35, -0.2, 0, 0, 0.2, 0.35$, and 0.6 , respectively. Eight singleton output MFs are defined for the control output $\Delta u(k)$ and the linguistic names are specified in the same way as the input MFs as negative large (NL), negative medium (NM), negative small (NS), negative zero (NZ), positive zero (PZ), positive small (PS), positive medium (PM), and positive large (PL). The locations of these singleton MFs are at $-1, -0.7, -0.3, 0, 0, 0.3, 0.7$, and 1 , respectively.

The first layer fuzzy controller regulates the output temperature as the system time delay is fixed and known. Based on the understanding of the process and the knowledge of a conventional controller's behavior, 12 fuzzy rules are established as shown in Table 7.8. The resultant control action at the sampling time k is obtained in an incremental form as

$$u(k) = u(k - 1) + f(e(k), \Delta e(k)) \quad (7.36)$$

where $f(e(k), \Delta e(k))$ is the inferencing result based on COA defuzzification calculation.

In order to compensate for the unknown time delay and process dynamics variations, an adaptive fuzzy controller is designed to change the output universe of discourse of the first layer fuzzy controller based on the system performance observation in terms of rise time and overshoot. It is observed that changes in rise time and overshoot of the system output indicate the system time delay variation.

TABLE 7.8
First Layer Fuzzy Control Rules

		Change in Error							
		NL	NM	NS	NZ	PZ	PS	PM	PL
Error	NL				NL				
	NM			NM				PS	
	NS	—			NM	PS		—	
	NZ				NZ				
	PZ					PZ			
	PS	—		NS	PM			—	
	PM		NS				PM		
	PL				PL				

Source: Gao, Z., Trautzsch, T.A., and Dawson, J.G., *35th IAS Annu. Meet., IEEE Ind. Appl. Soc.*, 2, 1232, 2000. With permission.

TABLE 7.9
Adaptive Fuzzy Control Rules

Rise Time Rules

If tracking is	Then adjust is
SS	PS
MS	PM
VS	PL

Overshoot Rules

If overshoot is	Then adjust is
<i>L</i>	NL
<i>M</i>	NM
<i>S</i>	NS

Source: Gao, Z., Trautzsch, T.A., and Dawson, J.G., *35th IAS Annu. Meet., IEEE Ind. Appl. Soc.*, 2, 1232, 2000. With permission.

Rise time degree can be classified as very slow (VS), medium slow (MS), and slightly slow (SS). An increase in the output scaling factor can help to speed up the response. The overshoot degree is classified as large (*L*), medium (*M*), and small (*S*). Longer delay results in larger overshoot, which can be alleviated by reducing the loop gain appropriately. Based on this intuitive knowledge, six fuzzy rules are generated as the adaptation control law as listed in Table 7.9.

The developed self-organizing fuzzy controller is simulated for a tank temperature system shown in Figure 7.16. The temperature of the tank fluid with constant flow rates is controlled by adjusting the temperature of the incoming fluid. The incoming fluid temperature is adjusted by a mixing valve that controls the ratio of the hot and cold fluids in the supply line to the tank. The distance from the mixing valve to the tank represents the material transport delay in pipes. The temperature and the pressure of the fluids also affect the system delay level.

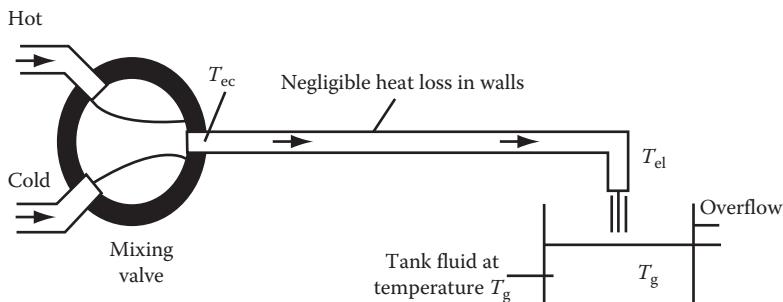


FIGURE 7.16 Tank temperature control system. (From Gao, Z., Trautzsch, T.A., and Dawson, J.G., *35th IAS Annu. Meet., IEEE Ind. Appl. Soc.*, 2, 1232, 2000. With permission.)

For comparison purposes, simulations are performed with three controllers and the results are compared against each other. The three controllers are a conventional PID controller, a Smith predictor controller (SPC), and a self-organizing FLC. For the nominal plant with 10 s time delay, the responses are shown in [Figure 7.17a](#). As expected, the SPC has the best response since the analytical plant model and the time delay are precisely known. Both the PID and self-organizing fuzzy controller have longer rising time and overshoot due to the lack of prediction mechanism. [Figure 7.17b](#) shows the results when the time delay is increased to 15 s, and [Figure 7.17c](#) is the case when the time delay is 20 s. In both cases, the self-organizing fuzzy controller is able to adapt itself and provides a satisfactory response. However, as illustrated in [Figure 7.17c](#), the PID controller drives the system unstable, where the amplitude of the output response increases as time increases. The SPC oscillates around the set point due to the mismatch generated by the inaccurate time delay parameter used in the plant model.

From the simulation result comparisons, the SPC provides the best response when an accurate plant model and the system delay are known. When the time delay is unknown, the proposed self-organizing fuzzy controller shows a significant improvement in maintaining the overall system performance and preserving the closed-loop stability. The rising time and overshoot degree are monitored by the adaptive layer fuzzy controller and the first layer output MFs are adjusted in real time. Thus, a satisfactory system performance is achieved and a limited numbers of fuzzy rules are adequate for unknown system dynamics variations or unpredicted uncertainties.

7.1.5 FUZZY MODEL REFERENCE LEARNING CONTROL

By combining the concept of the linguistic self-organizing control (Procyk and Mamdani, 1979) and the idea of the conventional model reference adaptive control (Åström and Wittenmark, 1989; Narendra and Annaswamy, 1989), Layne and Passino introduced the fuzzy model reference learning control (FMRLC) technique in 1992, which is able to tune the control parameters and remember the values that had been adjusted in the past. The schematic diagram of this type fuzzy control system is shown in [Figure 7.18](#), which contains four main components: the plant, the fuzzy controller to be tuned, the reference model, and the learning mechanism to modify the fuzzy rules online. Similar to the conventional model reference adaptive controller, this fuzzy control scheme employs a reference model to provide the desired closed-loop specifications (performance objectives) for synthesizing and tuning fuzzy control rules automatically, so that the closed-loop system behaves the same as the predefined reference model in the presence of various plant variations.

In [Figure 7.18](#), the inputs to the fuzzy controller are the error $e(k)$ and the change of error $c(k)$ as

$$e(k) = y_d(k) - y(k) \quad (7.37)$$

$$c(k) = e(k) - e(k - 1) \quad (7.38)$$

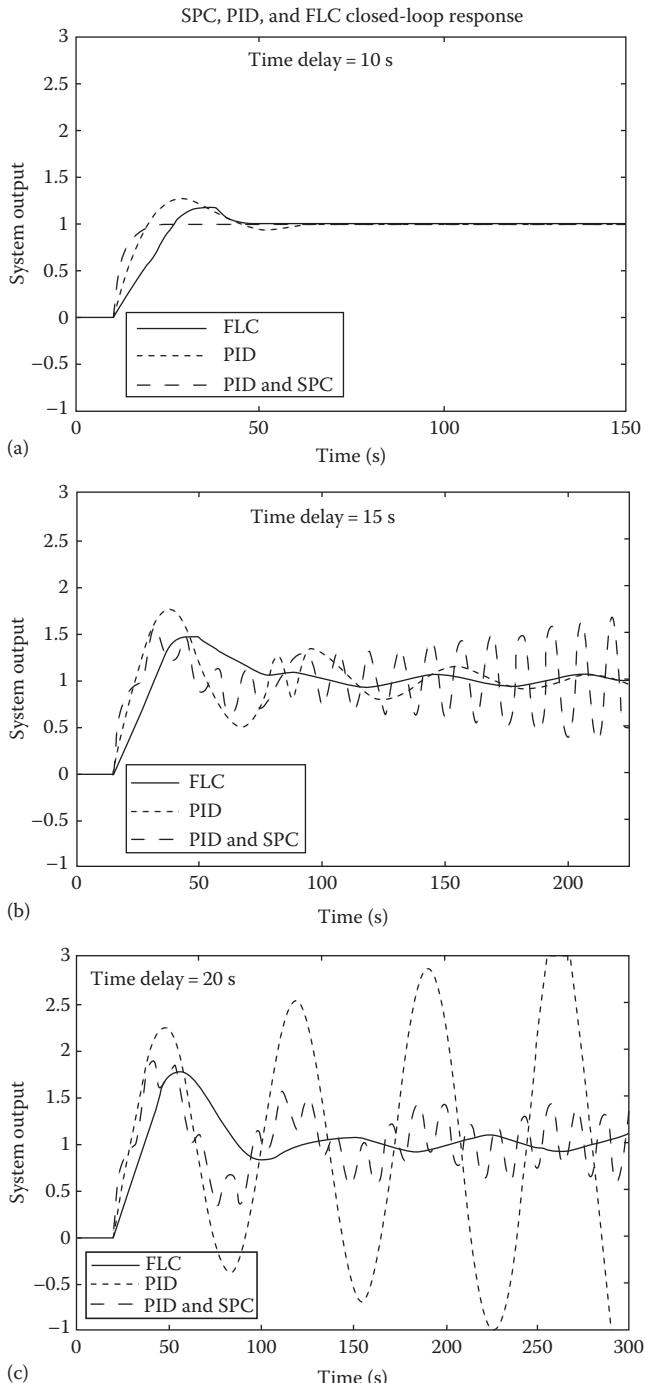


FIGURE 7.17 Control performance comparison: (a) time delay = 10 s, (b) time delay = 15 s, and (c) time delay = 20 s. (From Gao, Z., Trautzsch, T.A., and Dawson, J.G., *35th IAS Annu. Meet., IEEE Ind. Appl. Soc.*, 2, 1232, 2000. With permission.)

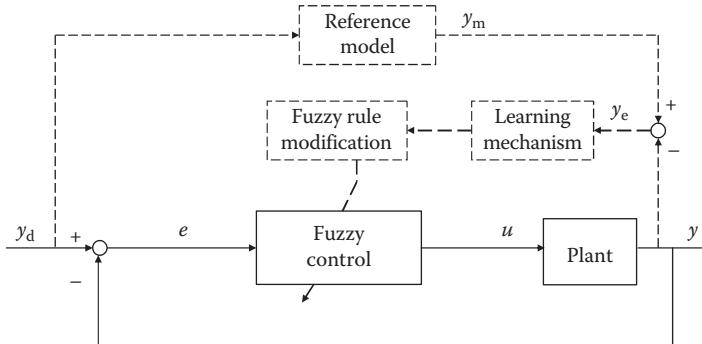


FIGURE 7.18 FMRLC system.

(i.e., a PD fuzzy controller) and the rule base of the fuzzy controller is in the form of

$$R: \text{IF } e \text{ is } E \text{ AND } c \text{ is } C, \text{ THEN } u \text{ is } U \quad (7.39)$$

where

e and c denote the error $e(k)$ and the change of the error $c(k)$

u represents the control action $u(k)$

E and C are the linguistic variables of e and c

U is the linguistic variable of u

The reference model in Figure 7.18 is specified to quantify the desired closed-loop system performance, where the error signal is

$$y_e(k) = y_m(k) - y(k) \quad (7.40)$$

Given that the reference model characterizes certain design criteria, such as the rise time, overshoot, etc., the desired performance of the controlled system is achieved if the error signal y_e is maintained to be small over the entire operating period. If the system performance is met (i.e., y_e is small), then the learning mechanism will not make a significant modification to the fuzzy controller. Otherwise, the desired system performance objective is not achieved and the learning mechanism will adjust the fuzzy controller. During the fuzzy rule modification process, first, the learning mechanism performs the function of mapping from the error signal $y_e(k)$ to the required changes in the plant input $p(k)$, which is necessary to force the plant output $y(k)$ follow the desired reference output $y_m(k)$. The inputs to the online learning mechanism block in Figure 7.18 are the error $y_e(k)$ and the change of error $y_c(k)$ as

$$y_e(k) = y_m(k) - y(k) \quad (7.41)$$

$$y_c(k) = y_e(k) - y_e(k-1) \quad (7.42)$$

and the modification rule is in the form of

$$R: \text{IF } y_e \text{ is } Y_e \text{ AND } y_c \text{ is } Y_c, \text{ THEN } p \text{ is } P \quad (7.43)$$

where

y_e and y_c denote the error signal $y_e(k)$ and the change of the error $y_c(k)$

p is the necessary changes to the plant input

Y_e , Y_c , and P are the linguistic variables of y_e , y_c , and p , respectively

Unlike the standard linguistic self-organizing control scheme, where the fuzzy rule base modification is performed on the entire fuzzy relation space and thus the computational load is usually high. In this FMRLC strategy, the fuzzy rule base modification is executed only to the fuzzy rules that have been fired. Assume the center of the activated consequent MF is b_m . The fuzzy control rule base is modified by shifting the center position by the value of p as

$$b_m(k) = b_m(k - 1) + p(k) \quad (7.44)$$

The output MFs of the fuzzy rules that are not activated do not need to be modified, which realizes local learning and the fuzzy rule modification in the past is remembered.

This FMRLC scheme has been studied in simulations and practical implementations for a variety of applications, including a cart-pendulum system (Layne and Passino, 1992), antiskid brakes in adverse road conditions (Layne et al., 1992, 1993), cargo ship steering (Layne and Passino, 1993), fault-tolerant aircraft control (Kwong et al., 1995), two-link flexible robot (Moudgal et al., 1994, 1995), magnetic ball suspension (Kwong and Passino, 1996), liquid level control in a surge tank (Zumberge and Passino, 1996), and others. In general, this control strategy achieves faster convergence than the conventional control method with satisfactory disturbance rejection property, such as wind disturbance rejection in cargo ship steering system, and can be easily extended to control complex multivariable plants. It is independent of the explicit mathematical model of the underlying plant, which is required for the conventional counterpart controller. However, there is no guarantee of the system stability and signal convergence. The control performance largely depends on a judicious choice of the reference model and a different reference input may make the system unstable.

Example 7.5 (Layne and Passino, 1996)

In this example, the FMRLC scheme is designed and employed to control the velocity of a single-stage rocket. The mathematical model for this process is expressed by the following differential equation:

$$\frac{dv(t)}{dt} = c(t) \left(\frac{m}{M - mt} \right) - g_0 \left(\frac{R}{R + y(t)} \right) - 0.5v^2(t) \left(\frac{\rho_a A C_d}{M - mt} \right) \quad (7.45)$$

where

$v(t)$ is the rocket velocity at time t

$c(t)$ is the velocity of the exhaust gases

$y(t)$ is the altitude of the rocket (above sea level)

$g_0 = 9.8 \text{ m/s}^2$ is the acceleration due to gravity at sea level

$R = 6.37 \times 10^6 \text{ m}$ is the radius of the earth

$\rho_a = 1.21 \text{ kg/m}^3$ is the air density

$A = 1.0 \text{ m}^2$ is the maximum cross-sectional area of the rocket

$C_d = 0.3$ is the drag coefficient for the rocket

$M = 15,000 \text{ kg}$ is the initial mass of the rocket and fuel

$m = 100 \text{ kg/s}$ is the exhaust gases mass flow rate (approximately constant for some solid propellant rockets)

This rocket dynamics described in Equation 7.45 is a typical nonlinear time-varying process with a continuously decreasing mass due to the loss of fuel resulting from combustion and exhaust. The process input is the velocity of the exhaust gases $c(t)$ and the process output is the velocity of the rocket $v(t)$. The inputs to the fuzzy controller are the error of the rocket velocity $v_e(t)$ and the change of error of the rocket velocity $v_c(t)$. In this fuzzy controller design, 11 fuzzy sets are defined for each control input as shown in Figure 7.19, such that the MFs are triangular and evenly distributed within the universe of discourse. The control output is the velocity of the exhaust gases $c(k)$. The output fuzzy sets are also assumed to be triangular shaped with a width of 0.4 on the normalized universe of discourse. The gain parameters for the error $v_e(t)$, the change of error $v_c(t)$, and the control output $c(k)$ are specified as $g_e = \frac{1}{4000}$, $g_c = \frac{1}{2000}$, and $g_u = 10,000$, respectively. The fuzzy control rule base is initially defined with all zero entries. The sampling period is chosen as 100 ms.

The reference model for this process is expressed by the following differential equation:

$$\frac{dv_m(t)}{dt} = -0.2v_m(t) + 0.2v_r(t) \quad (7.46)$$

where

$v_m(t)$ specifies the desired system performance for the rocket velocity

$v_r(t)$ is the desired rocket velocity entering into the reference model

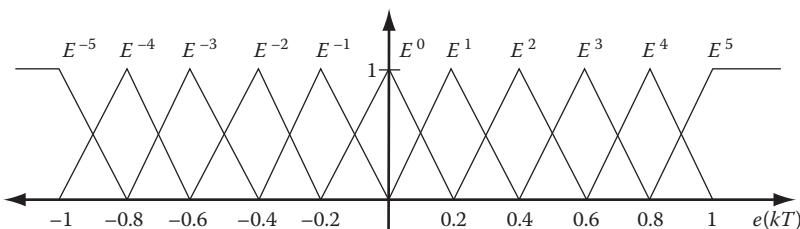


FIGURE 7.19 Input MFs. (From Layne, J.R. and Passino, K.M., *J. Intelligent Fuzzy Syst.*, 4, 33, 1996. With permission.)

TABLE 7.10**Fuzzy Rule Base for Control Rule Modification**

		Change in Error										
		-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
Error	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0
	-0.8	-1.0	-1.0	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2
	-0.6	-1.0	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4
	-0.4	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6
	-0.2	-1.0	-1.0	-0.8	-0.6	-0.4	-0.3	0.0	0.2	0.4	0.6	0.8
	0.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
	0.2	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.0
	0.4	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.0	1.0
	0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.0	1.0	1.0
	0.8	-0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.0	1.0	1.0	1.0
	1.0	0.0	0.2	0.4	0.6	0.8	1.0	1.0	1.0	1.0	1.0	1.0

Source: Layne, J.R. and Passino, K.M., *J. Intelligent Fuzzy Syst.*, 4, 33, 1996. With permission.

The inputs to the online learning mechanism block are the error $v_e(k)$ and the change of error $v_c(k)$ as

$$v_e(k) = v_m(k) - v(k) \quad (7.47)$$

$$v_c(k) = v_e(k) - v_e(k - 1) \quad (7.48)$$

The 11 fuzzy sets are defined with triangular MFs, which are evenly distributed within the universe of discourse. The normalizing gains for $v_e(k)$, $v_c(k)$, and $p(k)$ are specified as $g_{v_e} = \frac{1}{4000}$, $g_{v_c} = \frac{1}{2000}$, and $g_p = 10,000$, respectively. For the rocket process, an increase in the exhaust gas velocity generally results in an increase in the velocity of the exhaust gases. Consequently, the fuzzy rule base shown in Table 7.10 is employed for this rule modification process.

The simulation results for the FMRLC of the rocket are shown in Figure 7.20, which exhibits a satisfactory tracking performance with the reference model even when the mass of the rocket continuously decreases from the initial mass due to fuel loss. As the desired rocket velocity increases as step functions, the actual velocity follows the reference model response in a quick transient time with small oscillations. The simulation application illustrates the effectiveness of this fuzzy control strategy in controlling a general nonlinear time-varying process.

7.2 MODEL-BASED FUZZY CONTROL

The aforementioned knowledge-based fuzzy control techniques have been applied to various practical industrial processes due to the easy construction of fuzzy control rules. They are designed and tuned based on human knowledge without any system analytical models. On the other hand, fuzzy controllers can be also designed using an explicit system mathematical model, if obtainable. This type of fuzzy controller is

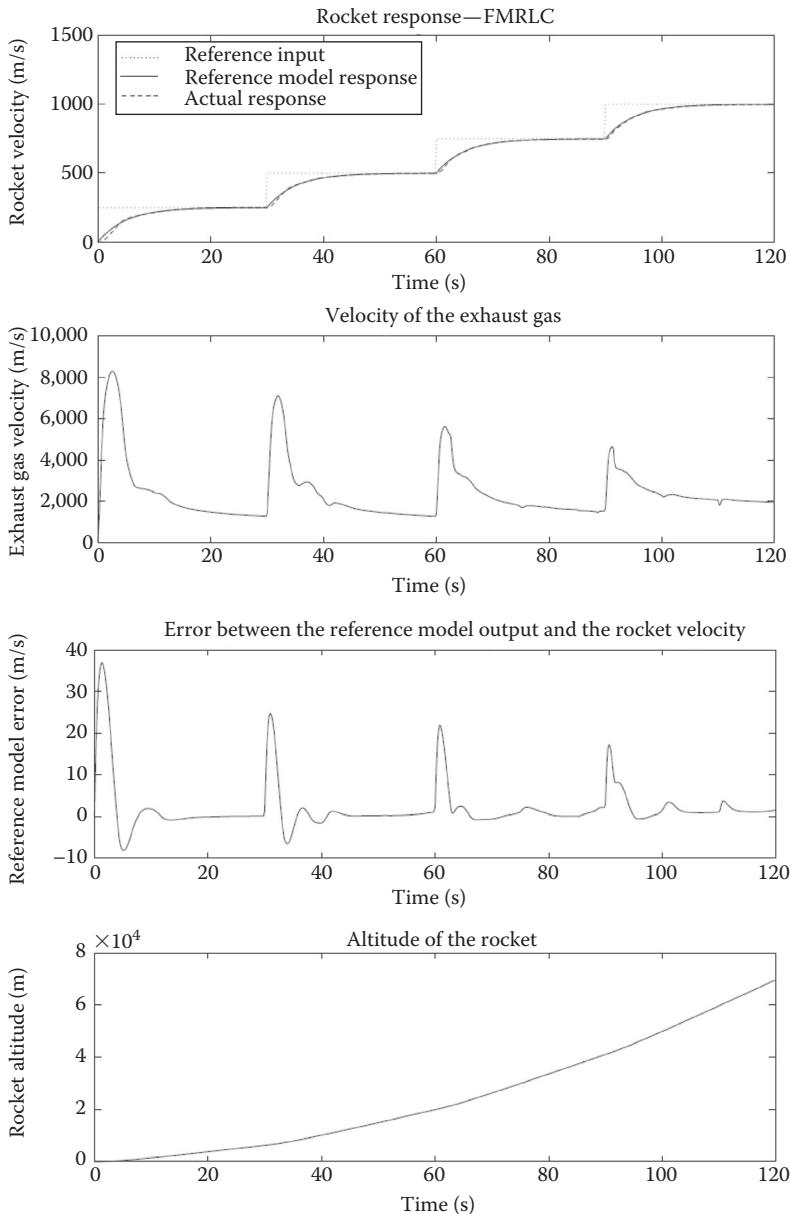


FIGURE 7.20 Simulation result for FMRLC control of the rocket system. (From Layne, J.R. and Passino, K.M., *J. Intelligent Fuzzy Syst.*, 4, 33, 1996. With permission.)

named as model-based fuzzy controller and has been studied extensively in academic research, where a system model can be used off-line during controller design or online as a part of the controller in its direct or inverse form.

7.2.1 FUZZY INVERSE CONTROL

Fuzzy inverse control is the simplest approach to designing a fuzzy controller for a nonlinear system with a known plant model, where the controller is directly obtained from the plant inverse model. For simplicity, consider a nonlinear single-input single-output (SISO) system without any delay from the input to the output. The system dynamics can be represented as a fuzzy system as

$$R: \text{IF } y(k) \text{ is } Y_0 \text{ AND } y(k-1) \text{ is } Y_1 \text{ AND } \dots \text{ AND } y(k-n_y+1) \text{ is } Y_{n_y-1} \\ u(k) \text{ is } U_0 \text{ AND } U(k-1) \text{ is } U_1 \text{ AND } \dots \text{ AND } u(k-n_u+1) \text{ is } U_{n_u-1}, \\ \text{THEN } y(k+1) \text{ is } Y \quad (7.49)$$

where

$u(k)$ is the system input at sampling time k

$y(k)$ is the system output

n_u and n_y are the orders of the system input and output, respectively

This fuzzy system can be denoted as

$$y(k+1) = f(\mathbf{x}(k), u(k)) \quad (7.50)$$

where

$\mathbf{x}(k)$ is a vector of the current and the previous system outputs, and the previous system inputs, i.e., $\mathbf{x}(k) = [y(k), y(k-1), \dots, y(k-n_y+1), u(k-1), \dots, u(k-n_u+1)]$

$f(\cdot)$ is the system nonlinear input–output mapping represented by the fuzzy system

This fuzzy model predicts the system output value $y(k+1)$ at the next sampling time. Consequently, the fuzzy inverse model can be designed in such a way that, given the desired system output at the next sampling time as $y_d(k+1)$, the required system input $u(k)$ can be obtained as

$$u(k) = f^{-1}(\mathbf{x}(k), y_d(k+1)) \quad (7.51)$$

The overall open-loop fuzzy inverse control system is shown in Figure 7.21.

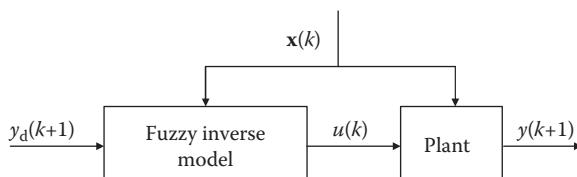


FIGURE 7.21 Fuzzy inverse control system.

As any other open-loop control schemes, the system static and dynamic characteristics must be well known and accurate. This inverse model-based control approach can only be applied to stable systems with minimum phase so that the inverted system dynamics is also stable. It only works without any disturbance, modeling error, and system constraint. If the sampling time is set too long, the system response of the fuzzy inverse controller may exhibit considerable delay.

This fuzzy inverse control method has been proposed by a number of researchers, such as Pedrycz (1993), Verbruggen and Babuška (1999), and Verbruggen et al. (1999). The majority of approaches result in generating a fuzzy relation rather than a fuzzy control rule base (Harris et al., 1993). In most cases, only an approximate fuzzy inverse relation can be derived. Consider the forward fuzzy relation equation:

$$A \circ R(X, Y) = B \quad (7.52)$$

where

A is a fuzzy set in X

$R(X, Y)$ is a binary fuzzy relation on the Cartesian product space $X \times Y$

The set-relation composition of A and R results in another fuzzy set B in Y , whose MF is derived as

$$\mu_B(y) = \mu_{A \circ R}(y) = \max_{x \in X} \min[\mu_A(x), \mu_R(x, y)] \quad (7.53)$$

Then the fuzzy inverse problem can be stated as follows: Given a fuzzy relation R and a fuzzy set B , determine the fuzzy set A , such that $A \circ R = B$. This solution exists if the inverse composition $A = B \circ R^{-1}$ can be found.

DEFINITION 7.1 (Lin and Lee, 1996)

For two real numbers a and b ($a, b \in [0, 1]$), α -operator is defined as

$$a \alpha b = \begin{cases} 1 & \text{if } a \leq b \\ b & \text{if } a > b \end{cases} \quad (7.54)$$

In the same principle, the α -operation of two fuzzy sets A and B forms a fuzzy relation $A \overset{\alpha}{\leftrightarrow} B$ in the Cartesian space $X \times Y$ as

$$\mu_{A \overset{\alpha}{\leftrightarrow} B}(x, y) = \mu_A(x) \alpha \mu_B(y) = \begin{cases} 1 & \text{if } \mu_A(x) \leq \mu_B(y) \\ \mu_B(y) & \text{if } \mu_A(x) > \mu_B(y) \end{cases} \quad (7.55)$$

the α -operation of the fuzzy relation R and the fuzzy set B is denoted as $R \overset{\alpha}{\leftarrow} B$ and is defined as

$$\mu_{R \overset{\alpha}{\leftarrow} B}(x) = \min_{y \in Y} [\mu_R(x, y) \alpha \mu_B(y)] \quad (7.56)$$



THEOREM 7.1 (Lin and Lee, 1996)

If there is no condition that the following inequality holds:

$$\max_{x \in X} \mu_R(x, y) < \mu_B(y) \quad \text{for some } y \in Y \quad (7.57)$$

then the fuzzy inverse relation exists and the largest fuzzy result A that satisfies $A \circ R = B$ is \hat{A} with the property as

$$\hat{A} = R \xrightarrow{\alpha} B \quad (7.58)$$

and its MF is

$$\mu_{\hat{A}}(x) = \mu_{R \xrightarrow{\alpha} B}(x) = \min_{y \in Y} [\mu_R(x, y) \alpha \mu_B(y)] \quad (7.59)$$

■

Example 7.6 (Lin and Lee, 1996)

For a fuzzy input $A = 0.2/x_1 + 0.8/x_2 + 1/x_3 = (0.2 \ 0.8 \ 1)$ and a fuzzy relation

$$R(X, Y) = \begin{pmatrix} 0.7 & 1 & 0.4 \\ 0.5 & 0.9 & 0.6 \\ 0.2 & 0.6 & 0.3 \end{pmatrix}, \text{ the fuzzy output } B \text{ can be derived as}$$

$$\begin{aligned} B &= A \circ R \\ &= (0.2 \ 0.8 \ 1) \circ \begin{pmatrix} 0.7 & 1 & 0.4 \\ 0.5 & 0.9 & 0.6 \\ 0.2 & 0.6 & 0.3 \end{pmatrix} \\ &= (0.2 \vee 0.5 \vee 0.2 \ 0.2 \vee 0.8 \vee 0.6 \ 0.2 \vee 0.6 \vee 0.3) \\ &= (0.5 \ 0.8 \ 0.6) \\ &= 0.5/y_1 + 0.8/y_2 + 0.6/y_3 \end{aligned} \quad (7.60)$$

Next, suppose the fuzzy relation R and the fuzzy set B are given, the largest fuzzy set A that satisfies $A \circ R = B$ is \hat{A} as

$$\begin{aligned} \hat{A} &= R \xrightarrow{\alpha} B \\ &= \begin{pmatrix} 0.7 & 1 & 0.4 \\ 0.5 & 0.9 & 0.6 \\ 0.2 & 0.6 & 0.3 \end{pmatrix} \xrightarrow{\alpha} \begin{pmatrix} 0.5 \\ 0.8 \\ 0.6 \end{pmatrix} \\ &= \begin{pmatrix} 0.5 \wedge 0.8 \wedge 1 \\ 1 \wedge 0.8 \wedge 1 \\ 1 \wedge 1 \wedge 1 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.8 \\ 1 \end{pmatrix} \\ &= 0.5/x_1 + 0.8/x_2 + 1/x_3 \end{aligned} \quad (7.61)$$

It is obvious that

$$\begin{aligned}
\hat{A} \circ R &= (0.5 \quad 0.8 \quad 1) \circ \begin{pmatrix} 0.7 & 1 & 0.4 \\ 0.5 & 0.9 & 0.6 \\ 0.2 & 0.6 & 0.3 \end{pmatrix} \\
&= (0.5 \vee 0.5 \vee 0.2 \quad 0.5 \vee 0.8 \vee 0.6 \quad 0.4 \vee 0.6 \vee 0.3) \\
&= (0.5 \quad 0.8 \quad 0.6) \\
&= 0.5/\gamma_1 + 0.8/\gamma_2 + 0.6/\gamma_3 \\
&= B
\end{aligned} \tag{7.62}$$

and

$$A = (0.2 \quad 0.8 \quad 1) \subseteq \hat{A} = (0.5 \quad 0.8 \quad 1) \tag{7.63}$$

7.2.2 FUZZY INVERSE CONTROL FOR A SINGLETON FUZZY MODEL

For a singleton fuzzy model, Babuška (1995, 1998) and Baranyi et al. (1998) developed an exact fuzzy inversion approach. Consider a nonlinear SISO dynamic system in the form of a fuzzy singleton model as

$$\begin{aligned}
R: \text{IF } &y(k) \text{ is } A_1 \text{ AND } y(k-1) \text{ is } A_2 \text{ AND } y(k-n+1) \text{ is } A_n \\
&u(k) \text{ is } B_1 \text{ AND } u(k-1) \text{ is } B_2 \text{ AND } u(k-m+1) \text{ is } B_m, \\
\text{THEN } &y(k+1) \text{ is } c
\end{aligned} \tag{7.64}$$

where

A_1, \dots, A_n and B_1, \dots, B_m are fuzzy sets
 c is a fuzzy singleton

Define an artificial vector $\mathbf{x}(k) = [y(k), y(k-1), \dots, y(k-n+1), u(k-1), \dots, u(k-m+1)]$ in the multidimensional Cartesian product space $\mathbf{X} = A_1 \times \dots \times A_n \times B_2 \times \dots \times B_m$. For simplicity, denote the fuzzy set $U = B_1$, and then the fuzzy rule in Equation 7.64 can be simplified as

$$R: \text{IF } \mathbf{x}(k) \text{ is } \mathbf{X} \text{ AND } u(k) \text{ is } U, \text{ THEN } y(k+1) \text{ is } c \tag{7.65}$$

Suppose the number of fuzzy sets for $\mathbf{x}(k)$ is M and the number of fuzzy sets for $u(k)$ is N . For the i th fired fuzzy set \mathbf{X}_i and the j th fired fuzzy set U_j , the (ij) th fuzzy rule R_{ij} can be represented as

$$R_{ij}: \text{IF } \mathbf{x}(k) \text{ is } \mathbf{X}_i \text{ AND } u(k) \text{ is } U_j, \text{ THEN } y(k+1) \text{ is } c_{ij} \tag{7.66}$$

This rule base consists of $M \cdot N$ possible rules in total and the degree of fulfillment of each rule premise can be calculated as $\beta_{ij} = \mu_{X_i}[\mathbf{x}(k)] \cdot \mu_{U_j}[u(k)]$. The defuzzified output $y(k+1)$ is calculated as an average of the consequences c_{ij} weighted by the normalized degrees of fulfillment β_{ij} as

$$y(k+1) = \frac{\sum_{i=1}^M \sum_{j=1}^N \beta_{ij} \cdot c_{ij}}{\sum_{i=1}^M \sum_{j=1}^N \beta_{ij}} = \frac{\sum_{i=1}^M \sum_{j=1}^N \mu_{X_i}[\mathbf{x}(k)] \cdot \mu_{U_j}[u(k)] \cdot c_{ij}}{\sum_{i=1}^M \sum_{j=1}^N \mu_{X_i}[\mathbf{x}(k)] \cdot \mu_{U_j}[u(k)]} \quad (7.67)$$

The objective of inverting the fuzzy model $y(k+1) = f(\mathbf{x}(k), u(k))$ is to find the system inverse mapping such that $u(k) = f^{-1}(\mathbf{x}(k), y_d(k+1))$, where $y_d(k+1)$ represents the desired system output at the next sampling time. Based on this principle, the fuzzy rule in Equation 7.66 is rewritten as

$$R_{ij}: \text{IF } u(k) \text{ is } U_j, \text{ THEN } [\text{IF } \mathbf{x}(k) \text{ is } \mathbf{X}_i, \text{ THEN } y(k+1) \text{ is } c_{ij}] \quad (7.68)$$

With the notation $c_j = \frac{\sum_{i=1}^M \mu_{X_i}[\mathbf{x}(k)] \cdot c_{ij}}{\sum_{i=1}^M \mu_{X_i}[\mathbf{x}(k)]}$, the fuzzy rules containing identical fuzzy sets U_j for the model input $u(k)$ are aggregated and further denoted as

$$R_j: \text{IF } u(k) \text{ is } U_j, \text{ THEN } y(k+1) \text{ is } c_j \quad (7.69)$$

This SISO dynamic fuzzy singleton system is invertible if the kernel of each U_j is a single point ($|\ker(U_j)| = 1$) and the fuzzy rule base is monotonic ($\ker(U_l) \geq \ker(U_m) \rightarrow c_l \geq c_m$ or $\ker(U_l) \leq \ker(U_m) \rightarrow c_l \leq c_m, \forall l, m \neq m$). Define a triangular fuzzy MF C_j for the desired system output $y_d(k+1)$, where $\ker(C_j) = c_j$ and $\sum_{j=1}^N \mu_{C_j}[y_d(k+1)] = 1$. Define a singleton fuzzy MF u_j for the variable $u(k)$ with $u_j = \ker(U_j)$. With Mamdani's min implication, the fuzzy inverse model can be constructed as

$$\text{IR}_j: \text{IF } y_d(k+1) \text{ is } C_j, \text{ THEN } u(k) \text{ is } u_j \quad (7.70)$$

where each triangular fuzzy MF can be represented as

$$\begin{aligned} \mu_{C_1}[y(k+1)] &= \max\left(0, \min\left(1, \frac{c_2 - y_d(k+1)}{c_2 - c_1}\right)\right) \\ &\vdots \\ \mu_{C_j}[y(k+1)] &= \max\left(0, \min\left(\frac{y_d(k+1) - c_{j-1}}{c_j - c_{j-1}}, \frac{c_{j+1} - y_d(k+1)}{c_{j+1} - c_j}\right)\right) \quad (7.71) \\ &\vdots \\ \mu_{C_N}[y(k+1)] &= \max\left(0, \min\left(\frac{y_d(k+1) - c_{N-1}}{c_N - c_{N-1}}, 1\right)\right) \end{aligned}$$

In this way, the required system input $u(k)$ can be directly calculated to achieve the desired system output $y_d(k+1)$ at the next sampling instant as

$$u(k) = \frac{\sum_{i=1}^M \mu_{C_j}[y_d(k+1)] \cdot u_i}{\sum_{i=1}^M \mu_{C_j}[y_d(k+1)]} \quad (7.72)$$

As stated previously, since no feedback information from the system output is extracted and utilized in this fuzzy inverse control system scheme, stable control performance can only be guaranteed for stable and minimum phase systems. Nonminimum phase systems will be destabilized by the fuzzy inverse model control structure, since it always attempts to drive the system to the desired goal directly. Fuzzy model predictive controller is an effective approach to deal with the nonminimum phase systems, which will be explained in detail in Section 7.2.3. A model-plant mismatch may cause a steady-state error at the system output, which can be compensated by incorporating into feedback control schemes, that is, the fuzzy internal model control (IMC) scheme, which will be addressed in Section 7.2.4.

7.2.3 FUZZY MODEL-BASED PREDICTIVE CONTROL

In order to overcome the problems existing in the open-loop approach presented above, a time domain control method, a fuzzy model-based predictive control (FMBPC) technique, has been extensively studied especially in Europe. Instead of using system inverse model to calculate control action, it utilizes an explicit fuzzy forward model to describe the system nonlinearity and the required control effort is derived and updated online at each sampling time through an optimization approach. In this control scheme, a fuzzy forward model is used to predict system output values at the future sampling time instants over a prediction horizon H_p as $\hat{y} = [\hat{y}(k+1), \dots, \hat{y}(k+H_p)]^T$. The future control actions over a control horizon H_c , $\mathbf{u} = [u(k), \dots, u(k+H_c-1)]^T$, are calculated by minimizing a given objective function as

$$J = \sum_{i=1}^{H_p} \alpha_i [y_d(k+i) - \hat{y}(k+i)]^2 + \sum_{i=1}^{H_c} \beta_i \Delta u(k+i-1)^2 \quad (7.73)$$

which is intended to maintain the predicted system output \hat{y} as close as possible to the desired output signal y_d . The control signal after the control horizon H_c remains constant as $u(k+H_c-1) = u(k+H_c) = \dots = u(k+H_p)$. The basic principle of this control scheme is shown in Figure 7.22 to illustrate the relationship of the prediction horizon H_p and the control horizon H_c .

The predicted output values $\hat{y} = [\hat{y}(k+1), \dots, \hat{y}(k+H_p)]^T$ depend on the current state variables and the future control signals $\mathbf{u} = [u(k), \dots, u(k+H_c-1)]^T$. The first item in the objective function (Equation 7.73) is to minimize the system output error over the prediction horizon H_p and the second item represents a penalty on the control effort (related, for instance, to control energy) over the control horizon H_c . The parameters α_i and β_i denote the weighting factors of the output error signal and

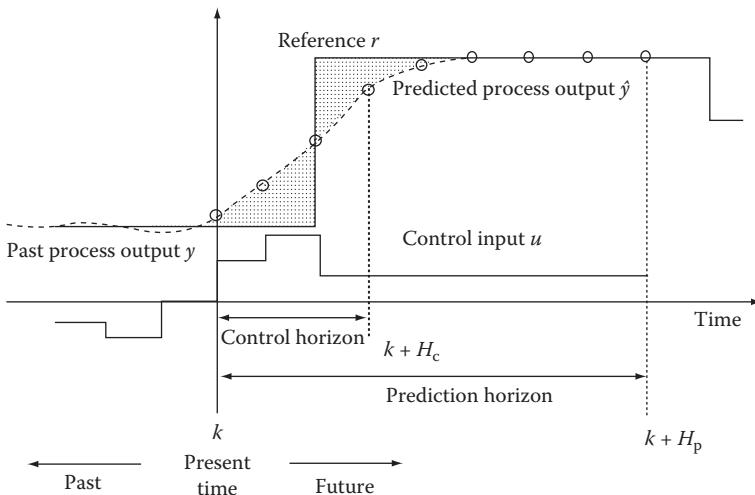


FIGURE 7.22 Basic principle of the FMBPC. (From Babuška, R., *Fuzzy Modeling for Control*, Kluwer Academic Publishers, Boston, MA, 1998. With permission.)

the control effort over the horizon time zones. Other plant constraint specifications, such as the degree and rate constraints of the control action or other system variables, can also be included in as a part of the objective function for optimization calculation. In general, an explicit solution is obtainable for a quadratic objective function with a linear time-invariant system model without any system constraint. In other cases, iterative numerical optimization calculation will be necessary to acquire the mathematical solution of the control effort.

During the control operation, at each sampling time, only the current control action $u(k)$ is applied to the plant, the output $y(k+1)$ is obtained from the optimization calculation and the horizons move forward with the updated value. In general, the control action $u(k+1)$ calculated at the time instant $k+1$ is different from the one calculated at the time instant k , since more up-to-date information about the system is used for the optimization computation.

Fuzzy model-based predictive control can be viewed as a generalization of the fuzzy inverse control method, considering $H_p = H_c = 1$ and the vector $\beta = 0$ in the objective function (Equation 7.73). In other words, when no system constraint or penalty on the control effort is considered, the one-step-ahead FMBPC strategy is equivalent to the fuzzy inverse control method, where the inverse is computed by means of objective function minimization calculation.

The schematic diagram of a typical FMBPC system is shown in Figure 7.23, where the fuzzy model acts as the numerical predictor of the system. Due to the explicit usage of the system fuzzy model and the numerical optimization calculation, the FMBPC method can effectively realize multivariable optimal control with various system nonlinearity and constraints. Furthermore, the system fuzzy model can be adapted online to minimize the difference between the modeled system output and

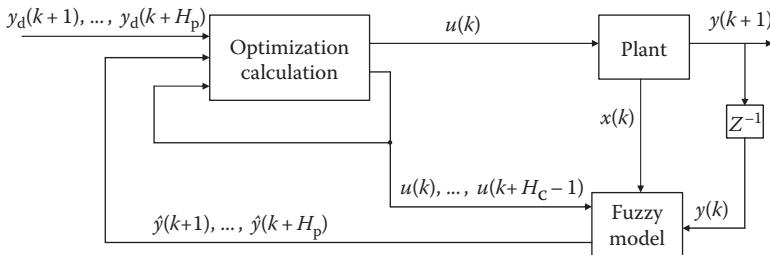


FIGURE 7.23 Fuzzy model-based predictive control.

the actual output, so this control scheme is able to control time-varying systems as well. This control scheme is easy to understand for plant engineers and operators with a limited knowledge of control theory and is suitable for various predefined control goals, such as set-point regulation, trajectory planning, process batch control, etc.

Even though some methods have been devised to simplify the optimization calculation, the computational load is still a critical issue in real-time applications for complex nonlinear systems, which will extend transient response for fast systems due to short sampling time and extensive computation load. Compared to Mamdani relational fuzzy models, Takagi–Sugeno fuzzy model representations are used more commonly in this model-based predictive control strategy due to the easy model establishment by decomposing the complex nonlinear modeling problem into a set of simpler linear models valid within certain operating regimes. Furthermore, standard identification techniques such as least-squares method can be easily applied to determine the system parameters.

Example 7.7 (Mahfouf et al., 2000)

This simulation example involves the application of the proposed fuzzy model-based predictive controller to a continuous stirred tank reactor (CSTR) as a test bed. The system schematic diagram is shown in Figure 7.24. The reactants flow into the vessel and the products are taken out. The vessel is stirred constantly to keep the contents as uniform as possible in composition and temperature. In often times, it is provided with an internal or external means of heat exchange. The overall system model consists of two nonlinear ordinary differential equations based on the mass and energy balance requirements as shown in Equations 7.74 and 7.75.

Mass balance:

$$\frac{dC_A}{dt} = \frac{F_i}{V}(C_{Ai} - C_A) - k_0 \exp\left(-\frac{E}{RT}\right) C_A \quad (7.74)$$

Energy balance:

$$\frac{dT}{dt} = \frac{F_i}{V}(T_i - T) - \frac{\Delta H_R}{c_p \rho} k_0 \exp\left(-\frac{E}{RT}\right) C_A - \frac{Q}{c_p \rho V} \quad (7.75)$$

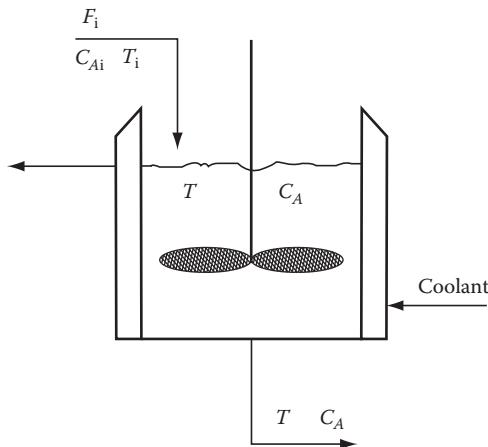


FIGURE 7.24 Continuous stirred tank reactor. (From Mahfouf, M., Linkens, D.A., and Abbot, M.F., *Trans. Inst. Chem. Eng. Chem. Eng. Res. Design*, 78, 590, 2000. With permission.)

where

C_A is the concentration of A in reactor

C_{Ai} is the concentration of A in inlet stream

F_i is the inlet-flow rate

V is the volume of reactor

k_0 is the reaction rate constant

E is the activation energy

R is the gas constant

T is the reactor temperature

T_i is the temperature of inlet stream

ΔH_R is the reaction heat

c_p is the specific heat

ρ is the density

Q is the heat removed from the tank reactor system

For this system with $Q(t)$ as input and $C_A(t)$ as output, it can be represented as a Takagi–Sugeno fuzzy model as

R^i : IF $C_A(t)$ is B^i ,

$$\text{THEN } \hat{C}_A(t+1) = a_1^i C_A(t) + a_2^i C_A(t-1) + b_1^i Q(t) + b_2^i Q(t-1) + k_i \quad (7.76)$$

where

$i = 1, \dots, p$ represents the number of fuzzy rules

$\hat{C}_A(t+1)$ is the one-step-ahead model prediction result

a_1^i, a_2^i, b_1^i , and b_2^i are real-valued parameters

k_i is the offset term

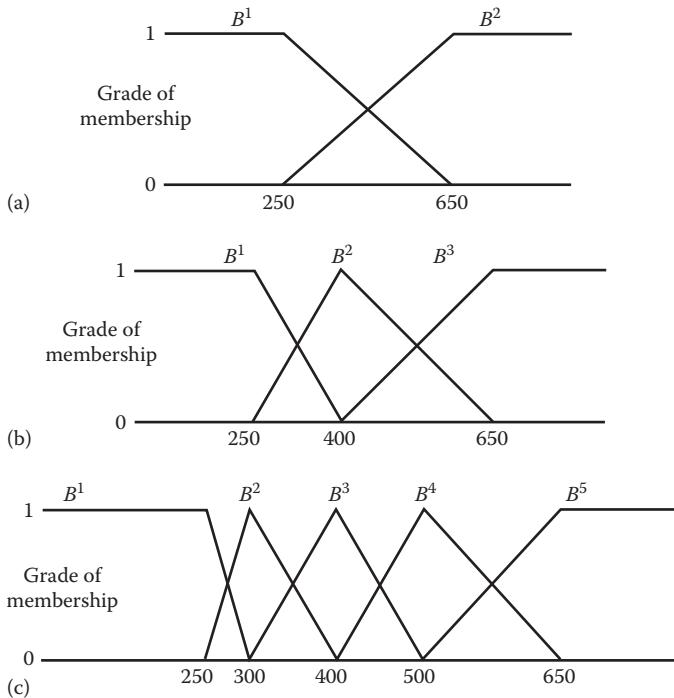


FIGURE 7.25 Fuzzy partitioning of the input space: (a) two-partition fuzzy model, (b) three-partition fuzzy model, and (c) five-partition fuzzy model. (From Mahfouf, M., Linkens, D.A., and Abbot, M.F., *Trans. Inst. Chem. Eng. Chem. Eng. Res. Design*, 78, 590, 2000. With permission.)

Three different fuzzy partitions of the input space are adopted in the simulation, such as two-partition, three-partition, and five-partition, as shown in Figure 7.25.

The objective function is formulated as

$$J = \sum_{j=N_1}^{N_2} [y_d(t+j) - P(z^{-1})\dot{y}(t+j)]^2 + \sum_{i=1}^{N_u} \lambda(j) \Delta u(t+j-1)^2 \quad (7.77)$$

where

$N_1 = 1$ is the minimum prediction horizon

$N_2 = 10$ is the maximum prediction horizon

N_u is the control horizon

$P(z^{-1})$ is the specification in the model-following context

$\lambda(j)$ is the weighting factor of the control effort; in this example, $\lambda(j) = 0$

Figure 7.26 shows two simulation results based on the proposed fuzzy model-based predictive controller with a two-partition fuzzy model and the control horizon is specified as $N_u = 1$. The solid line is the case when the model-following polynomial is set as $P(z^{-1}) = 10(1 - 0.9z^{-1})$. The system response tracks the

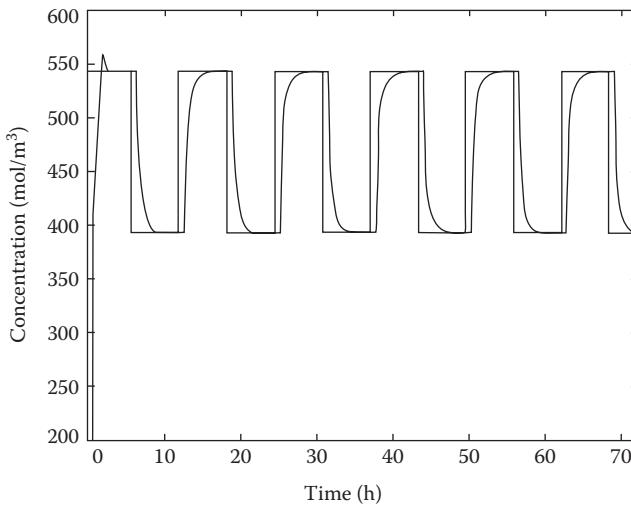


FIGURE 7.26 FMBPC result (solid line: $P(z^{-1}) = 10(1 - 0.9z^{-1})$; dashed line: $P(z^{-1}) = 3.33(1 - 0.7z^{-1})$). (From Mahfouf, M., Linkens, D.A., and Abbot, M.F., *Trans. Inst. Chem. Eng. Chem. Eng. Res. Design*, 78, 590, 2000. With permission.)

set-point value effectively although sluggish, which is primarily because of the slow dynamics of the model-following polynomial. In the other case, a faster dynamics is adopted as $P(z^{-1}) = 3.33(1 - 0.7z^{-1})$ and the system output response is illustrated in dashed line, which displays a faster tracking performance, but at the expense of producing larger overshoots during the first two step iterations.

Figure 7.27 compares the control performance with three-partition fuzzy model and two-partition fuzzy model, when the model-following polynomial is set as $P(z^{-1}) = 10(1 - 0.9z^{-1})$. The system output response with three-partition fuzzy model is shown in solid line, which is obviously slower in transient time than the response with two-partition fuzzy model. In general, extra partitions in the fuzzy modeling will introduce more sluggishness in the output response due to the additional computation load.

7.2.4 FUZZY INTERNAL MODEL CONTROL

For a fuzzy inverse control system described in Section 7.2.1, some discrepancy may exist between the actual plant and the model, and different forms of disturbances may act on the system, both of which cannot be compensated for due to the open-loop structure. In such conditions, the fuzzy IMC scheme is suitable to overcome these problems in practical applications. IMC system is named since the control structure contains the internal model of the controlled plant, which is very useful especially for control of nonlinear systems. Fuzzy logic and neural networks are two powerful tools to model system nonlinearity and it is easy to define their structures and parameters based on input/output measurements. Hence, they have been used extensively in the design of fuzzy/neural internal model controller (Piegat, 2001).

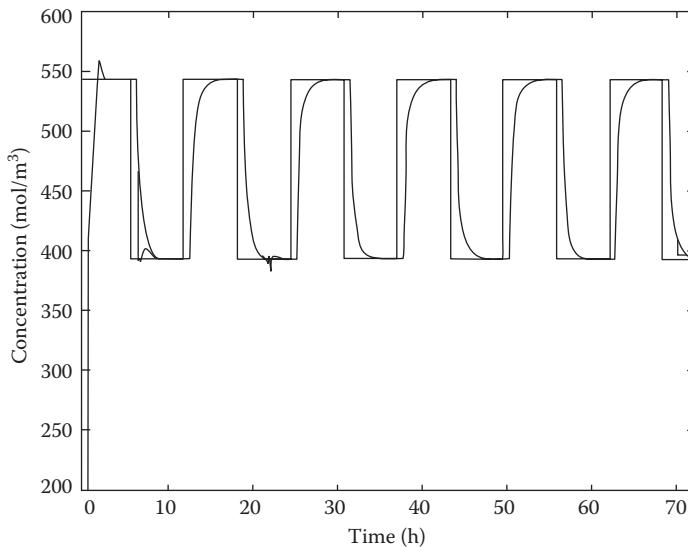


FIGURE 7.27 FMBPC result (solid line: with three-partition fuzzy model; wavy line: with two-partition fuzzy model). (From Mahfouf, M., Linkens, D.A., and Abib, M.F., *Trans. Inst. Chem. Eng. Chem. Eng. Res. Design*, 78, 590, 2000. With permission.)

The idea of constructing an internal model controller is very intuitive. Consider the open-loop fuzzy inverse control system in Figure 7.28, where the controller ($C = P^{-1}$) and the plant P are both stable, and no disturbance exists to influence the plant. In such a perfect condition, the system output y is equal to the desired output value y_d as $y = y_d$. However, in practical applications, various types of disturbances may exist and affect the real plant operation as shown in Figure 7.29. Thus, the introduction of a feedback loop will be necessary to compensate for various disturbances and thus the IMC system is constructed as shown in Figure 7.30, where the combined disturbance effect is denoted as $d = d_1G + d_2$ from Figure 7.29 to Figure 7.30. The purpose of the model P^* working in parallel with the plant is to subtract the effect of the control action u from the system output y .

The fuzzy IMC system can be transformed consecutively to an equivalent classical control structure as shown from Figure 7.31a through c, where the plant model P^* becomes the internal element of the controller R and the resultant controller R in Figure 7.31c is calculated as $R = \frac{C}{1 - CP^*}$.

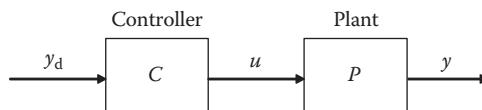


FIGURE 7.28 Open-loop fuzzy inverse control system (without any disturbance $\rightarrow y = y_d$). (From Piegat, A., *Fuzzy Modeling and Control*, Physica-Verlag, NY, 2001. With permission.)

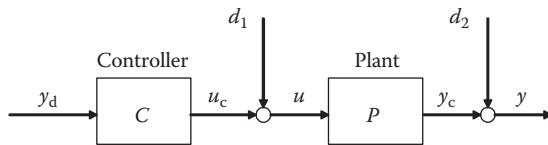


FIGURE 7.29 Open-loop fuzzy inverse control system (with disturbances $\rightarrow y \neq y_d$). (From Pieglat, A., *Fuzzy Modeling and Control*, Physica-Verlag, NY, 2001. With permission.)

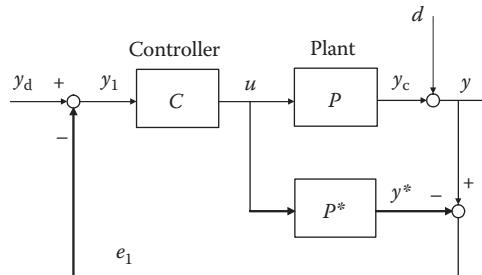


FIGURE 7.30 Fuzzy IMC system. (From Pieglat, A., *Fuzzy Modeling and Control*, Physica-Verlag, NY, 2001. With permission.)

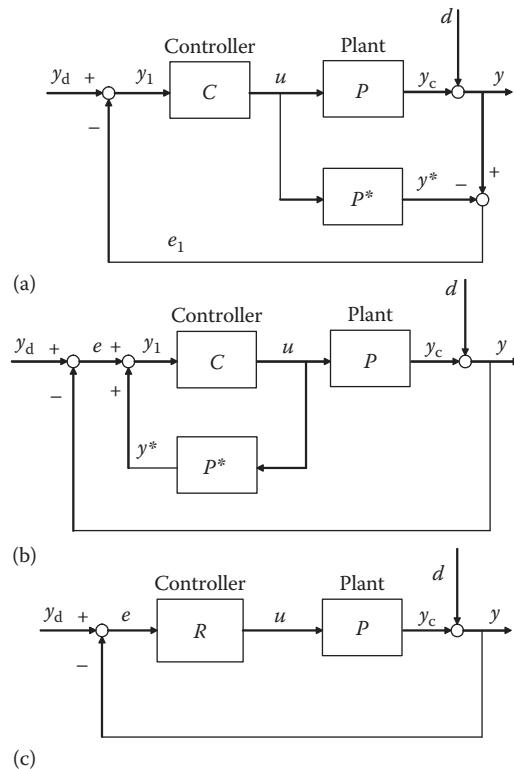


FIGURE 7.31 Consecutive transformation of a fuzzy IMC system. (a) A typical fuzzy IMC system, (b) internal feedback signal P^* to the controller C , and (c) resultant controller R . (From Pieglat, A., *Fuzzy Modeling and Control*, Physica-Verlag, NY, 2001. With permission.)

Remarks:

1. If the plant model P^* is accurate ($P^* = P$) and no disturbance exists ($d = 0$), then the feedback signal $e_1 = 0$. In such a circumstance, the closed-loop feedback control system operates the same as the open-loop control system shown in [Figure 7.28](#). On the other hand, if disturbances exist ($d \neq 0$), then the feedback signal $e_1 = d$. With the feedback-loop configuration, the fuzzy IMC system is hence able to cancel the effect of unmeasured output-additive disturbances.
2. If the plant model P^* is inaccurate ($P^* \neq P$) and disturbances exist ($d \neq 0$), then the feedback signal $e_1 = (P - P^*)u + d$. In other words, the feedback signal e_1 depends on both the modeling error $P - P^*$ and the disturbances d .
3. The fuzzy IMC structure can only be applied to minimum phase plants. For nonminimum phase plants, the zeros of the plant model P^* are positive and then the poles of P^{*-1} are unstable. Thus, the controller C as well as the whole IMC system would be unstable.
4. If the plant model P^* is accurate ($P^* = P$), both the plant P and the controller C are stable, then the resultant overall IMC system is stable. This facilitates the task in examining the stability of the complete system by studying the property of each individual component element.
5. If the controller is an exact inverse of the plant ($C = P^{-1}$) and the closed-loop system is stable, then the overall fuzzy IMC system has the perfect control property with error-free output signal as $y(k) = y_d(k)$, $\forall k$.
6. If the plant is unstable, two additional feedback loops can be introduced into the fuzzy IMC system structure to stabilize both the plant P and the model P^* before closing the system feedback loop as shown in [Figure 7.32](#).

Example 7.8 (Baranyi et al., 1998)

In this example, the fuzzy internal model controller is applied to a nonlinear fermenter with pressure dynamic characteristics. The fermenter has a volume of 40 L and is normally filled with 25 L water. At the bottom of the tank, air is fed into the water at a constant rate. This nonlinear chemical process has one input $u(k)$, the outlet valve position at the top of the tank, and one output $y(k)$, the air pressure above the water level in the head space. A fuzzy identification approach is used to establish the system model due to the difficulty in deriving its accurate physical model. Experiments are designed to collect the identification data and validation data. The sampling time is 5 s. A nonlinear first-order regression fuzzy model $y(k+1) = f(y(k), u(k))$ is constructed and the antecedent MFs are shown in [Figure 7.33](#). The kernels of the MFs are [1, 1.2, 1.6, 2.2] for the pressure $y(k)$ and [0, 54, 77, 100] for the valve position $u(k)$.

The constructed fuzzy model rule base consists of 16 rules as shown in [Table 7.11](#), where each entry represents the pair parameters $(b_{i,j}, s_{i,j})$ of the consequent triangular MF. The peak of the triangle $b_{i,j}$ is estimated by the least-squares method and the base length $s_{i,j}$ is optimized by the Levenberg–Marquardt method to minimize the mean squared error.

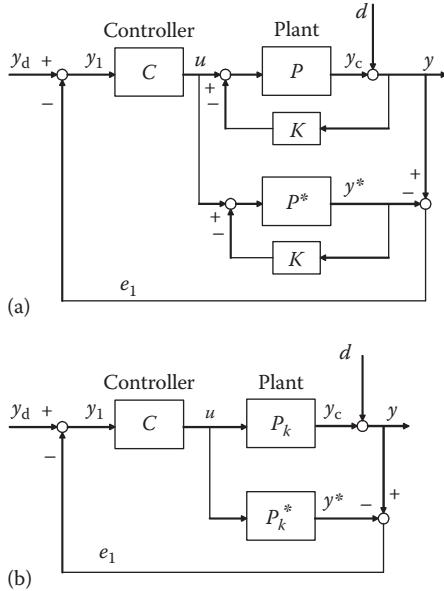


FIGURE 7.32 Fuzzy IMC system for an unstable plant: (a) complete structure and (b) simplified structure. (From Piegat, A., *Fuzzy Modeling and Control*, Physica-Verlag, NY, 2001. With permission.)

The obtained fuzzy model is tested with the validation data set and the results are illustrated in Figure 7.34, where the dashed line represents the model output and the solid line is the actual measured output. It is obvious that the established fuzzy model approximates the real process with satisfactory accuracy.

Next, a fuzzy IMC system is constructed as shown in Figure 7.35 to deal with the output-additive disturbances and process time-varying changes. The fuzzy model executed in parallel with the process predicts the process output one step ahead $\hat{y}(k+1)$. In the ideal operating condition without any disturbance or plant-model mismatch, the output model output will be the same as the process output as $y(k+1) = \hat{y}(k+1)$. The feedback signal $e(k+1)$ is zero and the process output $y(k+1)$ perfectly follows the reference signal $r(k+1)$. On the other hand, in order to simulate a more realistic situation, three types of disturbances are considered in the simulation:

1. Output-additive measurement noise d_1 , zero mean, normally distributed with a variance of 2.5×10^5 Pa
2. Output-additive step disturbance $d_2 = -0.1 \times 10^5$ Pa, which is applied at the time $t = 125$ s
3. Model-plant mismatch as a random change in the consequence parameters ($0 \leq \Delta b_{ij} \leq 0.1 \times 10^5$ Pa, for $1 \leq i, j \leq 16$) at the time $t = 250$ s

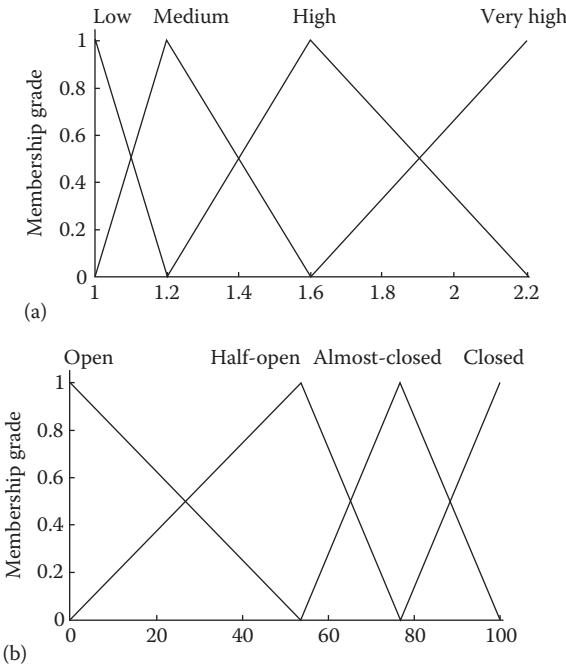


FIGURE 7.33 Antecedent MFs of the fuzzy model: (a) pressure (10^5 pa) and (b) valve position (% closed). (From Baranyi, P., Bavelaar, I.M., Babuška, R., Kóczy, L.T., Titli, A., and Verbruggen, H.B., *Int. J. Syst. Sci.*, 29, 711, 1998. With permission.)

In Figure 7.35, a first-order Butterworth feedback filter is introduced to damp the high-frequency component of signal and filter out the possible measurement noise as $f(k+1) = 0.51f(k) + 0.25e(k+1) + 0.25e(k)$ and the resultant output signal is shown in Figure 7.36, which justifies a good tracking property and a satisfactory disturbance rejection ability of the proposed fuzzy inverse model control scheme.

TABLE 7.11
Fuzzy Model Rule Base

		Valve Position $u(k)$			
$y(k+1)$		Open	Half-Open	Almost-Closed	Closed
Pressure $y(k)$	Low	(1.04, 0.08)	(1.05, 0.11)	(1.08, 0.08)	(1.11, 0.02)
	Medium	(1.13, 0.13)	(1.19, 0.11)	(1.23, 0.10)	(1.33, 0.09)
	High	(1.46, 0.14)	(1.51, 0.11)	(1.63, 0.10)	(1.79, 0.09)
	Very high	(1.80, 0.10)	(2.04, 0.13)	(2.12, 0.10)	(2.35, 0.10)

Source: Baranyi, P., Bavelaar, I.M., Babuška, R., Kóczy, L.T., Titli, A., and Verbruggen, H.B., *Int. J. Syst. Sci.*, 29, 711, 1998. With permission.

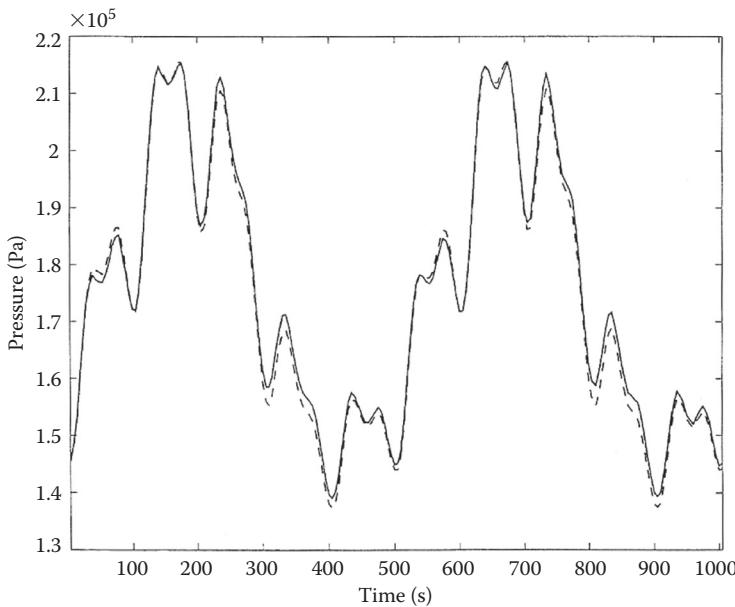


FIGURE 7.34 Fuzzy model validation. (From Baranyi, P., Bavelaar, I.M., Babuška, R., Kóczy, L.T., Titli, A., and Verbruggen, H.B., *Int. J. Syst. Sci.*, 29, 711, 1998. With permission.)

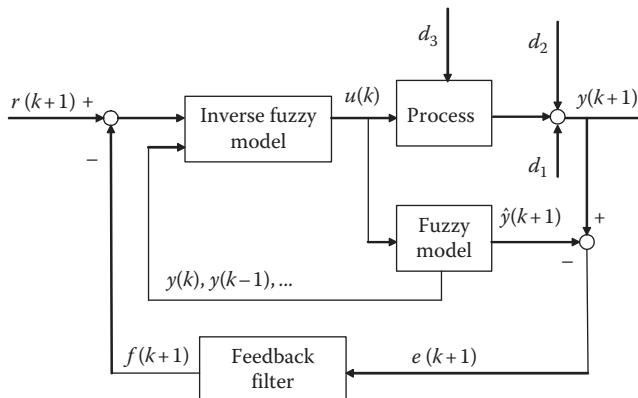


FIGURE 7.35 A fuzzy IMC system to compensate for unpredictable process variations. (From Baranyi, P., Bavelaar, I.M., Babuška, R., Kóczy, L.T., Titli, A., and Verbruggen, H.B., *Int. J. Syst. Sci.*, 29, 711, 1998. With permission.)

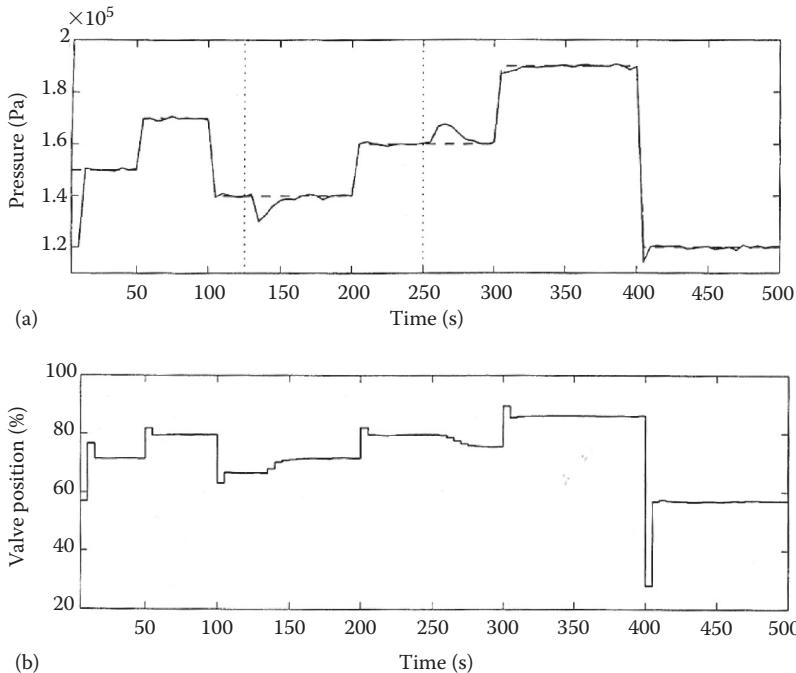


FIGURE 7.36 System simulation result: (a) desired and actual system output and (b) system input signal. (From Baranyi, P., Bavelaar, I.M., Babuška, R., Kóczy, L.T., Titli, A., and Verbruggen, H.B., *Int. J. Syst. Sci.*, 29, 711, 1998. With permission.)

REFERENCES

- Åström, K. and Wittenmark, B., *Adaptive Control*, Addison-Wesley Publishing Company, Reading, MA, 1989.
- Babuška, R., *Fuzzy Modeling for Control*, Kluwer Academic Publishers, Boston, MA, 1998.
- Babuška, R., Sousa, J., and Verbruggen, H.B., Model-based design of fuzzy control systems, *Proceedings of the International Conference EUFIT'95*, Vol. 1, Aachen, Germany, pp. 837–841, 1995.
- Baranyi, P., Bavelaar, I.M., Babuška, R., Kóczy, L.T., Titli, A., and Verbruggen, H.B., A method to invert a linguistic fuzzy model, *International Journal of Systems Science*, 29(7): 711–721, 1998.
- Gao, Z., Trautzsch, T.A., and Dawson, J.G., A stable self-tuning fuzzy logic control system for industrial temperature regulation, *35th IAS Annual Meeting, IEEE Industry Applications Society*, 2: 1232–1240, 2000.
- Golob, M., Decomposition of a fuzzy controller based on the inference break-up method, *Advances in Soft Computing: Engineering Design and Manufacturing*, Roy, R., Furuhashi, T., and Chawdhry, P.K. (Eds.), Springer, New York, pp. 215–227, 1999.
- Harris, C.J., Moore, C.G., and Brown, M., *Intelligent Control, Aspects of Fuzzy Logic and Neural Nets*, World Scientific, Singapore, 1993.
- Kwong, W.A. and Passino, K.M., Dynamically focused fuzzy learning control, *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 26(1): 53–74, 1996.

- Kwong, W.A., Passino, K.M., Laukonen, E.G., and Yurkovich, S. Expert supervision of fuzzy learning systems for fault tolerant aircraft control, *Proceeding of the IEEE, Special Issue on Fuzzy Logic in Engineering Applications*, 83(3): 466–483, 1995.
- Layne, J.R. and Passino, K.M., Fuzzy model reference learning control, *IEEE Conference on Control Applications*, Dayton, OH, pp. 686–691, 1992.
- Layne, J.R. and Passino, K.M., Fuzzy model reference learning control for cargo ship steering, *IEEE Control Systems Magazine*, 13(6): 23–34, 1993.
- Layne, J.R. and Passino, K.M., Fuzzy model reference learning control, *Journal of Intelligent and Fuzzy Systems*, 4(1): 33–47, 1996.
- Layne, J.R., Passino, K.M., and Yurkovich, S., Fuzzy learning control for anti-skid braking systems, *IEEE Conference on Decision and Control*, Tucson, AZ, pp. 2523–2528, 1992.
- Layne, J.R., Passino, K.M., and Yurkovich, S., Fuzzy learning control for anti-skid braking systems, *IEEE Transactions on Control System Technology*, 1(2): 122–129, 1993.
- Lee, C.K. and Pang, W.H., Rule-based adaptive control in a servomotor control system, *IEEE Conference Publication, Michael Faraday House*, 2(389): 1088–1093, 1994.
- Li, H., Chen, C.L.P., and Huang, H.-P., *Fuzzy Neural Intelligent Systems—Mathematical Foundation and the Applications in Engineering*, CRC Press LLC, Boca Raton, FL, 2001a.
- Li, W., Chang, X.G., Wahl, F.M., and Farrell, J., Tracking control of a manipulator under uncertainty by fuzzy P + ID controller, *Fuzzy Sets and Systems*, 122L: 125–137, 2001b.
- Lin, C.T. and Lee, C.S.G., *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*, Prentice Hall PTR, Upper Saddle River, NJ, 1996.
- Mahfouf, M., Linkens, D.A., and Abbod, M.F., Adaptive fuzzy TSK model-based predictive control using a CARIMA model structure, *Transactions of the Institute of Chemical Engineers, Chemical Engineering Research and Design—Special Topic Issue, Invited Paper*, 78(A): 590–596, 2000.
- Mamdani, E.H., Applications of fuzzy algorithms for control of simple dynamic plant, *Proceedings of the IEE*, 121(12): 1585–1588, 1974.
- Mamdani, E.H., Applications of fuzzy logic to approximate reasoning using linguistic systems, *Fuzzy Sets and Systems*, 26: 1182–1191, 1977.
- Mamdani, E.H. and Assilian, S., An experiment in linguistic synthesis with a fuzzy logic controller, *International Journal of Man-Machine Studies*, 7: 1–13, 1975.
- Moudgal, V.G., Kwong, W.A., Passino, K.M., and Yurkovich, S., Fuzzy learning control for a flexible-link robot, *IEEE Transactions on Fuzzy Systems*, 3(2): 199–210, 1995.
- Moudgal, V.G., Passino, K.M., and Yurkovich, S., Rule-based control for a flexible-link robot, *IEEE Transactions on Control Systems Technology*, 2(4): 392–405, 1994.
- Narendra, K. and Annaswamy, A., *Adaptive Systems*, Prentice Hall, Englewood Cliffs, NJ, 1989.
- Passino, K.M. and Yurkovich, S., *Fuzzy Control*, Addison-Wesley, Reading, MA, 1998.
- Pedrycz, W., *Fuzzy Control and Fuzzy Systems*, John Wiley & Sons, New York, 1993.
- Pedrycz, W., *Fuzzy Sets Engineering*, CRC Press, Boca Raton, FL, 1995.
- Piegat, A., *Fuzzy Modeling and Control*, Physica-Verlag, New York, 2001.
- Procyk, T.J. and Mamdani, E.H., A linguistic self-organizing process controller, *Automatica*, 15(1): 15–30, 1979.
- Verbruggen, H.B. and Babuška, R., *Fuzzy Logic Control Advances in Applications*, World Scientific Publishing Co. Pte. Ltd., Singapore, 1999.
- Verbruggen, H.B., Zimmermann, H.-J., and Babuška, R., *Fuzzy Algorithms for Control*, Kluwer Academic Publishers, Boston, MA, 1999.
- Vishnupad, P.S., Intelligent optimization and control of grinding processes with monitoring, Master Thesis, Purdue University, West Lafayette, IN, 1994.
- Zadeh, L.A., Fuzzy sets, *Information and Control*, 8: 338–353, 1965.
- Zadeh, L.A., Fuzzy algorithms, *Information and Control*, 12: 94–102, 1968.

- Zadeh, L.A., Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Transaction on Systems, Man, and Cybernetics*, 1: 28–44, 1973.
- Zadeh, L.A., The concept of a linguistic variable and its application to approximate reasoning, *Information Science*, 8: 199–257, 1975.
- Zumbrue, J. and Passino, K.M., A case study in intelligent control for a process control experiment, *IEEE International Symposium on Intelligent Control*, Dearborn, MI, pp. 37–42, 1996.

8 Stability Analysis Method

The stability property of a control system is an important issue that must be considered in evaluating the system control performance. Since fuzzy controllers are essentially nonlinear and the dynamic behavior of a controlled system is not well known, the closed-loop dynamic behavior is usually very complex. Therefore, the stability of a fuzzy control system must be carefully investigated to guarantee the desired system performance in the presence of different variations and uncertainties. Up to now, various mathematical analysis and design techniques of fuzzy dynamic systems have been devised to ensure the nonlinear fuzzy control system stability. Among the existing methods, Lyapunov stability analysis and passivity theory are two important approaches that have been used extensively in this area.

8.1 LYAPUNOV STABILITY ANALYSIS

Lyapunov stability theory is one of the most useful approaches to studying the stability of nonlinear systems, which was introduced by the Russian mathematician Alexandre Mikhailovich Lyapunov in the late nineteenth century. This theory contains two methods to facilitate system stability analysis. The Lyapunov's direct method constructs a special scalar function of the system state variables and then determines the system's overall stability based on the time-variation mode of this scalar function. The Lyapunov's indirect method derives the local stability property of the nonlinear system around an equilibrium point from the examination on its local linearized approximation.

8.1.1 MATHEMATICAL PRELIMINARIES

Before analyzing fuzzy system stability, this section provides some definitions and notations related to a general nonlinear system and its equilibrium points, which are important concepts in studying system's Lyapunov stability property. As described by Slotine and Li (1991), consider a general nonlinear autonomous (time-invariant) dynamic system expressed in continuous domain as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \quad (8.1)$$

where

$\mathbf{x} = [x_1, \dots, x_n]^T \in \Re^n$ is the system state vector
 $\mathbf{f}(\cdot)$ is an n -dimensional nonlinear function vector

DEFINITION 8.1

A state \mathbf{x}^* is an equilibrium point of the system (Equation 8.1) if once $\mathbf{x}(t)$ is equal to \mathbf{x}^* , it remains equal to \mathbf{x}^* for all $t \geq 0$. In other words, the constant vector \mathbf{x}^* satisfies

$$0 = \mathbf{f}(\mathbf{x}^*) \quad (8.2)$$

Without loss of generality, by introducing a new state vector $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$, the system (Equation 8.1) can be transformed as

$$\dot{\bar{\mathbf{x}}} = \mathbf{f}(\bar{\mathbf{x}} + \mathbf{x}^*) \quad (8.3)$$

And $\bar{\mathbf{x}} = 0$ is the equilibrium point of the transformed system (Equation 8.3). Therefore, the study of the behavior of the original system (Equation 8.1) in the neighborhood of equilibrium point \mathbf{x}^* will derive the same results by investigating the manner of the transformed system (Equation 8.3) in the adjacency of the origin $\bar{\mathbf{x}} = 0$. ■

DENOTATION 8.1

\mathbf{B}_R : A spherical region for the state vector centered at the origin in state space as $\|\mathbf{x}\| < R$ ($R > 0$), where $\|\cdot\|$ denotes a norm of the vector, for example, an Euclidian norm.

S_R : The sphere itself in the state space as $\|\mathbf{x}\| = R$. ■

DEFINITION 8.2

The equilibrium state $\mathbf{x} = 0$ is stable (or Lyapunov stable) if for any $R > 0$, there exists $r(R) > 0$, such that if $\|\mathbf{x}(0)\| < r(R)$, then $\|\mathbf{x}(t)\| < R$ for all $t \geq 0$. Otherwise, the equilibrium point is unstable. The notation $r(R)$ means that r is a function of R . ■

DEFINITION 8.3

The equilibrium state $\mathbf{x} = 0$ is asymptotically stable if it is stable and additionally, there exists some $\delta > 0$, such that $\mathbf{x}(t) \rightarrow 0$ as $t \rightarrow \infty$ for $\|\mathbf{x}(0)\| < \delta$. The spherical ball \mathbf{B}_δ is called the domain of attraction of the equilibrium point, that is, the combination set of all points where trajectories initiated eventually converge to the origin.

An equilibrium point that is Lyapunov stable but not asymptotically stable is called marginally stable. ■

DEFINITION 8.4

The equilibrium state $\mathbf{x} = 0$ is exponentially stable if there exists two positive numbers α and λ such that

$$\|\mathbf{x}(t)\| \leq \alpha \|\mathbf{x}(0)\| e^{-\lambda t}, \quad \forall t > 0 \quad (8.4)$$

for all $\|\mathbf{x}(0)\| < r$. The positive number λ is called the rate of exponential convergence. ■

DEFINITION 8.5

The equilibrium state $\mathbf{x} = 0$ is globally asymptotically (or exponentially) stable if it is asymptotically (or exponentially) stable for any initial state $\mathbf{x}(0) \in \mathbb{R}^n$. ■

For complex nonlinear systems, it is usually difficult to solve the ordinary differential Equations 8.1 to find the system solution for any t and $\mathbf{x}(0)$. In such circumstances, Lyapunov's method is one of the most widely used methods in nonlinear control system analysis, which provides an effective approach to determining the stability property of the controlled system without solving the system differential equations.

8.1.2 LYAPUNOV'S DIRECT METHOD

Considering the general nonlinear autonomous system (Equation 8.1), if a scalar function $V(\mathbf{x})$ is continuously differentiable with respect to its arguments, then its derivative regarding to time t can be obtained by the chain rule as

$$\dot{V}(\mathbf{x}) = \frac{dV(\mathbf{x})}{dt} = \frac{\partial V}{\partial \mathbf{x}} \dot{\mathbf{x}} = \nabla V(\mathbf{x})^T \cdot \mathbf{f}(\mathbf{x}) \quad (8.5)$$

where $\nabla V(\mathbf{x}) = \left[\frac{\partial V}{\partial x_1}, \frac{\partial V}{\partial x_2}, \dots, \frac{\partial V}{\partial x_n} \right]^T$ is the gradient of V with respect to \mathbf{x} .

DEFINITION 8.6

A continuous function $V: \mathbb{R}^n \rightarrow \mathbb{R}$ is positive definite if $V(0) = 0$, and $V(\mathbf{x}) > 0$ for all $\mathbf{x} \neq 0$. A continuous function $V: \mathbb{R}^n \rightarrow \mathbb{R}$ is positive semidefinite if $V(\mathbf{0}) = 0$ and $V(\mathbf{x}) \geq 0$ for all \mathbf{x} . A continuous function $V: \mathbb{R}^n \rightarrow \mathbb{R}$ is negative definite if $-V$ is a positive definite function. A continuous function $V: \mathbb{R}^n \rightarrow \mathbb{R}$ is negative semidefinite if $-V$ is a positive semidefinite function. The positive-definiteness property implies that the function V has a unique minimum at the origin. ■

THEOREM 8.1

If in a spherical ball \mathbf{B}_R , the scalar function $V(\mathbf{x})$ is positive definite and has continuous partial derivatives, and if its time derivative along any state trajectory of system (Equation 8.1) is negative semidefinite, that is, $\dot{V}(\mathbf{x}) \leq 0$, then the function $V(\mathbf{x})$ is called a Lyapunov function for the nonlinear system (Equation 8.1).

1. Denote the origin $\mathbf{x}=0$ as an equilibrium point for the nonlinear system (Equation 8.1). If there exists a scalar function $V(\mathbf{x})$ in a spherical region \mathbf{B}_R with continuous first partial derivatives such that
 - $V(\mathbf{x})$ is positive definite (locally in \mathbf{B}_R)
 - $\dot{V}(\mathbf{x})$ is negative semidefinite (locally in \mathbf{B}_R)then the equilibrium point $\mathbf{x}=0$ is stable. If in addition, the derivative $\dot{V}(\mathbf{x})$ is locally negative definite in \mathbf{B}_R , then the origin $\mathbf{x}=0$ is asymptotically stable.
2. Denote the origin $\mathbf{x}=0$ as an equilibrium point for the nonlinear system (Equation 8.1). Assume that there exists a scalar function $V(\mathbf{x})$ of the states \mathbf{x} with continuous first partial derivatives such that
 - $V(\mathbf{x})$ is positive definite
 - $\dot{V}(\mathbf{x})$ is negative definite
 - $V(\mathbf{x}) \rightarrow \infty$ as $\|\mathbf{x}\| \rightarrow \infty$then the equilibrium point $\mathbf{x}=0$ is globally asymptotically stable. ■

Lyapunov's direct method can also be referred to as Lyapunov's second method. Unfortunately, up to now, there is no systematic method to design a suitable Lyapunov candidate function $V(\mathbf{x})$. Sometimes, it is just based on trial and error to find the Lyapunov function to have the above properties to guarantee the overall system stability. In these circumstances, Lyapunov's indirect method will be an alternative way to the straightforward proof of the local stability of the nonlinear system, which is explained in Section 8.1.3.

8.1.3 LYAPUNOV'S INDIRECT METHOD

Based on the viewpoint that a nonlinear system performs similarly to its local linearized approximation within a small operating range around the equilibrium point, the local stability property of a nonlinear system is therefore guaranteed by the so-called Lyapunov's indirect linearization method.

Consider the origin $\mathbf{x}=0$ as the equilibrium point for the nonlinear autonomous system in Equation 8.1. Assume that $\mathbf{f}(\mathbf{x})$ is continuously differentiable. Then the system dynamics can be expanded in Taylor's series as

$$\dot{\mathbf{x}} = \left. \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=0} \cdot \mathbf{x} + \mathbf{f}_{\text{hot}}(\mathbf{x}) \quad (8.6)$$

where $\mathbf{f}_{\text{hot}}(\mathbf{x})$ represents the higher order terms of \mathbf{x} around the equilibrium point $\mathbf{x} = 0$. Denote a constant $n \times n$ matrix \mathbf{A} as the Jacobian matrix of \mathbf{f} with respect to \mathbf{x} at $\mathbf{x} = 0$:

$$\mathbf{A} = \left. \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=0} \quad (8.7)$$

THEOREM 8.2

The system $\dot{\mathbf{x}} = \mathbf{Ax}$ is constructed as the local linearized approximation of the original nonlinear system (Equation 8.1) at the equilibrium point $\mathbf{x} = 0$. And the following properties can be obtained as

- Equilibrium point $\mathbf{x} = 0$ for the actual nonlinear system is locally asymptotically stable if all eigenvalues λ_i of \mathbf{A} are strictly in the left-half complex plane.
- Equilibrium point $\mathbf{x} = 0$ for the actual nonlinear system is unstable if at least one eigenvalue of \mathbf{A} is strictly in the right-half complex plane.
- If all eigenvalues of \mathbf{A} are in the left-half complex plane, but at least one of them is on the $j\omega$ axis, then nothing can be concluded about the stability of the origin $\mathbf{x} = 0$ for the original nonlinear system from Lyapunov's indirect method (the equilibrium point may be stable, asymptotically stable, or unstable for the nonlinear system).

Following the same principle, in the controller design process, for a general nonlinear system with a control input vector \mathbf{u} as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (8.8)$$

where

\mathbf{x} is an $n \times 1$ state vector

\mathbf{u} is an $m \times 1$ input vector

$\mathbf{f}(\cdot)$ is an $n \times 1$ nonlinear function vector with $\mathbf{f}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$

The system can be expressed in its expansion form as

$$\dot{\mathbf{x}} = \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \right|_{\mathbf{x}=0, \mathbf{u}=0} \cdot \mathbf{x} + \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \right|_{\mathbf{x}=0, \mathbf{u}=0} \cdot \mathbf{u} + \mathbf{f}_{\text{hot}}(\mathbf{x}, \mathbf{u}) \quad (8.9)$$

where $\mathbf{f}_{\text{hot}}(\mathbf{x}, \mathbf{u})$ represents the higher order terms of \mathbf{x} and \mathbf{u} around the equilibrium point $(\mathbf{x} = 0, \mathbf{u} = 0)$. Denote the $n \times n$ Jacobian matrix \mathbf{A} and the $n \times m$ matrix Jacobian \mathbf{B} as

$$\begin{aligned} \mathbf{A} &= \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \right|_{\mathbf{x}=0, \mathbf{u}=0} \\ \mathbf{B} &= \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \right|_{\mathbf{x}=0, \mathbf{u}=0} \end{aligned} \quad (8.10)$$

By neglecting the higher order terms, the original nonlinear system (Equation 8.8) can be simplified as

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \quad (8.11)$$

which is called the system local Lyapunov-linearized approximation at the equilibrium point ($\mathbf{x} = 0, \mathbf{u} = 0$). If there exists an $m \times n$ gain matrix \mathbf{K} (a state-feedback controller) such that all eigenvalues of $\mathbf{A} - \mathbf{B} \cdot \mathbf{K}$ have negative real parts, that is, $\mathbf{A} - \mathbf{B} \cdot \mathbf{K}$ is a Hurwitz matrix, the control law $\mathbf{u} = \mathbf{g}(\mathbf{x}) = -\mathbf{K} \cdot \mathbf{x}$ will thus ensure the system equilibrium point as asymptotically stable. However, this result is valid only within a small operating range around the origin ($\mathbf{x} = 0, \mathbf{u} = 0$), where the original nonlinear system preserves its linear characteristics and therefore its asymptotic stability is held locally. ■

In this method, the term “indirect” is stressed since the system stability property is indirectly obtained from the local linearized system around the operating point. This method is also named as Lyapunov’s first method.

8.1.4 LYAPUNOV’S METHOD TO THE TS FUZZY CONTROL SYSTEM

The application of Lyapunov’s direct method to the stability analysis of a Takagi–Sugeno (TS) closed-loop fuzzy control system was led by Tanaka and Sugeno (1990, 1992), who provided a sufficient condition for the system asymptotic stability. With this distinguished TS fuzzy representation, the overall system nonlinear dynamics can be exhibited by a set of fuzzy implications, which describe the system local behavior in the state space.

Consider a TS fuzzy system model in the form of

$$\begin{aligned} R^i: & \text{ IF } x(k) \text{ is } A_1^i \text{ AND } \dots \text{ AND } x(k-n+1) \text{ is } A_n^i \text{ AND} \\ & u(k) \text{ is } B_1^i \text{ AND } \dots \text{ AND } u(k-m+1) \text{ is } B_m^i, \\ & \text{THEN } x^i(k+1) = a_0^i + a_1^i x(k) + \dots + a_n^i x(k-n+1) + b_1^i u(k) + \dots + b_m^i u(k-m+1) \end{aligned} \quad (8.12)$$

where

R^i ($i = 1, 2, \dots, l$) denotes the i th rule, l is the number of fuzzy rules

$x(k), \dots, x(k-n+1)$ are the system state variables

$u(k), \dots, u(k-m+1)$ are the input variables

$x^i(k+1)$ is the output from the i th rule

A_p^i ($p = 0, 1, \dots, n$) and B_q^i ($q = 0, 1, \dots, m$) are the input fuzzy sets whose membership functions (MFs) denoted by the same symbols are continuous piecewise-polynomial functions

a_p^i and b_q^i are consequent parameters

Expressed in a vector form, the TS fuzzy system model (Equation 8.12) can be rewritten as

R^i : IF $\mathbf{x}(k)$ is \mathbf{P}^i AND $\mathbf{u}(k)$ is \mathbf{Q}^i ,

$$\text{THEN } x^i(k+1) = a_0^i + \sum_{p=1}^n a_p^i x(k-p+1) + \sum_{q=1}^m b_q^i u(k-q+1) \quad (8.13)$$

where

$$\begin{aligned}\mathbf{x}(k) &= [x(k), \dots, x(k-n+1)]^T \\ \mathbf{P}^i &= [A_1^i, \dots, A_n^i]^T \\ \mathbf{u}(k) &= [u(k), \dots, u(k-m+1)]^T \\ \mathbf{Q}^i &= [B_1^i, \dots, B_m^i]^T\end{aligned}$$

For simplicity, consider the following TS fuzzy system model with zero input and no offset term a_0^i :

$$\begin{aligned}R^i: \text{IF } x(k) \text{ is } A_1^i \text{ AND } \dots \text{ AND } x(k-n+1) \text{ is } A_n^i, \\ \text{THEN } x^i(k+1) = a_1^i x(k) + \dots + a_n^i x(k-n+1)\end{aligned} \quad (8.14)$$

where \mathbf{R}^i ($i=1, 2, \dots, l$). The consequent part of the i th fuzzy implication can be written in a matrix form as

$$\mathbf{x}^i(k+1) = \mathbf{A}_i \mathbf{x}(k) \quad (8.15)$$

where

$$\mathbf{x}(k) = [x(k), \dots, x(k-n+1)]^T$$

$$\mathbf{A}_i = \begin{bmatrix} a_1^i & a_2^i & \cdots & a_{n-1}^i & a_n^i \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

Each linear component $\mathbf{A}_i \mathbf{x}(k)$ is called a linear subsystem. And then the output of the fuzzy system is inferred by fuzzy implication calculation as

$$\mathbf{x}(k+1) = \frac{\sum_{i=1}^l \alpha_i \mathbf{x}^i(k+1)}{\sum_{i=1}^l \alpha_i} = \frac{\sum_{i=1}^l \alpha_i \mathbf{A}_i \mathbf{x}(k)}{\sum_{i=1}^l \alpha_i} \quad (8.16)$$

where α_i is the firing strength of the i th fuzzy rule defined as

$$\alpha_i = \prod_{p=1}^n \mu_{A_p^i}[x_0(k-p+1)] = \mu_{A_1^i}[x_0(k)] \cdot \dots \cdot \mu_{A_n^i}[x_0(k-n+1)] \quad (8.17)$$

with the properties as

$$\begin{aligned} \sum_{i=1}^l \alpha_i &> 0 \\ \alpha_i &\geq 0, \quad i = 1, 2, \dots, l \end{aligned} \tag{8.18}$$

where $x_0(\cdot)$ denotes the crisp input value of $x(\cdot)$ at the current time instance.

THEOREM 8.3 (Tanaka and Sugeno, 1992)

The equilibrium point of the fuzzy system (Equation 8.16) $\mathbf{x}(k)=0$ is globally asymptotically stable if there exists a common positive definite matrix \mathbf{P} for all the subsystems such that

$$\mathbf{A}_i^T \mathbf{P} \mathbf{A}_i - \mathbf{P} < 0 \quad \text{for } i = 1, 2, \dots, l \tag{8.19}$$

Note that this theorem only provides a sufficient condition in order to ensure the stability of the TS fuzzy system model (Equation 8.15). All the \mathbf{A}_i matrices are stable if there exists a common positive definite matrix \mathbf{P} . However, a common positive definite matrix \mathbf{P} does not always exist even if all the \mathbf{A}_i matrices are stable, and a fuzzy system may be globally asymptotically stable even if there does not exist a common positive definite matrix \mathbf{P} . Furthermore, a fuzzy system is not always globally asymptotically stable even if all the \mathbf{A}_i matrices are stable. Therefore, in the Theorem 8.4, a necessary condition is provided to ensure the existence of a common positive definite matrix \mathbf{P} . ■

THEOREM 8.4 (Tanaka and Sugeno, 1992)

Assume that \mathbf{A}_i is a stable and nonsingular matrix for $i = 1, 2, \dots, l$. $\mathbf{A}_i \mathbf{A}_j$ is a stable matrix for $i, j = 1, 2, \dots, l$ if there exists a common positive definite matrix \mathbf{P} such that

$$\mathbf{A}_i^T \mathbf{P} \mathbf{A}_i - \mathbf{P} < 0 \tag{8.20}$$

Next, we will consider a feedback closed-loop connection of a fuzzy TS system model R^i and a fuzzy TS controller C^j as shown in Figure 8.1, where $r(k)$ is the reference input to the system. ■

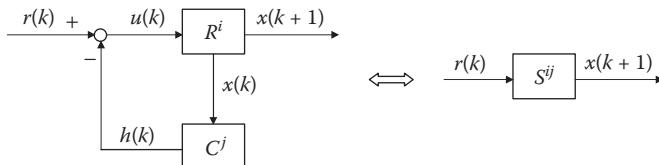


FIGURE 8.1 Fuzzy block diagram for a feedback closed-loop connection.

THEOREM 8.5 (Tanaka and Sugeno, 1992)

Assume the fuzzy model R^i and the fuzzy controller C^j have the following expression as

$$\begin{aligned} R^i: \text{IF } \mathbf{x}(k) \text{ is } \mathbf{P}^i \text{ AND } \mathbf{u}(k) \text{ is } \mathbf{Q}^i, \\ \text{THEN } x^i(k+1) = a_0^i + \sum_{p=1}^n a_p^i x(k-p+1) + b^i u(k) \end{aligned} \quad (8.21)$$

$$\begin{aligned} C^j: \text{IF } \mathbf{x}(k) \text{ is } \mathbf{G}^j \text{ AND } \mathbf{u}(k) \text{ is } \mathbf{H}^j, \\ \text{THEN } h^j(k) = c_0^j + \sum_{p=1}^n c_p^j x(k-p+1) \end{aligned} \quad (8.22)$$

where

$$\begin{aligned} \mathbf{P}^i &= [A_1^i, \dots, A_n^i]^T \\ \mathbf{Q}^i &= [B_1^i, \dots, B_m^i]^T \\ \mathbf{G}^j &= [C_1^j, \dots, C_n^j]^T \\ \mathbf{H}^j &= [D_1^j, \dots, D_m^j]^T \end{aligned}$$

The result of the feedback connection of R^i and C^j is equivalent to the following fuzzy block S^{ij} as

$$\begin{aligned} S^{ij}: \text{IF } \mathbf{x}(k) \text{ is } (\mathbf{P}^i \text{ AND } \mathbf{G}^j) \text{ AND } \mathbf{u}(k) \text{ is } (\mathbf{Q}^i \text{ AND } \mathbf{H}^j), \\ \text{THEN } x^{ij}(k+1) = a_0^i - b^i c_0^j + b^i r(k) + \sum_{p=1}^n (a_p^i - b^i c_p^j) x(k-p+1) \end{aligned} \quad (8.23)$$

where

$$\begin{aligned} i &= 1, \dots, l_1 \\ j &= 1, \dots, l_2 \end{aligned}$$
■

Example 8.1

Suppose the fuzzy model R^i and the fuzzy controller C^j are in the form of

$$\begin{aligned} R^1: \text{IF } x(k) \text{ is } A_1^1 \text{ AND } x(k-1) \text{ is } A_2^1 \text{ AND } u(k) \text{ is } B^1, \\ \text{THEN } x^1(k+1) = x(k) + 2x(k-1) - u(k) \\ R^2: \text{IF } x(k) \text{ is } A_1^2 \text{ AND } x(k-1) \text{ is } A_2^2 \text{ AND } u(k) \text{ is } B^2, \\ \text{THEN } x^2(k+1) = 0.2x(k) - 3x(k-1) + 0.5u(k) \end{aligned} \quad (8.24)$$

$$\begin{aligned} C^1: \text{IF } x(k) \text{ is } C_1^1 \text{ AND } x(k-1) \text{ is } C_2^1, \\ \text{THEN } h^1(k) = k_1^1 x(k) + k_2^1 x(k-1) \\ C^2: \text{IF } x(k) \text{ is } C_1^2 \text{ AND } x(k-1) \text{ is } C_2^2, \\ \text{THEN } h^2(k) = k_1^2 x(k) + k_2^2 x(k-1) \end{aligned} \quad (8.25)$$

From Theorem 8.5, the feedback closed-loop connection S^{ij} can be derived as

- S^{11} : IF $x(k)$ is $(A_1^1 \text{ AND } C_1^1)$ AND $x(k-1)$ is $(A_2^1 \text{ and } C_2^1)$ AND $u(k)$ is B^1 ,
 THEN $x^{11}(k+1) = (1 + k_1^1)x(k) + (2 + k_2^1)x(k-1) - r(k)$
- S^{12} : IF $x(k)$ is $(A_1^1 \text{ and } C_1^2)$ AND $x(k-1)$ is $(A_2^1 \text{ and } C_2^2)$ AND $u(k)$ is B^1 ,
 THEN $x^{12}(k+1) = (1 + k_1^2)x(k) + (2 + k_2^2)x(k-1) - r(k)$
- S^{21} : IF $x(k)$ is $(A_1^2 \text{ and } C_1^1)$ AND $x(k-1)$ is $(A_2^2 \text{ and } C_2^1)$ AND $u(k)$ is B^2 ,
 THEN $x^{21}(k+1) = (0.2 - 0.5k_1^1)x(k) + (-3 - 0.5k_2^1)x(k-1) + 0.5r(k)$
- S^{22} : IF $x(k)$ is $(A_1^2 \text{ and } C_1^2)$ AND $x(k-1)$ is $(A_2^2 \text{ and } C_2^2)$ AND $u(k)$ is B^2 ,
 THEN $x^{22}(k+1) = (0.2 - 0.5k_1^2)x(k) + (-3 - 0.5k_2^2)x(k-1) + 0.5r(k)$ (8.26)

As derived by Tanaka and Sugeno (1992), the feedback closed-loop connection of a fuzzy TS system model and a fuzzy TS controller is equivalent to a dynamic TS fuzzy system, which preserves the following two properties:

1. All the fuzzy sets in the premise parts are described by continuous piecewise-polynomial MFs.
2. All the consequent parts are described by linear equations.

Assume the reference input $r(k) = 0$, and for the linear subsystem S^{11} , S^{12} , S^{21} , and S^{22} , we obtain the four matrices as

$$A^{11} = \begin{bmatrix} 1 + k_1^1 & 2 + k_2^1 \\ 1 & 0 \end{bmatrix}, \quad A^{12} = \begin{bmatrix} 1 + k_1^2 & 2 + k_2^2 \\ 1 & 0 \end{bmatrix}$$

$$A^{21} = \begin{bmatrix} 0.2 - 0.5k_1^1 & -3 - 0.5k_2^1 \\ 1 & 0 \end{bmatrix}, \quad A^{22} = \begin{bmatrix} 0.2 - 0.5k_1^2 & -3 - 0.5k_2^2 \\ 1 & 0 \end{bmatrix}$$

Therefore, the general fuzzy theory, such as the stability requirement in Theorems 8.3 and 8.4, can be applied to the feedback-connected fuzzy system, and an appropriate model-based fuzzy controller can be designed correspondingly, where the parameters of the fuzzy controller (k_1^1 , k_2^1 , k_1^2 , and k_2^2) are determined to guarantee the stability of each subsystem of the overall closed-loop system.

8.1.5 STABILITY CONCEPTS FOR NONAUTONOMOUS SYSTEMS

Consider a general nonlinear nonautonomous (time-varying) dynamic system expressed in continuous domain as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) \quad (8.27)$$

where

$\mathbf{x} = [x_1, \dots, x_n]^T \in \mathbb{R}^n$ is the system state vector
 $\mathbf{f}(\cdot, \cdot)$ is an n -dimensional nonlinear function vector

DEFINITION 8.7

A state \mathbf{x}^* is an equilibrium point of the system (Equation 8.27) if once \mathbf{x} is equal to \mathbf{x}^* , it remains equal to \mathbf{x}^* for all $t \geq t_0$. Therefore,

$$0 = \mathbf{f}(\mathbf{x}^*, t), \quad t \geq t_0 \quad (8.28)$$

■

DEFINITION 8.8

The equilibrium state $\mathbf{x} = 0$ is stable at t_0 (or Lyapunov stable) if for any $R > 0$, there exists $r(R, t_0) > 0$, such that if $\|\mathbf{x}(t_0)\| < r(R, t_0)$, then $\|\mathbf{x}(t)\| < R$ for all $t \geq t_0$. The notation $r(R, t_0)$ means that r is a function of R and t_0 . ■

DEFINITION 8.9

The equilibrium state $\mathbf{x} = 0$ is asymptotically stable at t_0 if it is stable and additionally, there exists some $\delta(t_0) > 0$, such that $\mathbf{x}(t) \rightarrow 0$ as $t \rightarrow \infty$ for $\|\mathbf{x}(t_0)\| < \delta(t_0)$. ■

DEFINITION 8.10

The equilibrium state $\mathbf{x} = 0$ is exponentially stable at t_0 if there exist two positive numbers α and λ such that

$$\|\mathbf{x}(t)\| \leq \alpha \|\mathbf{x}(t_0)\| e^{-\lambda(t-t_0)}, \quad \forall t > t_0 \quad (8.29)$$

for all $\|\mathbf{x}(t_0)\| < r$. The positive number λ is called the rate of exponential convergence. ■

DEFINITION 8.11

The equilibrium state $\mathbf{x} = 0$ is globally asymptotically (or exponentially) stable if it is asymptotically (or exponentially) stable for any initial state $\mathbf{x}(t_0) \in \mathbb{R}^n$. ■

DEFINITION 8.12

The equilibrium state $\mathbf{x} = 0$ is locally uniformly stable if the radius $r(R, t_0)$ in Definition 8.9 is independent of t_0 , for example, for any $R > 0$, there exists $r(R) > 0$, such that if $\|\mathbf{x}(t_0)\| < r(R)$, then $\|\mathbf{x}(t)\| < R$ for all $t \geq t_0$. ■

DEFINITION 8.13

The equilibrium state $\mathbf{x} = 0$ is uniformly asymptotically stable if it is uniformly stable and furthermore, there exists some $\delta > 0$, which is independent of t_0 , such that for $\|\mathbf{x}(t_0)\| < \delta$ any system trajectory $\mathbf{x}(t)$ converges to 0 as $t \rightarrow \infty$, uniformly in t_0 .

The equilibrium state $\mathbf{x} = 0$ is uniformly asymptotically stable if the ball of attraction \mathbf{B}_δ is defined as the whole state space. ■

8.1.5.1 Lyapunov's Direct Method

For the nonlinear nonautonomous system (Equation 8.27), if a scalar time-varying function $V(\mathbf{x}, t)$ is continuously differentiable with respect to its arguments, then its derivative along a system trajectory is obtained as

$$\dot{V}(\mathbf{x}, t) = \frac{dV}{dt} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial \mathbf{x}} \dot{\mathbf{x}} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial \mathbf{x}} \cdot \mathbf{f}(\mathbf{x}, t) \quad (8.30)$$

DEFINITION 8.14

A time-varying continuous function $V(\mathbf{x}, t)$ is locally positive definite if $V(0, t) = 0$ and there exists a time-invariant positive definite function $V_0(\mathbf{x})$ such that

$$V(\mathbf{x}, t) \geq V_0(\mathbf{x}), \quad \forall t \geq t_0 \quad (8.31)$$

In other words, this function $V(\mathbf{x}, t)$ is locally positive definite if it dominates a time-invariant positive definite function. With the same principle, a function $V(\mathbf{x}, t)$ is locally positive semidefinite if it dominates a time-invariant positive semidefinite function. A function $V(\mathbf{x}, t)$ is locally negative definite if $-V(\mathbf{x}, t)$ is locally positive definite. A function $V(\mathbf{x}, t)$ is locally negative semidefinite if $-V(\mathbf{x}, t)$ is locally positive semidefinite. ■

DEFINITION 8.15

A scalar function $V(\mathbf{x}, t)$ is decrescent if $V(0, t) = 0$, and if there exists a time-invariant positive definite function $V_1(\mathbf{x})$ such that

$$V(\mathbf{x}, t) \leq V_1(\mathbf{x}), \quad \forall t \geq t_0 \quad (8.32)$$

THEOREM 8.6

If in a spherical ball \mathbf{B}_R around the equilibrium point $\mathbf{x} = 0$, the scalar time-varying function $V(\mathbf{x}, t)$ is positive definite and its derivative along the system trajectory is negative semidefinite, then the function is called a Lyapunov function for the nonlinear nonautonomous system.

Denote the origin $\mathbf{x} = 0$ as an equilibrium point for the nonlinear system (Equation 8.27). If there exists a scalar function $V(\mathbf{x}, t)$ with continuous first partial derivatives such that

- $V(\mathbf{x}, t)$ is positive definite
- $\dot{V}(\mathbf{x}, t)$ is negative semidefinite

then the equilibrium point $\mathbf{x} = 0$ is stable in the sense of Lyapunov. If additionally, $V(\mathbf{x}, t)$ is decrescent, then the equilibrium point $\mathbf{x} = 0$ is uniformly stable. If $\dot{V}(\mathbf{x}, t)$ is negative definite, then the equilibrium point is uniformly asymptotically stable. If furthermore, $V(\mathbf{x}) \rightarrow \infty$ as $\|\mathbf{x}\| \rightarrow \infty$, then the equilibrium point is globally uniformly asymptotically stable. ■

8.1.5.2 Lyapunov's Indirect Method

Consider the origin $\mathbf{x} = 0$ as the equilibrium point for the nonlinear nonautonomous system (Equation 8.27), Assume that $\mathbf{f}(\cdot, \cdot)$ is continuously differentiable with respect to \mathbf{x} . Denote the $n \times n$ Jacobian matrix as

$$\mathbf{A}(t) = \left. \frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial \mathbf{x}} \right|_{\mathbf{x}=0} \quad (8.33)$$

Then for any fixed time t ,

$$\dot{\mathbf{x}} = \mathbf{A}(t) \cdot \mathbf{x} + \mathbf{f}_{\text{hot}}(\mathbf{x}, t) \quad (8.34)$$

If the following condition is satisfied,

$$\lim_{\|\mathbf{x}\| \rightarrow 0} \left(\sup \frac{\|\mathbf{f}_{\text{hot}}(\mathbf{x}, t)\|}{\|\mathbf{x}\|} \right) = 0, \quad \forall t \geq 0 \quad (8.35)$$

then Lyapunov's indirect method applies.

THEOREM 8.7

The system $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}$ is called the local linearized approximation of the original nonlinear nonautonomous system (Equation 8.27) around the equilibrium point

$\mathbf{x}=0$. If the linearized system with condition (Equation 8.35) is uniformly asymptotically stable, then the equilibrium point $\mathbf{x}=0$ of the original nonautonomous system (Equation 8.27) is also uniformly asymptotically stable. ■

8.2 PASSIVITY APPROACH

Another promising method in stability analysis for a general nonlinear control system is passivity approach, where the fuzzy controller is considered as a nonlinear mapping relating the inputs to the outputs and the system absolute stability is derived based on the input-output dynamics characteristics. This method does not need an explicit system mathematical expression and the stability of the nonlinear closed-loop fuzzy control system can be guaranteed in a straightforward way.

8.2.1 PASSIVITY CONCEPT

8.2.1.1 Continuous-Time Case

Consider a continuous system in the state-space form as

$$\dot{\mathbf{x}} = f(\mathbf{x}, u), \quad \mathbf{x} \in \mathbf{R}^n, \quad u \in \mathbf{R} \quad (8.36)$$

$$y = h(\mathbf{x}, u), \quad y \in \mathbf{R} \quad (8.37)$$

where $f(\cdot, \cdot)$ and $h(\cdot, \cdot)$ are smooth mappings with their arguments $\mathbf{x} \in \mathbf{R}^n$, $u \in \mathbf{R}$, and $y \in \mathbf{R}$. Furthermore, assume that for any $u \in \mathbf{R}$ and for any initial condition $\mathbf{x}(t_0)$, the supply function $s(u, y) = u^T y \in \mathbf{R}$.

DEFINITION 8.16 (Xu and Shin, 2005)

System (Equation 8.36) with a properly chosen output (Equation 8.37) is said to be passive with respect to the supply rate $s(u, y)$ if there exists a positive definite function V , with $V(0)=0$, regarded as the storage function, such that the following inequality is satisfied for all $\mathbf{x}(t_0)$,

$$V[\mathbf{x}(t_f)] - V[\mathbf{x}(t_0)] \leq \int_{t_0}^{t_f} s(u(\sigma), y(\sigma)) d\sigma = \int_{t_0}^{t_f} u(\sigma)^T y(\sigma) d\sigma, \quad \forall \mathbf{x}, u \quad (8.38)$$

The system is strictly input passive if there exists a constant $\varepsilon > 0$ such that the system is passive with respect to $s(u, y) = u^T y - \varepsilon u^T u$. The system is strictly output passive if there exists a constant $\varepsilon > 0$ such that $s(u, y) = u^T y - \varepsilon y^T y$. ■

8.2.1.2 Discrete-Time Case

Consider the discrete-time system in the form of

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), u(k)), \quad \mathbf{x} \in \mathbf{R}^n, \quad u \in \mathbf{R} \quad (8.39)$$

$$y(k) = h(\mathbf{x}(k), u(k)), \quad y \in \mathbf{R} \quad (8.40)$$

where $f(\cdot, \cdot)$ and $h(\cdot, \cdot)$ are smooth mappings with their arguments $\mathbf{x} \in \mathbf{R}^n$, $u \in \mathbf{R}$ and $y \in \mathbf{R}$. Furthermore, assume that for any $u \in \mathbf{R}$ and for any initial condition $\mathbf{x}(0)$, the supply function $s(u(k), y(k)) = u(k)^T y(k) \in \mathbf{R}$ for all $k \geq 0$.

DEFINITION 8.17 (Xu and Shin, 2005)

A system (Equation 8.39) with a properly chosen output (Equation 8.40) is said to be passive with respect to the supply rate $s(u(k), y(k))$ if there exists a positive definite function V , with $V(0) = 0$, regarded as the storage function, such that the following inequality is satisfied for all $\mathbf{x}(0)$, and for all $k \in \mathbb{Z}_+ := \{0, 1, 2, \dots\}$ (Ferrari-Trecate et al., 2002; López, 2002),

$$V[\mathbf{x}(k+1)] - V[\mathbf{x}(k)] \leq s(u(k), y(k)) = u(k)^T y(k), \quad \forall \mathbf{x}(k), u(k) \forall k \quad (8.41)$$

This inequality can be rewritten as

$$V[\mathbf{x}(k+1)] - V[\mathbf{x}(0)] \leq \sum_{i=0}^k s(u(i), y(i)) = \sum_{i=0}^k u(i)^T y(i), \quad \forall \mathbf{x}(0), \forall u(k), \forall k \quad (8.42)$$

The system is strictly input passive if there exists a constant $\varepsilon > 0$ such that the system is passive with respect to $s(u(k), y(k)) = u(k)^T y(k) - \varepsilon u(k)^T u(k)$. The system is strictly output passive if there exists a constant $\varepsilon > 0$ such that $s(u(k), y(k)) = u(k)^T y(k) - \varepsilon y(k)^T y(k)$. ■

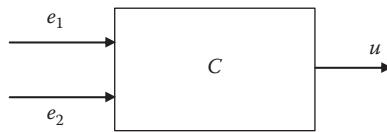
8.2.2 SECTORIAL FUZZY CONTROLLER

It has been known that a fuzzy controller can be considered as a nonlinear static mapping relating inputs to outputs, which can be expressed in the form of fuzzy rule table. The control rules represent a human expert's commonsense control strategy in the operation process. Many fuzzy controllers considered in the literature share the same distinguished characters such as the following (Hwang and Yeh, 1996; Kwong and Passino, 1996; Siettos et al., 2000; Ustundag et al., 2002):

1. Fuzzy control rules are antisymmetric about the off-diagonal of the table (odd symmetry).
2. Numeric values in the fuzzy table gradually increase/decrease from left to right within a row, while within a column they gradually increase/decrease from top to bottom (monotony).

3. Control decision corresponding to the central element of the rule table is usually zero (i.e., output is null for null inputs), and the value around the central area is small.

These characteristics of the nonlinear fuzzy controller reflect the general properties of a system and the consistency of the operator's control action. Based on the features of the fuzzy controllers used in real-world applications, it is necessary to extract some numerical formula that can help the stability analysis of fuzzy control systems. The fuzzy controller considered here contains two real input variables, e_1 and e_2 . After fuzzification, specific fuzzy rules are fired by a fuzzy inference engine and combined to obtain consequences through defuzzification procedure, thereby resulting in a scalar output value $u = \Phi(e_1, e_2)$, which is related to the two real input variables as



where C represents the fuzzy controller as an input–output nonlinear static mapping. A class of fuzzy controllers, which has the specific sectorial properties of this mapping, has been established as sectorial fuzzy controllers (Calcev, 1996, 1998; Calcev et al., 1998), whose properties can be described as the following.

8.2.2.1 Inputs

Consider the fuzzy controllers involving two scalar inputs e_1 and e_2 , which are normalized to the same range $[-L, L]$. Therefore, $2N+1$ input fuzzy sets, whose linguistic names are E_i , $i = -N, \dots, -1, 0, 1, \dots, +N$, are uniformly defined within the range. The properties of the corresponding input MFs are as follows:

1. Summation of MFs values is 1:

$$\sum_{i=-N}^N \mu_{E_i}(e) = 1$$

2. For the input value outside of the range of $[-L, L]$,

$$e > L \Rightarrow \mu_{E_N}(e) = 1 \quad \text{AND} \quad e < -L \Rightarrow \mu_{E_{-N}}(e) = 1$$

3. E_i and E_{-i} cover intervals that are symmetric with respect to 0:

$$0 \leq \mu_{E_i}(e) = \mu_{E_{-i}}(-e) \leq 1$$

4. At one time, each input value fires at most two adjacent fuzzy sets, with complementary membership grades:

$$\text{if } |i - j| > 1 \Rightarrow \mu_{E_i}(e)\mu_{E_j}(e) = 0$$

5. Fuzzy sets for the input MFs must be convex:

$$\mu_{E_i}[\lambda e + (1 - \lambda)e'] \geq \min [\mu_{E_i}(e), \mu_{E_i}(e')], \quad \forall e \neq e', \quad \forall \lambda \in [0, 1]$$

For E_0 , the fuzzy set must be strictly convex to guarantee the uniqueness of the zero-state equilibrium of the fuzzy control system. For example, it is not allowed to use a trapezoidal MF for E_0 .

8.2.2.2 Rule Base

The fuzzy rule base, which is designed for the two inputs e_1 and e_2 , one output u , consists of a set of fuzzy if-then rules in the form of

$$R: \text{IF } e_1 \text{ is } E_i \text{ AND } e_2 \text{ is } E_j, \text{ THEN } u \text{ is } \text{out}_{f(i,j)}$$

where

$f(i, j)$ can be any function whose value at i

j is an integer, relating the indices i and j of the input fuzzy sets to the index of the output fuzzy set $\text{out}_{f(i,j)}$ with the center value $U_{f(i,j)}$

The total number of fuzzy rules is $(2N + 1)^2$. The function $f(i, j)$ has the following properties:

1. Zero inputs result in zero output:

$$f(0, 0) = 0$$

2. Control rules are odd symmetric with respect to 0:

$$f(i, j) = -f(-i, -j), \quad \forall i, j$$

3. If one of the inputs is maintained to be constant, the corresponding output will be a convex function with a sectorial property:

$$j \cdot [f(i, j) - f(i, 0)] \geq 0, \quad \forall i, j \geq 0$$

$$i \cdot [f(i, j) - f(0, j)] \geq 0, \quad \forall i, j \geq 0$$

The property of the center value of the output MF is

$$U_0 = 0, \quad U_i = -U_{-i} \quad \text{and} \quad i \geq j \Rightarrow U_i \geq U_j$$

As an illustrative example, [Table 8.1](#) shows a lookup table with odd symmetry and monotony properties in linguistic terms.

8.2.2.3 Output

The scalar controller output u is obtained by the center average defuzzification, using either min or product inference method. If min operator is adopted here,

TABLE 8.1
Sectorial Fuzzy Controller Lookup Table

e_2/e_1	E_{-2}	E_{-1}	E_0	E_1	E_2
E_{-2}	U_{-4}	U_{-3}	U_{-2}	U_{-1}	U_0
E_{-1}	U_{-4}	U_{-2}	U_{-1}	U_0	U_1
E_0	U_{-2}	U_{-1}	U_0	U_1	U_2
E_1	U_{-1}	U_0	U_1	U_2	U_3
E_2	U_0	U_1	U_2	U_3	U_4

$$u = \Phi(e_1, e_2) = \frac{\sum_{i,j} U_{f(i,j)} \cdot \min[\mu_{E_i}(e_1), \mu_{E_j}(e_2)]}{\sum_{i,j} \min[\mu_{E_i}(e_1), \mu_{E_j}(e_2)]} \quad (8.43)$$

Note that because of the complementary properties of the MFs stated above, only a maximum of four rules can be fired at any time. Hence, each of the two summations in the above formula contains at most four items, thus reducing the computation time.

DEFINITION 8.18 (Calcev, 1998)

A fuzzy controller satisfying the previous assumptions, with respect to the inputs, the rule base, and the output, is called a sectorial fuzzy controller. ■

8.2.3 PROPERTY OF SECTORIAL FUZZY CONTROLLER

PROPERTY 8.1 (Xu and Shin, 2005)

For any fuzzy controller addressed above, the input–output nonlinear mapping can be described by a continuous bounded Lipschitz function $\Phi(\cdot, \cdot)$ with the following properties:

- a. $|\Phi(e_1, e_2)| \leq u_M$ and $M = \max_{i,j} U_{f(i,j)}$
- b. $\Phi(0, 0) = 0$ (steady state condition)
- c. $\Phi(e_1, e_2) = -\Phi(-e_1, -e_2)$ (odd symmetry)
- d. $\Phi(e_1, 0) = \Rightarrow e_1 = 0$
- e. $\Phi(\cdot, \cdot)$ is a sectorial function, in the sense that for every (e_1, e_2) , there exist $\lambda', \gamma' > 0$ such that

$$0 \leq e_1 \cdot [\Phi(e_1, e_2) - \Phi(0, e_2)] \leq \lambda' e_1^2 \quad (8.44)$$

$$0 \leq e_2 \cdot [\Phi(e_1, e_2) - \Phi(e_1, 0)] \leq \gamma' e_2^2 \quad (8.45)$$

Proof

Property 8.1a: For each fuzzy input partition, the consequence $U_{f(k,l)}$ is a finite number. In the defuzzification step, only a maximum of four rules can be activated at the same time, and the output can be formulated as a simple superposition of the MFs $U_{f(k,l)}$

$$u = \Phi(e_1, e_2) = \frac{\sum_{k=-N}^N \sum_{l=-N}^N U_{f(k,l)} \cdot \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}{\sum_{k=-N}^N \sum_{l=-N}^N \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}$$

It is obvious that at every time, the output is a convex combination of value $U_{f(k,l)}$, and because of the symmetric property, $u \in [-u_M, u_M]$, where $M = \max_{k,l} U_{f(k,l)}$.

Property 8.1b: When both of the inputs are 0,

$$\begin{aligned}\Phi(0, 0) &= \frac{U_{f(0,0)} \cdot \min[\mu_{E_k}(0), \mu_{E_l}(0)]}{\min[\mu_{E_k}(0), \mu_{E_l}(0)]} \\ &= \frac{U_0 \cdot \min[\mu_{E_k}(0), \mu_{E_l}(0)]}{\min[\mu_{E_k}(0), \mu_{E_l}(0)]} \\ &= \frac{0 \cdot \min[\mu_{E_k}(0), \mu_{E_l}(0)]}{\min[\mu_{E_k}(0), \mu_{E_l}(0)]} \\ &= 0\end{aligned}$$

Thus, $\Phi(0, 0) = 0$.

Property 8.1c:

$$\begin{aligned}\Phi(e_1, e_2) &= \frac{\sum_{k=-N}^N \sum_{l=-N}^N U_{f(k,l)} \cdot \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}{\sum_{k=-N}^N \sum_{l=-N}^N \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]} \\ &= \frac{\sum_{k=-N}^N \sum_{l=-N}^N -U_{f(-k,-l)} \cdot \min[\mu_{-E_k}(-e_1), \mu_{-E_l}(-e_2)]}{\sum_{k=-N}^N \sum_{l=-N}^N \min[\mu_{-E_k}(-e_1), \mu_{-E_l}(-e_2)]} \\ &= -\frac{\sum_{k=-N}^N \sum_{l=-N}^N U_{f(-k,-l)} \cdot \min[\mu_{-E_k}(-e_1), \mu_{-E_l}(-e_2)]}{\sum_{k=-N}^N \sum_{l=-N}^N \min[\mu_{-E_k}(-e_1), \mu_{-E_l}(-e_2)]} \\ &= -\Phi(-e_1, -e_2)\end{aligned}$$

Hence, $\Phi(e_1, e_2) = -\Phi(-e_1, -e_2)$

Property 8.1d:

$$\begin{aligned}
\Phi(e_1, 0) &= \frac{\sum_{k=-N}^N U_{f(k,0)} \cdot \min[\mu_{E_k}(e_1), \mu_0(0)]}{\sum_{k=-N}^N \min[\mu_{E_k}(e_1), \mu_0(0)]} \\
&= \frac{\sum_{k=-N}^N U_{f(k,0)} \cdot \min[\mu_{E_k}(e_1), 1]}{\sum_{k=-N}^N \min[\mu_{E_k}(e_1), 1]} \\
&= \frac{\sum_{k=-N}^N U_{f(k,0)} \cdot \mu_{E_k}(e_1)}{\sum_{k=-N}^N \mu_{E_k}(e_1)} \\
&= \sum_{k=-N}^N U_{f(k,l)} \cdot \mu_{E_k}(e_1) \\
&= U_{f(i,0)} \cdot \mu_{E_i}(e_1) + U_{f(i+1,0)} \cdot \mu_{E_{i+1}}(e_1)
\end{aligned}$$

Hence, $\Phi(e_1, 0) = 0$ is possible for either $U_{f(i,0)} = 0$ with $\mu_{E_{i+1}}(e_1) = 0$, or $U_{f(i+1,0)} = 0$ with $\mu_{E_i}(e_1) = 0$. Or in the simple way of expression, for $i = 0$ or $i = -1$. In both cases, $\mu_{E_0}(e_1) = 1 \Rightarrow e_1 = 0$. Thus, $\Phi(e_1, 0) = 0 \Rightarrow e_1 = 0$.

Property 8.1e: Due to the simple superposition of membership functions, at each sampling time only at most four rules can be activated simultaneously. Without loss of generality, let us assume one of the inputs is constant and prove the property by varying the other one. For convenience, set $e_2 = \text{constant} > 0$. The symmetric property of the fuzzy rule table allows us to assume $e_1 > 0$. The following four rules will be fired:

- R_{ij} : IF e_1 is E_i AND e_2 is E_j , THEN u is $\text{out}_{f(i,j)}$
- R_{ij+1} : IF e_1 is E_i AND e_2 is E_{j+1} , THEN u is $\text{out}_{f(i,j+1)}$
- R_{i+1j} : IF e_1 is E_{i+1} AND e_2 is E_j , THEN u is $\text{out}_{f(i+1,j)}$
- R_{i+1j+1} : IF e_1 is E_{i+1} AND e_2 is E_{j+1} , THEN u is $\text{out}_{f(i+1,j+1)}$

where $i, j \geq 0$. The fuzzy controller output can be obtained by the center average defuzzification

$$\begin{aligned}
u &= \Phi(e_1, e_2) = \frac{\sum_{k=-N}^N \sum_{l=-N}^N U_{f(k,l)} \cdot \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}{\sum_{k=-N}^N \sum_{l=-N}^N \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]} \\
&= \frac{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} U_{f(k,l)} \cdot \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}
\end{aligned}$$

Evaluate the expression

$$\begin{aligned}
& \Phi(e_1, e_2) - \Phi(0, e_2) \\
&= \frac{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} U_{f(k,l)} \cdot \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]} - \frac{\sum_{l=j}^{j+1} U_{f(0,l)} \cdot \mu_{E_l}(e_2)}{\sum_{l=j}^{j+1} \mu_{E_l}(e_2)} \\
&= \frac{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} U_{f(k,l)} \cdot \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]} - \sum_{l=j}^{j+1} U_{f(0,l)} \cdot \mu_{E_l}(e_2) \\
&= \frac{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} U_{f(k,l)} \cdot \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]} - \left\{ U_{f(0,j)} \cdot \mu_{E_j}(e_2) + U_{f(0,j+1)} \cdot [1 - \mu_{E_j}(e_2)] \right\} \\
&= \frac{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} U_{f(k,l)} \cdot \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]} \\
&\quad - \frac{\left\{ \sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)] \right\} \left\{ U_{f(0,j+1)} - \mu_{E_j}(e_2) \cdot [U_{f(0,j+1)} - U_{f(0,j)}] \right\}}{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]} \\
&= \frac{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} [U_{f(k,l)} - U_{f(0,j+1)}] \cdot \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]} \\
&\quad + \frac{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \left\{ \mu_{E_j}(e_2) [U_{f(0,j+1)} - U_{f(0,j)}] \right\} \cdot \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]} \\
&\geq \frac{\min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)] \cdot \left\{ \left[\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} U_{f(k,l)} \right] - 4 \cdot U_{f(0,j+1)} \right\}}{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]} \\
&\quad + \frac{\min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)] \cdot \sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \left\{ \mu_{E_j}(e_2) [U_{f(0,j+1)} - U_{f(0,j)}] \right\}}{\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)]}
\end{aligned}$$

Based on the property of the lookup table and the assumption on e_1 and e_2 , we have

$$e_1 \cdot \left\{ \left[\sum_{k=i}^{i+1} \sum_{l=j}^{j+1} U_{f(k,l)} \right] - 4 \cdot U_{f(0,j+1)} \right\} \geq 0$$

$$e_1 \cdot \sum_{k=i}^{i+1} \sum_{l=j}^{j+1} \left\{ \mu_{E_j}(e_2) [U_{f(0,j+1)} - U_{f(0,j)}] \right\} \geq 0, \quad \forall e_1 > 0, \forall e_2$$

and therefore

$$e_1 \cdot [\Phi(e_1, e_2) - \Phi(0, e_2)] \geq 0, \quad \forall e_1 > 0, \forall e_2$$

Because the input–output mapping $\Phi(\cdot, \cdot)$ is globally Lipschitz continuous, for some constant λ'

$$|\Phi(e_1, e_2) - \Phi(0, e_2)| \leq \lambda' |e_1 - 0|$$

For $\forall e_1 > 0, |e_1 - 0| = e_1$, and

$$\begin{aligned} e_1 \cdot [\Phi(e_1, e_2) - \Phi(0, e_2)] &\geq 0 \\ \Rightarrow \Phi(e_1, e_2) - \Phi(0, e_2) &\geq 0 \\ \Rightarrow |\Phi(e_1, e_2) - \Phi(0, e_2)| &= \Phi(e_1, e_2) - \Phi(0, e_2) \end{aligned}$$

Subsequently, the inequality $\Phi(e_1, e_2) - \Phi(0, e_2) \leq \lambda' e_1$ satisfies for the constant $\lambda' > 0$.

Hence,

$$0 \leq e_1 [\Phi(e_1, e_2) - \Phi(0, e_2)] \leq \lambda' e_1^2, \quad \forall e_1, \forall e_2 \quad (8.46)$$

Similarly,

$$0 \leq e_2 \cdot [\Phi(e_1, e_2) - \Phi(e_1, 0)] \leq \gamma' e_2^2, \quad \forall e_1, \forall e_2 \quad (8.47)$$

■

8.2.4 PASSIVITY OF SECTORIAL FUZZY CONTROLLER IN CONTINUOUS DOMAIN

THEOREM 8.8 (Xu and Shin, 2005)

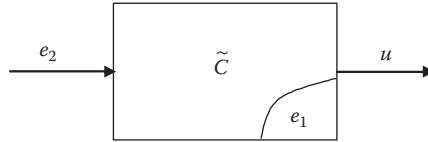
A sectorial fuzzy controller is input–output passive stable in continuous domain.

Proof In continuous case of fuzzy control design, e_1 represents the error $e(t)$, e_2 represents the change of the error $\dot{e}(t)$, and u is the control action $\Phi(e_1, e_2)$. Thus, the fuzzy control system can be defined as

$$\dot{e}_1 = e_2$$

$$u = \Phi(e_1, e_2)$$

Then, the fuzzy controller can be considered as a single-input single-output (SISO) nonlinear system with internal dynamics, where e_2 is the input, u is the output, and e_1 is the state as shown in the following figure:



where \tilde{C} is another expression of the fuzzy controller C with e_1 as the state variable. Following Property 8.1, it is obvious that this system should have $u=0$ as the equilibrium point.

Property 8.1b and e leads to

$$0 \leq e_1 \cdot \Phi(e_1, 0) \leq \lambda' e_1^2$$

$$0 \leq e_2 \cdot \Phi(0, e_2) \leq \gamma' e_2^2$$

$$\text{Let } \Delta_{e_2}(e_1, e_2) = \Phi(e_1, e_2) - \Phi(e_1, 0)$$

$$\Delta_{e_1}(e_1, e_2) = \Phi(e_1, e_2) - \Phi(0, e_2)$$

Then it can be found that

$$0 \leq e_2 \cdot \Delta_{e_2}(e_1, e_2) \leq \gamma' e_2^2$$

$$0 \leq e_1 \cdot \Delta_{e_1}(e_1, e_2) \leq \lambda' e_1^2$$

Applying definition of passivity results in

$$\begin{aligned}
 \int_0^t e_2(\tau) \cdot \Phi(e_1(\tau), e_2(\tau)) d\tau &= \int_0^t \dot{e}_1(\tau) \cdot \Phi(e_1(\tau), 0) d\tau + \int_0^t e_2(\tau) \cdot \Delta_{e_2}(e_1(\tau), e_2(\tau)) d\tau \\
 &\geq V[e_1(t)] - V[e_1(0)]
 \end{aligned} \tag{8.48}$$

where

$$V[e_1(t)] - V[e_1(0)] = \int_0^t \dot{e}_1(\tau) \cdot \Phi(e_1(\tau), 0) d\tau = \int_{e_1(0)}^{e_1(t)} \Phi(e_1, 0) de_1$$

$V[e_1(t)]$ is a storage function with $V(0)=0$

Thus, the sectorial fuzzy controller is input–output passive stable in continuous domain. ■

8.2.5 PASSIVITY OF SECTORIAL FUZZY CONTROLLER IN DISCRETE DOMAIN

THEOREM 8.9 (Xu and Shin, 2005)

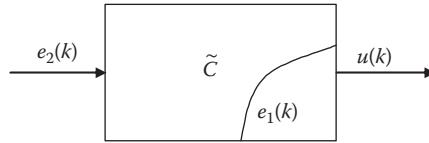
A sectorial fuzzy controller is input–output passive stable in discrete-time domain.

Proof In discrete-time case of fuzzy control design, $e_1(k)$ represents the error $e(k)$, $e_2(k)$ represents the change of the error $\Delta e(k)$, and $u(k)$ is the control action $\Phi(e_1, e_2)$. Thus, the fuzzy control system can be defined as

$$e_1(k) = e_1(k - 1) + e_2(k)$$

$$u(k) = \Phi(e_1(k), e_2(k))$$

Similarly, the fuzzy controller can be considered as a SISO nonlinear system with $e_2(k)$ as the input, $u(k)$ as the output, and $e_1(k)$ as the state, the fuzzy control system should have $u(k)=0$ as the equilibrium point and is shown as follows:



Property 8.1b and e leads to

$$0 \leq e_1(k) \cdot \Phi(e_1(k), 0) \leq \lambda' e_1^2(k)$$

$$0 \leq e_2(k) \cdot \Phi(0, e_2(k)) \leq \gamma' e_2^2(k)$$

$$\text{Let } \Delta_{e_2}(e_1(k), e_2(k)) = \Phi(e_1(k), e_2(k)) - \Phi(e_1(k), 0)$$

$$\Delta_{e_1}(e_1(k), e_2(k)) = \Phi(e_1(k), e_2(k)) - \Phi(0, e_2(k))$$

Then it can be found that

$$0 \leq e_2(k) \Delta_{e_2}(e_1(k), e_2(k)) \leq \gamma' e_2^2(k)$$

$$0 \leq e_1(k) \Delta_{e_1}(e_1(k), e_2(k)) \leq \lambda' e_1^2(k)$$

Applying the definition of passivity results in

$$\begin{aligned}
e_2(k) \cdot u(k) &= e_2(k) \cdot \Phi(e_1(k), e_2(k)) \\
&= e_2(k) \cdot \Phi(e_1(k), 0) + e_2(k) \cdot \Delta_{e_2}(e_1(k), e_2(k)) \\
&\geq e_2(k) \cdot \Phi_1(e_1(k), 0) = e_2(k) \cdot \frac{\sum_{k=i}^{i+1} U_{f(k,0)}^1 \cdot \mu_{E_k}[e_1(k)]}{\sum_{k=i}^{i+1} \mu_{E_k}[e_1(k)]} = e_2(k) \cdot \frac{\sum_{k=i}^{i+1} U_{f(k,0)}^1 \cdot \mu_{E_k}[e_1(k)]}{1} \\
&= e_2(k) \cdot \left\{ \sum_{k=i}^{i+1} U_{f(k,0)}^1 \cdot \mu_{E_k}[e_1(k)] \right\} \\
&= e_2(k) \cdot \left\{ U_{f(i+1,0)}^1 - \mu_{E_i}[e_1(k)] [U_{f(i+1,0)}^1 - U_{f(i,0)}^1] \right\} \\
&\geq e_2(k) \cdot \left\{ U_{f(i+1,0)}^1 - [U_{f(i+1,0)}^1 - U_{f(i,0)}^1] \right\} \\
&= e_2(k) \cdot U_{f(i,0)}^1 = U_{f(i,0)}^1 \cdot e_2(k) = U_{f(i,0)}^1 \cdot [e_1(k) - e_1(k-1)] \\
&= U_{f(i,0)}^1 \cdot e_1(k) - U_{f(i,0)}^1 \cdot e_1(k-1) \\
&= V[e_1(k+1)] - V[e_1(k)]
\end{aligned} \tag{8.49}$$

where $V[e_1(k)]$ is the storage function with $V(0)=0$. Thus, the sectorial fuzzy controller is input–output passive stable in discrete-time domain. ■

8.3 CONCLUSION

The issue of stability analysis is nontrivial in the fuzzy control design because of the presence of various uncertainties, which is caused by numerous possibilities for implementation of linguistic rules, shapes of fuzzy MFs, logic operators, and defuzzification procedures. Therefore, it is necessary to consider the stability of the closed-loop system such that through the feedback interconnection of the plant and the controller, the stability of the complete system as well as some dynamic performances is achieved. This chapter summarized two useful techniques, Lyapunov analysis method and passivity approach, in facilitating nonlinear analysis for fuzzy control systems.

REFERENCES

- Calcev, G., A passivity result for fuzzy control systems, *Proceedings of the 35th Conference on Decision and Control*, Kobe, Japan, pp. 2727–2728, 1996.
- Calcev, G., Some remarks in the stability of Mamdani fuzzy control systems, *IEEE Transactions on Fuzzy Systems*, 6(3): 436–442, 1998.
- Calcev, G., Gorez, R., and De Neyer, M., Passivity approach to fuzzy control systems, *Automatica*, 34(3): 339–344, 1998.

- Ferrari-Trecate, G., Cuzzola, F.A., Mignone, D., and Morari, M., Analysis of discrete-time piecewise affine and hybrid systems, *Automatica*, 38(12): 2139–2146, 2002.
- Hwang, C.-J. and Yeh, T.-T., A design of fuzzy self-organizing controller, *IEEE International Conference on Fuzzy Systems*, New Orleans, LA, Vol. 3, pp. 1567–1572, 1996.
- Kwong, W.A. and Passino, K.M., Dynamically focused fuzzy learning control, *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 26(1): 53–74, 1996.
- López, E.M.N., Dissipativity and passivity-related properties in nonlinear discrete-time systems, PhD dissertation, Universidad Politécnica de Cataluña, Barcelona, 2002.
- Siettos, C.I., Boudouvis, A.G., and Bafas, G.V., Implementation and performance of a fuzzy adaptive controller for a tubular reactor, *Journal of Systems Analysis-Modelling-Simulation*, 38: 725–739, 2000.
- Slotine, J.-J.E. and Li, W., *Applied Nonlinear Control*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1991.
- Tanaka, K. and Sugeno, M., Stability analysis of fuzzy systems using Lyapunov's direct method, *Proceedings of North American Fuzzy Information Processing Society (NAFIPS)*, New York, pp. 133–136, 1990.
- Tanaka, K. and Sugeno, M., Stability analysis and design of fuzzy control systems, *Fuzzy Sets and Systems*, 45: 135–156, 1992.
- Ustundag, B., Eksin, I., and Bir, A., A new approach to global optimization using a closed loop control system with fuzzy logic controller, *Advances in Engineering Software*, 33: 309–318, 2002.
- Xu, C. and Shin, Y.C., Design of a multi-level fuzzy controller for nonlinear systems and stability analysis, *IEEE Transactions on Fuzzy Systems*, 13(6): 761–778, 2005.

9 Intelligent Control for SISO Nonlinear Systems

In this chapter, a multilevel fuzzy control (MLFC) system is developed and implemented to deal with the real-world nonlinear plants with intrinsic uncertainties and time-varying parameters. The MLFC strategy has a hierarchical structure with an adaptation mechanism embedded in the lower level to tune the output membership functions (MFs) of the first layer fuzzy controller and can be used to control a system with an input–output monotonic relationship or a piecewise monotonic relationship. The stability of the resultant closed-loop system is theoretically proven in both continuous and discrete domains. A series of simulations carried out on different uncertain nonlinear systems are shown to demonstrate the effectiveness of the proposed control method in the presence of unknown external disturbance and model parameter variation. Finally, experimental implementation results of the MLFC are shown for a creep-feed grinding process and an end-milling process to maintain a constant force and achieve a maximum metal removal rate (MRR).

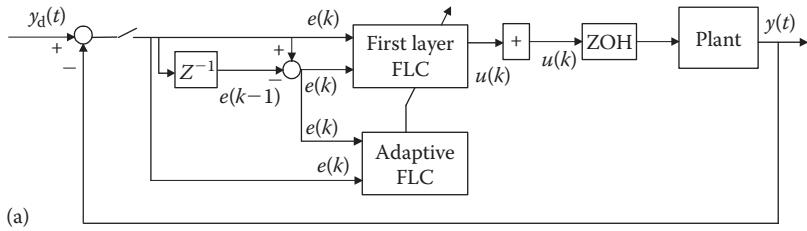
9.1 FUZZY CONTROL SYSTEM DESIGN

Fuzzy control is an appealing alternative to conventional control methods when systems follow some general operating properties and an accurate model of the plant is difficult to obtain. In this case, the system dynamics is captured qualitatively based on operator's experience and knowledge rather than by a mathematical model. This qualitative idea can easily be implemented using a fuzzy rule inference mechanism.

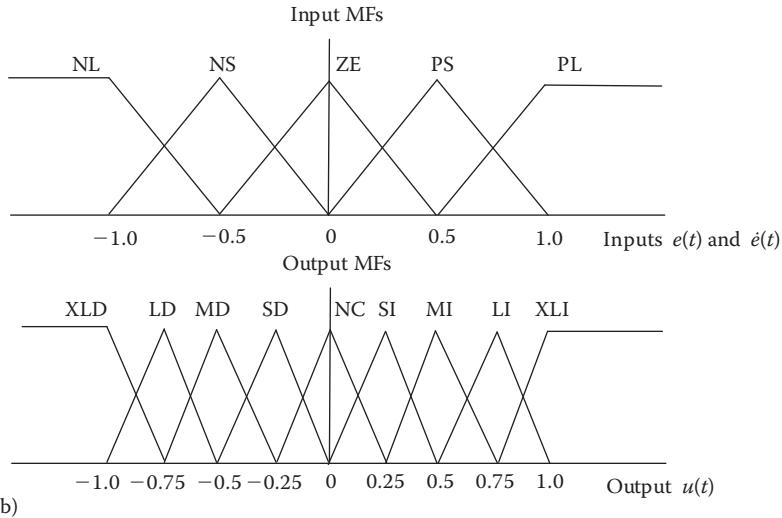
The MLFC developed here is a two-input single-output controller with a two-level, hierarchical structure as shown in [Figure 9.1a](#). The first layer fuzzy control rules are generated based on the experience of human operators. In order to correct the imprecision of the first layer fuzzy control rules and compensate for the time-varying behavior or system uncertainty, a self-organizing fuzzy controller is used to tune the output MFs of the first layer fuzzy controller based on the evaluation of system performance.

9.1.1 FIRST LAYER FUZZY CONTROLLER

The first layer fuzzy controller is a basic two-input single-output controller as shown in [Figure 9.1a](#). The error $e(k)$ and the change of the error $\Delta e(k)$ are actual values in form of crisp numbers. Fuzzification converts the numerical value into a linguistic variable, which can be understood by the fuzzy control system. Singleton fuzzification is



(a)



(b)

FIGURE 9.1 (a) MLFC system and (b) fuzzy MFs for inputs and output.

adopted in this fuzzy control scheme, which transforms a crisp value into a fuzzy singleton value. The two inputs are scaled by multiplying suitable scaling factors (GE and GC) based on the system performance as

$$\hat{e}(k) = e(k) \cdot GE = [y_d(k) - y(k)] \cdot GE \quad (9.1)$$

$$\Delta\hat{e}(k) = \Delta e(k) \cdot GC = [e(k) - e(k-1)] \cdot GC \quad (9.2)$$

where

$y_d(k)$ is the desired output at time instant k

$y(k)$ is the actual output

$e(k)$ is the actual error

$\Delta e(k)$ is the actual change of the error

GE is the scaling factor of the error

GC is the scaling factor of the change of error, which is explained in [Section 9.1.3](#)

Triangular MFs are defined over the ranges of input and output space, which linguistically describe the variable's universe of discourse as shown in Figure 9.1b. The universe of discourse for all inputs and output are all normalized in the interval

of $[-1, 1]$. The linguistic variables that represent the fuzzy sets of the input space have been classified as NL (negative large), NS (negative small), ZE (zero), PS (positive small), and PL (positive large). The linguistic variables that represent sets in the output space have been chosen as XLD (extra largely decrease), LD (largely decrease), MD (moderately decrease), SD (slightly decrease), NC (no change), SI (slightly increase), MI (moderately increase), LI (largely increase), and XLI (extra largely increase). The left and right halves of the triangle MFs for each linguistic label are chosen to provide MFs with equal overlaps with adjacent MFs. For both the input and output variables, MFs are defined to be symmetric, equi-spaced with an equal area so that each set can be described by its central value.

Selection of the number of MFs and their shapes are based on the process knowledge and intuition. The main rule applied in the proposed MLFC scheme is to define the fuzzy partitions over the entire operating ranges of the process variable and also to use enough number of MFs to provide adequate resolution.

Fuzzy control rules are composed of a series of fuzzy if-then rules where the conditions and the consequences are linguistic variables. This collection of fuzzy rules simplifies the input-output relation of the system in linguistic form as

$$R: \text{IF } e_1 \text{ is } E_i \text{ AND } e_2 \text{ is } E_j, \text{ THEN } u_1 \text{ is } U_{n(i,j)} \quad (9.3)$$

where

- e_1 is the error $e(k)$ at time instant k
- e_2 is the change of the error $\Delta e(k)$
- u_1 is the change of the control signal $\Delta u(k)$
- E_i is the linguistic variable of e_1
- E_j is the linguistic variable of e_2
- $U_{n(i,j)}$ is the linguistic variable of u_1

Further, assume that $n(i, j)$ represents any function whose value at i and j is an integer (Ying, 1994; Calcev, 1998; Aracil and Gordillo, 2000; Farinwata et al., 2000).

The knowledge of the increase/decrease relationship between the system input and the output will be used to construct the fuzzy control rule base. For example, if an increase in the system input will result in a decrease in the system output, the fuzzy control rule base will be constructed as shown in **Table 9.1**, where 11 uniformly spaced triangular MFs are defined for each controller input in the first layer fuzzy controller. Thus, there are a total of 121 control rules determined and stored in the rule base. The entries of the table represent the center values of output MFs corresponding to each fuzzy rule in the normalized universe of discourse. For instance, in the shaded element in Table 9.1, the rule indicates that “if error is 0.2 (i.e., y_d is greater than y by 0.2) and the change of the error is 0.4 (i.e., $y_d - y$ is increased by 0.4), then the control action u needs to be decreased by -0.6 .” On the other hand, if an increase in the system input will make the system output increase, the fuzzy control rule base will be established as the counterpart to Table 9.1, where each entry in the table will have the negative value to that in Table 9.1.

Mamdani’s minimum fuzzy implication and center average defuzzification are adopted. The fuzzy control law of the first layer fuzzy controller is shown below:

TABLE 9.1
Fuzzy Rule Table in the First Layer Fuzzy Controller

		Change in Error										
		-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
Error		-1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.6	0.3	0.1	0.0
	-0.8	1.0	1.0	1.0	1.0	1.0	0.8	0.6	0.3	0.1	0.0	-0.1
	-0.6	1.0	1.0	1.0	1.0	0.8	0.6	0.3	0.1	0.0	-0.1	-0.3
	-0.4	1.0	1.0	1.0	0.8	0.6	0.3	0.1	0.0	-0.1	-0.3	-0.6
	-0.2	1.0	1.0	0.8	0.6	0.3	0.1	0.0	-0.1	-0.3	-0.6	-0.8
	0.0	1.0	0.8	0.6	0.3	0.1	0.0	-0.1	-0.3	-0.6	-0.8	-1.0
	0.2	0.8	0.6	0.3	0.1	0.0	-0.1	-0.3	-0.6	-0.8	-1.0	-1.0
	0.4	0.6	0.3	0.1	0.0	-0.1	-0.3	-0.6	-0.8	-1.0	-1.0	-1.0
	0.6	0.3	0.1	0.0	-0.1	-0.3	-0.6	-0.8	-1.0	-1.0	-1.0	-1.0
	0.8	0.1	0.0	-0.1	-0.3	-0.6	-0.8	-1.0	-1.0	-1.0	-1.0	-1.0
	1.0	0.0	-0.1	-0.3	-0.6	-0.8	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0

$$u_1 = \frac{\sum_{i,j} \{ [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)] \cdot U_{n(i,j)} \}}{\sum_{i,j} [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)]} \times GU \quad (9.4)$$

where GU is the scaling factor of the change of the control action. $\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)$ denotes the intersection of fuzzy sets $\mu_{E_i}(e_1)$ and $\mu_{E_j}(e_2)$, which is defined as the *min* operation between two fuzzy sets:

$$\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2) = \min[\mu_{E_i}(e_1), \mu_{E_j}(e_2)] \quad (9.5)$$

The control action is obtained in an incremental form as

$$u(k) = u(k - 1) + \Delta u(k) \quad (9.6)$$

The fuzzy controller is implemented in a discrete-time form using the zero-order hold (ZOH), which holds its input signal for a specified sampling period to get a continuous signal out.

9.1.2 SELF-ORGANIZING FUZZY CONTROLLER

A nonadaptive controller requires an accurate model and the success of the control performance in general depends on the accuracy of the model. In order to compensate for the effects of model uncertainties, unexpected disturbances, time-varying parameters, and sensor noises, the rule base in the first layer fuzzy controller needs to be updated in real time to achieve better performance. A self-organizing fuzzy controller is embedded in the lower level fuzzy controller, and is also designed as a two-input single-output controller to minimize the number of fuzzy rules. The inputs are the same as the inputs of the first layer fuzzy controller. Output will be

used to tune the center of the output MFs of the first layer fuzzy controller. The shape of MFs and linguistic variables are kept same as those of the first layer fuzzy controller for easy computation.

The self-organizing fuzzy control rule is in the form of

$$R: \text{IF } e_1 \text{ is } E_i \text{ AND } e_2 \text{ is } E_j, \text{ THEN } C \text{ is } C_{m(i,j)} \quad (9.7)$$

where

e_1 , e_2 , E_i , and E_j have the same meanings as the description in the first layer fuzzy controller

C is the change of the center of the output MF of the first layer fuzzy controller

$C_{m(i,j)}$ is the linguistic variable of C

$m(i,j)$ shares the same property as $n(i,j)$ in the first layer fuzzy controller

The fuzzy rules of the self-organizing fuzzy controller are formulated in terms of linguistic statement and can be transformed into a lookup decision table as shown in Table 9.2. The lookup table can be treated like a human expert who makes the appropriate corrections of the first layer fuzzy controller based on the current value of the error and the change of the error. Based on the effectiveness of the last control action on the error reduction, the contribution of each activated fuzzy rule in the last step can be evaluated and modified accordingly. For example, the rule in the shaded area indicates that “if error is 0.2 (i.e., y_d is greater than y by 0.2) and the change of the error is 0.4 (i.e., $y_d - y$ is increased by 0.4), then the centers of the corresponding output MFs of the first layer fuzzy controller need to be decreased by -0.3.”

The zero entries in Table 9.2 mean that no correction is needed in the rule base of the first layer fuzzy controller. The entries near the center position have smaller values, which ensure a sufficiently fast convergence rate and a good damping ratio without overshoot when close to the set point. The further away from the desired

TABLE 9.2
Fuzzy Rule Table in the Self-Organizing Fuzzy Controller

		Change in Error										
		-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
Error	-1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.0	0.0	0.0	0.0	0.0
	-0.8	1.0	1.0	1.0	1.0	0.8	0.6	0.0	0.0	0.0	0.0	0.0
	-0.6	1.0	1.0	1.0	0.8	0.6	0.3	0.0	0.0	0.0	0.0	0.0
	-0.4	1.0	1.0	0.8	0.6	0.3	0.1	0.0	0.0	0.0	0.0	0.0
	-0.2	1.0	0.8	0.6	0.3	0.1	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.8	0.6	0.3	0.1	0.0	0.0	0.0	-0.1	-0.3	-0.6	-0.8
	0.2	0.0	0.0	0.0	0.0	0.0	0.0	-0.1	-0.3	-0.6	-0.8	-1.0
	0.4	0.0	0.0	0.0	0.0	0.0	-0.1	-0.3	-0.6	-0.8	-1.0	-1.0
	0.6	0.0	0.0	0.0	0.0	0.0	-0.3	-0.6	-0.8	-1.0	-1.0	-1.0
	0.8	0.0	0.0	0.0	0.0	0.0	-0.6	-0.8	-1.0	-1.0	-1.0	-1.0
	1.0	0.0	0.0	0.0	0.0	0.0	-0.8	-1.0	-1.0	-1.0	-1.0	-1.0

trajectories, the greater the entries' values are, which means the greater amount of the system output needs to be corrected. It can also be seen that the table is symmetric about the set point, as a positive error means that the output requires to be increased, while a negative error represents the requirement of decreasing the output.

Note that in [Table 9.2](#), the entries of the upper-right corner and the lower-left corner are kept as zero, which intends to reduce the probability of the oscillation in the output response. For example, when the system state falls in the upper-right corner, the error is negative and the change of the error is positive, which means that the error will decrease (in magnitude) from the initially negative value and will approach zero. This is just the desired moving trend for the system performance, and then there is no need to modify the first layer fuzzy control rules. Because all the learning information is embedded in the second layer fuzzy controller, this learning process will automatically construct an optimal fuzzy rule base that covers the appropriate operating range under various system conditions.

Using Mamdani's minimum fuzzy implication and center average defuzzification, the center of output MFs of the first layer fuzzy controller should be tuned as

$$U_{n(i,j)} \rightarrow U_{n(i,j)} + \frac{\sum_{i,j} \{ [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)] \cdot C_{m(i,j)} \}}{\sum_{i,j} [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)]} \quad (9.8)$$

In this way, an improved system performance is achieved and a limited number of fuzzy rules are adequate for the nonlinear and time-varying plants.

9.1.3 ONLINE SCALING FACTOR DETERMINATION SCHEME

Procyk and Mamdani (1979) and Lee (1990) have shown the scaling factors of the inputs and outputs have a significant impact on the performance of the resultant control system. Since most of real-world systems are highly nonlinear and time-varying, fixed scaling factors of the fuzzy controller may not necessarily suit every case. The different values of the scaling factors need to be determined to guarantee the system performance in the entire operating range.

As illustrated in [Figure 9.1b](#), the universes of discourse for all inputs and output are normalized within the unit range. The scaling factors will act as the gain of the expansion or contraction of the horizontal axes. Changing the gains will rescale the axis, which is analogous to the changes of the proportional/integral/derivative gain in a PID (proportional-integral-derivative) controller. In general, increasing the scaling factors of the error and the change of error (GE and GC) will reduce the rise-time while making the system performance to be sensitive around the set point. Decreasing the two factors has the opposite effect. On the other hand, a small output scaling factor of the change of the control action (GU) will extend the rise-time and result in a sluggish response (Passino and Yurkovich, 1998).

Researchers have established principles on how to determine the scaling factors heuristically without any mathematical process model (Linkens and Abbot, 1992; Hsu and Fann, 1996; Passino and Yurkovich, 1998). When the actual output is within the

tolerance band around the reference output, a relatively small scaling factor of error GE is designed so that a relative small control action is generated to reduce the possibility of severe oscillation around the reference output. In contrast, when the actual output is far away from the reference value, a relatively larger scaling factor GE can be chosen to produce a large control action to speed up the response of the system. The determination of the change of the error GC is based on the system dynamics, in other words, the maximum sensitivity of the system. It has a great effect on the control sensitivity, especially in transient time. A large value of GC makes the control action change drastically in each loop of control execution, and the system response will be faster and more unstable. The scaling factor for the change of the control action GU is the overall gain factor of the control system, which is specified by how much of the control action can be changed within the normal operating range. It sets a trade-off between the system response speed and its stability as stated by Haber et al. (2000).

Through these online scaling factor determination schemes, a satisfactory system response with a short transient time and small oscillation around the set point is guaranteed for nonlinear time-varying real-world systems under different working conditions.

9.2 STABILITY ANALYSIS

In this section, the passivity stability proof is presented for the proposed MLFC strategy. The analysis proves that, for a general nonlinear system, as long as its augmented system is passive stable, the resultant closed-loop system remains stable with relaxed-passivity condition in both continuous and discrete time domains.

9.2.1 MULTILEVEL FUZZY CONTROL STRUCTURE

Combining Equations 9.4 and 9.8 yields

$$\begin{aligned}
 u_1 &= \frac{\sum_{i,j} \left\{ [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)] \cdot U_{n(i,j)} \right\}}{\sum_{i,j} [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)]} \rightarrow \\
 u_1 &= \frac{\sum_{i,j} \left[[\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)] \cdot \left(U_{n(i,j)} + \frac{\sum_{i,j} \left\{ [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)] \cdot C_{m(i,j)} \right\}}{\sum_{i,j} [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)]} \right) \right]}{\sum_{i,j} [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)]} \\
 &= \frac{\sum_{i,j} \left\{ [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)] \cdot U_{n(i,j)} \right\}}{\sum_{i,j} [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)]} + \frac{\sum_{i,j} \left\{ [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)] \cdot C_{m(i,j)} \right\}}{\sum_{i,j} [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)]} \\
 &= \Phi_1(e_1, e_2) + \Phi_2(e_1, e_2)
 \end{aligned} \tag{9.9}$$

where

$$\Phi_1(e_1, e_2) = \frac{\sum_{i,j} \left\{ [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)] \cdot U_{n(i,j)} \right\}}{\sum_{i,j} [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)]} \quad (9.10)$$

$$\Phi_2(e_1, e_2) = \frac{\sum_{i,j} \left\{ [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)] \cdot C_{m(i,j)} \right\}}{\sum_{i,j} [\mu_{E_i}(e_1) \wedge \mu_{E_j}(e_2)]} \quad (9.11)$$

Note that Equations 9.4 and 9.8 share the same form of fuzzy control law. It is observed that $\Phi_1(e_1, e_2)$ represents the first layer fuzzy controller, while $\Phi_2(e_1, e_2)$ represents the self-organizing fuzzy controller in the same way.

THEOREM 9.1 (Slotine, 1991; Farinwata et al., 2000; Khalil, 2002)

Any parallel combination of two passive subsystems is also passive, the storage function of the complete system being simply the sum of the individual storage functions of the two subsystems. ■

THEOREM 9.2 (Slotine, 1991; Calcev, 1998; Farinwata et al., 2000; Khalil, 2002)

A sufficient condition for asymptotic stability of the fuzzy control closed-loop is the input–output passivity of the system itself. ■

9.2.2 STABILITY ANALYSIS IN CONTINUOUS-TIME CASE

In continuous case of MLFC design, e_1 represents the error $e(t)$, e_2 represents the change of the error $\dot{e}(t)$, and u_1 is the change of control signal $\dot{u}(t)$. Thus, the fuzzy control system can be defined as

$$\begin{aligned} \dot{e}_1 &= e_2 \\ u_1 &= \Phi_1(e_1, e_2) + \Phi_2(e_1, e_2) \end{aligned} \quad (9.12)$$

Then, the MLFC can be considered as a single-input single-output (SISO) nonlinear system with internal dynamics, where e_2 is the input, u_1 is the output, and e_1 is the state. As derived by Xu and Shin (2005), the MLFC C_1 in Figure 9.2a is input–output passive in continuous domain. From Theorem 9.1, with $u(t) = \int \dot{u}(\sigma)d\sigma$, the fuzzy controller C in Figure 9.2b is a parallel combination of fuzzy controller C_1 over the operating period. Thus, it is also passive with control inputs as $e(t)$ and $\dot{e}(t)$, and control output as $u(t) = \int \dot{u}(\sigma)d\sigma$. If the error signal $e(t)$ is defined as the state variable, the fuzzy controller C can be further presented as an equivalent expression \tilde{C} (Figure 9.2c), which is also passive stable. By exchanging the two functional blocks of $\frac{d}{dt}$ and \tilde{C} , the

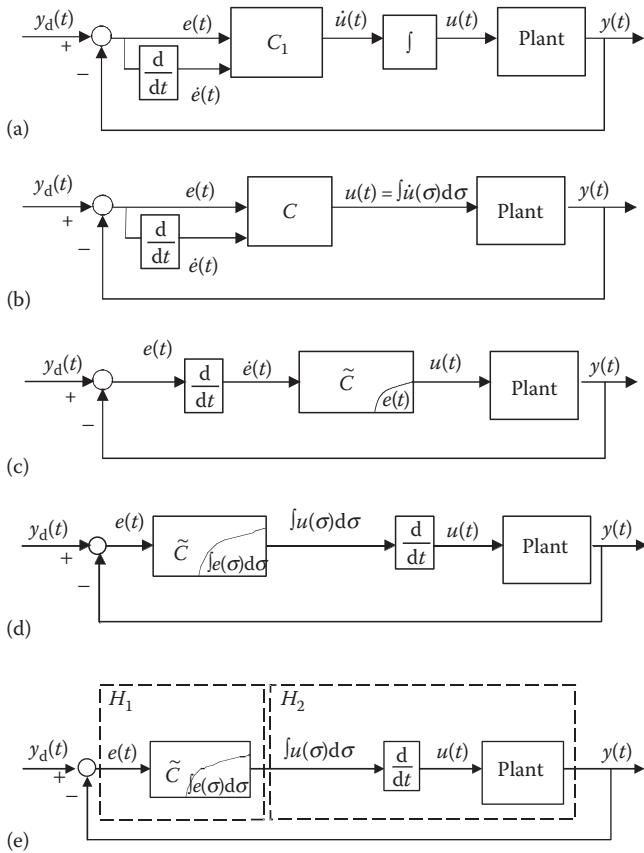


FIGURE 9.2 Closed-loop control system in continuous-time domain: (a) control inputs: $e(t)$ and $\dot{e}(t)$, control output: $\dot{u}(t)$, (b) control inputs: $e(t)$ and $\dot{e}(t)$, control output: $u(t) = \int \dot{u}(\sigma) d\sigma$, (c) control input: $e(t)$ and control output: $u(t)$, state variable: $\dot{e}(t)$, (d) control input: $e(t)$ and control output: $\int u(\sigma) d\sigma$, state variable: $\dot{e}(\sigma) d\sigma$, and (e) closed-loop control system for augmented plant.

closed-loop fuzzy control system can be represented as in Figure 9.2d. Furthermore, specifying \tilde{C} as the controller H_1 and the combination of $\frac{d}{dt}$ and the open-loop plant as the augmented plant H_2 , the fuzzy control closed-loop system is represented in Figure 9.2e. From Theorem 9.2, if two passive subsystems H_1 and H_2 are interconnected in a well-defined negative feedback connection, the closed-loop system is still stable, where H_1 includes the controller and H_2 is the system to be controlled. Thus, the requirement of the closed-loop system stability is the input-output passivity of the augmented system. The relaxed-passivity condition is

$$V[\mathbf{x}(t_f)] - V[\mathbf{x}(t_0)] \leq \int_{t_0}^{t_f} \left[\int u(\sigma) d\sigma \right]^T y(\sigma) d\sigma, \quad \forall \mathbf{x}, \forall u \quad (9.13)$$

which is a much looser requirement compared with the passivity condition in inequality equation (Equation 8.38) in Definition 8.16. As long as the open-loop plant satisfies the inequality equation (Equation 9.13), the closed-loop fuzzy control system is Lyapunov stable.

9.2.3 STABILITY ANALYSIS IN DISCRETE-TIME CASE

In discrete-time case of MLFC design, $e_1(k)$ represents the error $e(k)$, $e_2(k)$ represents the change of the error $\Delta e(k)$, and $u_1(k)$ is the change of control signal $\Delta u(k)$. Thus, the MLFC system can be defined as

$$\begin{aligned} e_1(k) &= e_1(k-1) + e_2(k) \\ u_1(k) &= \Phi_1(e_1(k), e_2(k)) + \Phi_2(e_1(k), e_2(k)) \end{aligned} \quad (9.14)$$

Therefore, the fuzzy controller can be considered as a SISO nonlinear system with $e_2(k)$ as the input, $u_1(k)$ as the output, and $e_1(k)$ as the state. As derived by Xu and Shin (2005), the MLFC C_1 in Figure 9.3a is input-output passive in discrete-time domain,

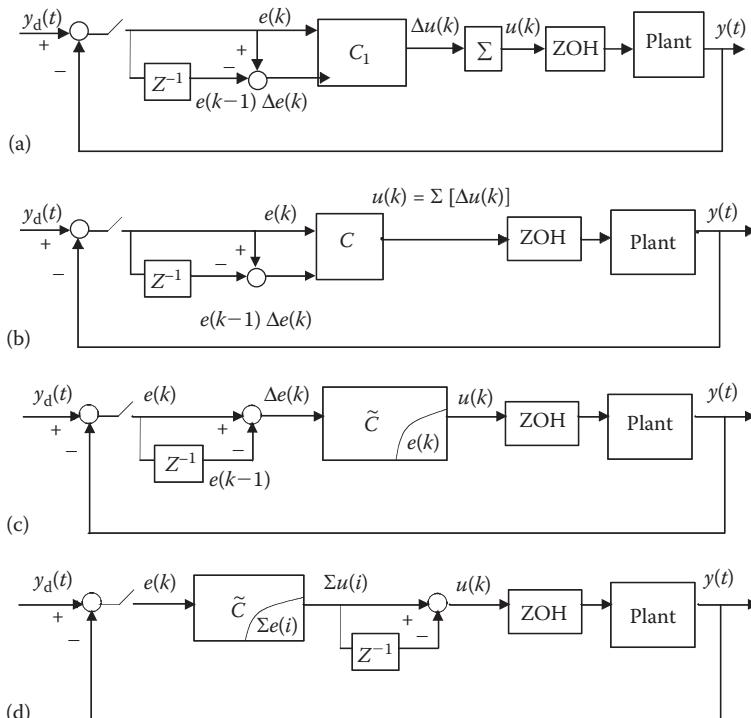


FIGURE 9.3 Closed-loop control system in discrete-time domain: (a) control inputs: $e(k)$ and $\Delta e(k)$, control output: $\Delta u(k)$, (b) control inputs: $e(k)$ and $\Delta e(k)$, control output: $u(k) = \Sigma[\Delta u(k)]$, (c) control input: $\Delta e(k)$, control output: $u(k)$, state variable: $e(k)$, (d) control input: $e(k)$, control output: $\Sigma u(i)$, state variable: $\Sigma e(i)$, and

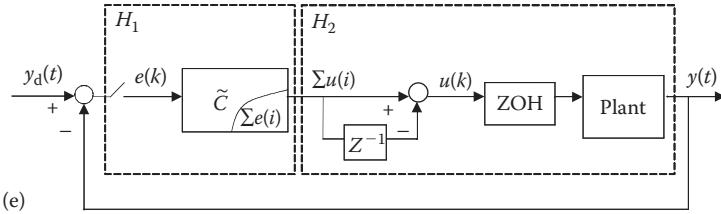


FIGURE 9.3 (continued) (e) closed-loop control system for augmented plant.

with the change of error $\Delta e(k)$ as input and the change of control action $\Delta u(k)$ as output. From Theorem 9.1, with $u(k) = u(k - 1) + \Delta u(k)$, the fuzzy controller C in Figure 9.3b is a parallel combination of fuzzy control C_1 over the operating period. Thus, it is also passive with control inputs as $e(k)$ and $\Delta e(k)$, and control output as $u(k) = \Sigma [\Delta u(k)]$. Denote \tilde{C} as an equivalent expression of the fuzzy controller C with $\Delta e(k)$ as the input, $u(k)$ as the output, and $e(k)$ as the state variable, which is also passive as shown in Figure 9.3c. Furthermore, the closed-loop system can be transformed as shown in Figure 9.3d. Specify \tilde{C} as the controller H_1 , the other parts as the augmented plant H_2 . The fuzzy control closed-loop system is represented as in Figure 9.3e. From Theorem 9.2, the requirement of the closed-loop system stable is the input–output passivity of the augmented system itself. The relaxed-passivity condition in discrete-time domain is

$$V[\mathbf{x}(k+1)] - V[\mathbf{x}(k)] \leq \left[\sum u(i) \right]^T y(k), \quad \forall \mathbf{x}(k), \forall u(k), \forall k \quad (9.15)$$

which is much looser compared with the passivity condition in inequality equation (Equation 8.42) in Definition 8.17. As long as the plant satisfies inequality equation (Equation 9.15), the closed-loop fuzzy control system is Lyapunov stable in discrete domain as well.

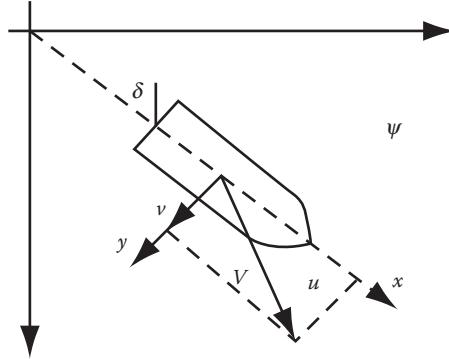
9.3 SIMULATION EXAMPLES

In order to illustrate the effectiveness of the proposed control method, it is useful to compare the performance of the closed-loop system with the MLFC and the conventional fuzzy logic controller (FLC). Computer simulations applied to the control of three nonlinear uncertain systems are provided in this section. It will be found that short transient time is achieved and steady-state errors are reduced tremendously.

9.3.1 CARGO SHIP STEERING

The controlled system considered here is the cargo ship. By applying Newton's laws of motion, the ship dynamics can be obtained. The motion of the ship is described by

a coordinate system that is fixed to the ship as the figure below (Layne and Passino, 1993; Passino and Yurkovich, 1998).



For simplification, the friction and hysteresis of the rudder machine are not included in the model, and the dynamics of the system is expressed as

$$\ddot{\psi}(t) + \left(\frac{1}{\tau_1} + \frac{1}{\tau_2} \right) \dot{\psi}(t) + \left(\frac{1}{\tau_1 \tau_2} \right) H[\dot{\psi}(t)] = \frac{K}{\tau_1 \tau_2} [\tau_3 \dot{\delta}(t) + \delta(t)] \quad (9.16)$$

where

$\psi(t)$ is the heading of the ship

$\delta(t)$ is the rudder angle

K , τ_1 , τ_2 , and τ_3 are parameters that are a function of the ship's constant forward velocity u and its length l

In particular,

$$K = K_0 \left(\frac{u}{l} \right) \quad (9.17)$$

$$\tau_i = \tau_{i0} \left(\frac{l}{u} \right), \quad i = 1, 2, 3 \quad (9.18)$$

where it is assumed that the ship is traveling with a constant velocity of 5 m/s and $l = 350$ m.

$H[\dot{\psi}(t)]$ is a nonlinear function of $\psi(t)$, which can be approximated by

$$H[\dot{\psi}(t)] = \bar{a} \dot{\psi}(t)^3 + \bar{b} \dot{\psi}(t) \quad (9.19)$$

where \bar{a} and \bar{b} are real-valued constants with \bar{a} being positive. In simulations, $\bar{a} = 1$ and $\bar{b} = 1$. The inputs to the fuzzy controller are the heading error and the change of the heading error expressed as

$$e(k) = \psi_r(k) - \psi(k) \quad (9.20)$$

$$c(k) = e(k) - e(k - 1) \quad (9.21)$$

respectively, where

$\psi_r(k)$ is the desired ship heading

$\psi(k)$ is the actual ship heading

and the controller output is the rudder angle $\delta(t)$ of the ship.

The scaling controller gains for the error, change in error, and the controller output are chosen via the design procedure to be $g_e = 1/\pi$ (since the error $e(k)$ can never be over 180°), $g_c = 100$ (since we have found via simulations that the ship does not move much faster than 0.01 rad/s), and $g_u = 8\pi/18$ (since we want to limit δ between $\pm 80^\circ$, we have $g_u = 80\pi/180 = 8\pi/18$) (Passino and Yurkovich, 1998). For a cargo ship, an increase in the rudder angle $\delta(t)$ will generally result in a decrease in the ship heading angle $\psi(k)$.

Case 9.1 Normal Condition with Ballast

This example typifies a cargo ship under the normal condition with ballast. The parameters are $K_0 = 5.88$, $\tau_{10} = -16.91$, $\tau_{20} = 0.45$, and $\tau_{30} = 1.43$. Use a reference input that commands $+40^\circ$ for 3,000 s, 0° for 3,000 s, -40° for 3,000 s, and then commands 0° for the next 3,000 s. Then repeat this 12,000 s sequence for 27,000 s.

The fuzzy control rules are designed according to the system behavior under the nominal condition. When comparing the results for the two control approaches of the FLC and the MLFC as shown in Figures 9.4 and 9.5, although the MLFC shows a little bit better system performance than the FLC in the overshoot and settling time, both system responses converge to the reference input and the performances of the controllers are similar. In fact, the maximum deviation between the two signals is observed to be less than 5° over the entire simulation. In other words, the MLFC and conventional FLC provide about the same performance under the nominal operation condition. However, it will be shown that superior performance can be achieved by the MLFC when certain variations appear in the cargo ship system.

The plots of fuzzy MFs for output in Figures 9.4 and 9.5 illustrate the locations of the output MFs at the end of the simulation time. In order to make the figures clear, only the output MFs corresponding to zero value of change in error are plotted. Each of the 11 triangles is the output MF in the normalized universe of control action, which is the consequence of the 11 input MFs of the error. The plots of rudder angle show the control actions for the cargo ship under normal condition. Since the regular FLC does not have the adaptation mechanism, the control actions are same in the beginning period ($[0-3,000]$ s) and the ending period ($[24,000-27,000]$ s). In contrast, the control action varies in case with the MLFC.

Case 9.2 Cargo Ship without Ballast

The purpose of this simulation example is to demonstrate the adaptive capability of the proposed MLFC with parameter variations in the nonlinear system under control. Consider a cargo ship without a ballast that weighs less than the one under the normal condition. The parameters in the system dynamics specified in Equations 9.16 through 9.18 are changed to $K_0 = 0.83$, $\tau_{10} = -2.88$, $\tau_{20} = 0.38$, and $\tau_{30} = 1.07$ in this case. The desired trajectory is the same as that in Case 9.1. From Figures 9.6 and 9.7, it is clear that the performance of the MLFC is much better than that of the FLC. Since the FLC has no ability to adapt to the plant variations, the response cannot follow the desired trajectory and shows large oscillations during the entire simulation time interval. In contrast, the MLFC can tune itself quickly and show good performance after short transient time. It can be seen

that the overshoot of the MLFC existing in the initial time interval [0–3,000] s can be eliminated successfully in the ending time interval [24,000–27,000] s. This proves that MLFC has the ability of adapting by itself to achieve the final tracking accuracy.

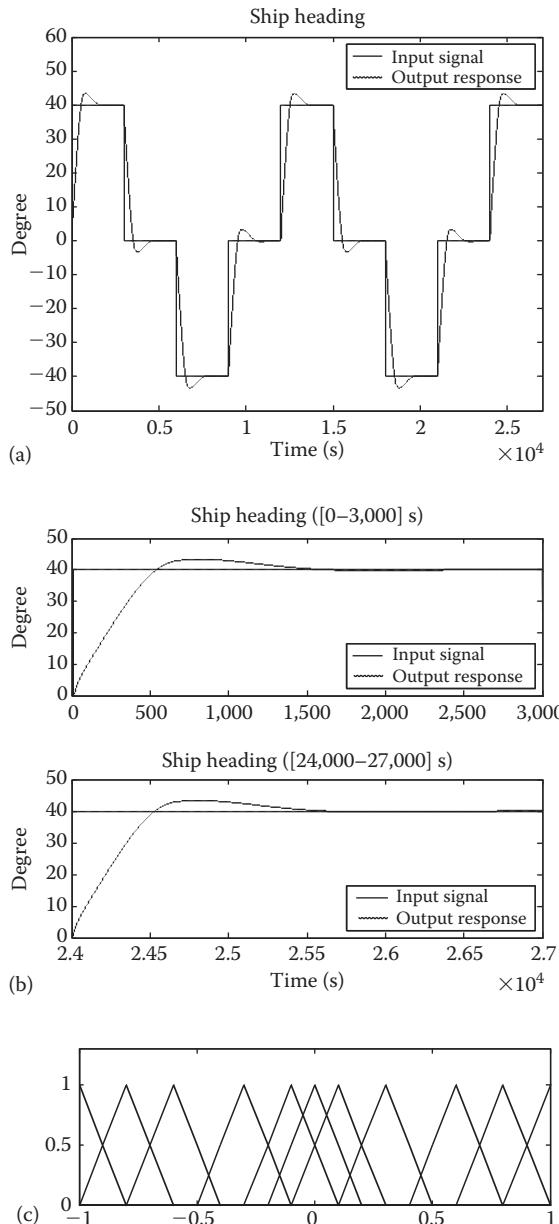


FIGURE 9.4 Simulation result for the FLC under normal condition with ballast: (a) ship heading during the entire operating time, (b) ship heading in the starting and ending periods, (c) fuzzy MFs for output,

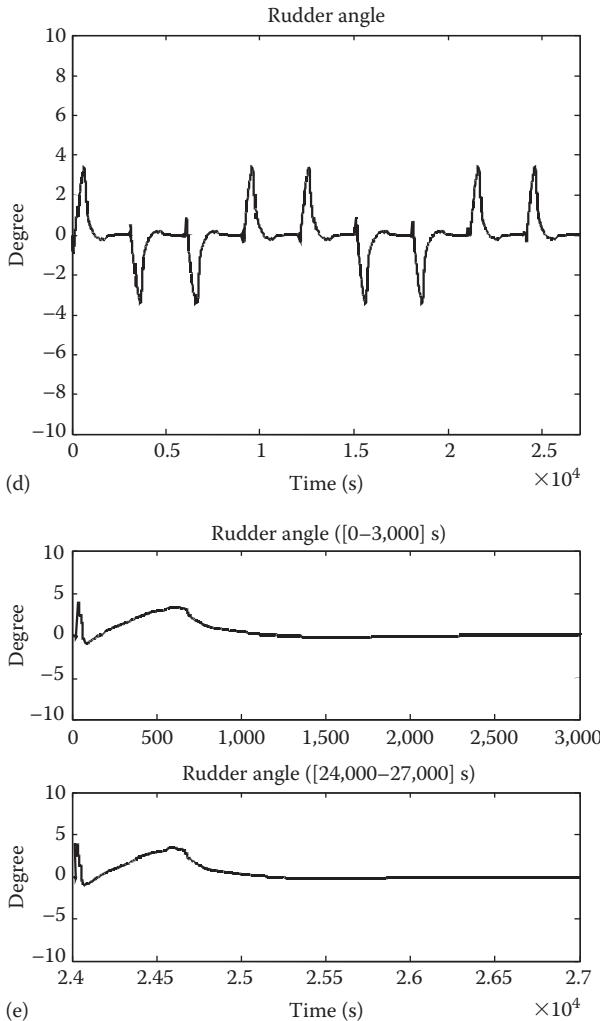


FIGURE 9.4 (continued) (d) rudder angle during the entire operating time, and (e) rudder angle in the starting and ending periods.

Case 9.3 With Wind Disturbance

In the actual process operation, there always exist some kinds of disturbance that will infuse into the system as the input signal. This simulation for the ship steering is designed to illustrate the learning and adaptive capability of the MLFC to compensate for disturbances to the system input. Suppose that a disturbance is introduced to the rudder command δ , which is sent into the plant as the control signal. The effect of this disturbance is similar to that of a gusting wind acting against the body of the ship. Specifically, the disturbance is chosen to be a periodical sine function as shown in [Figure 9.8](#).

$$w(t) = 0.5 \times (\pi/180) \times \sin(0.002\pi t) \quad (9.22)$$

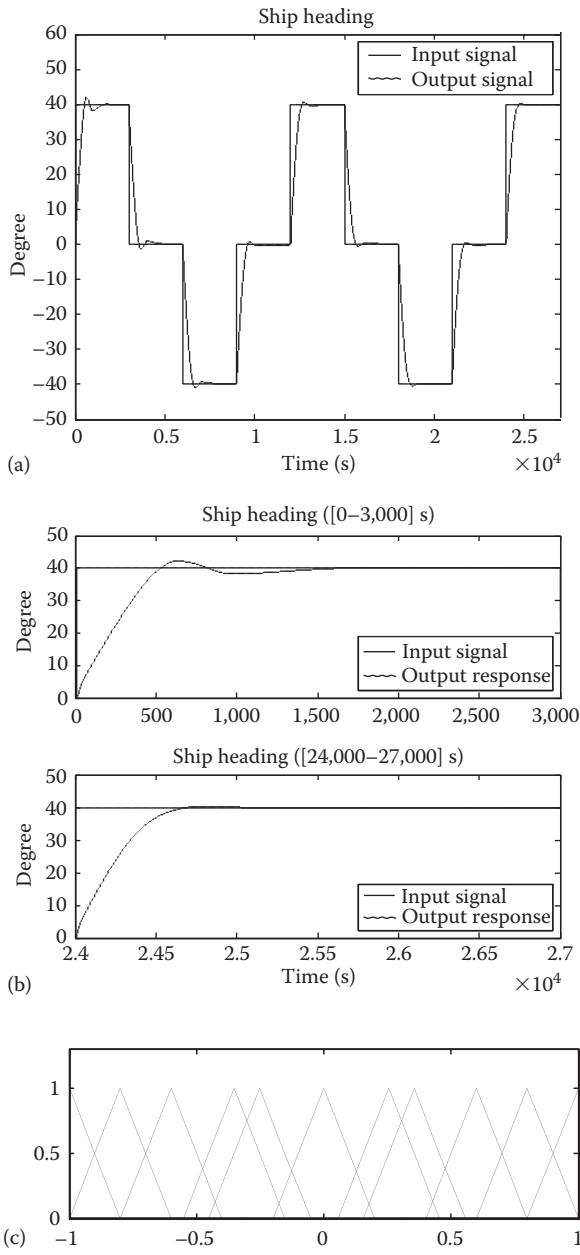


FIGURE 9.5 Simulation result for the MLFC under normal condition with ballast: (a) ship heading during the entire operating time, (b) ship heading in the starting and ending periods, (c) fuzzy MFs for output,

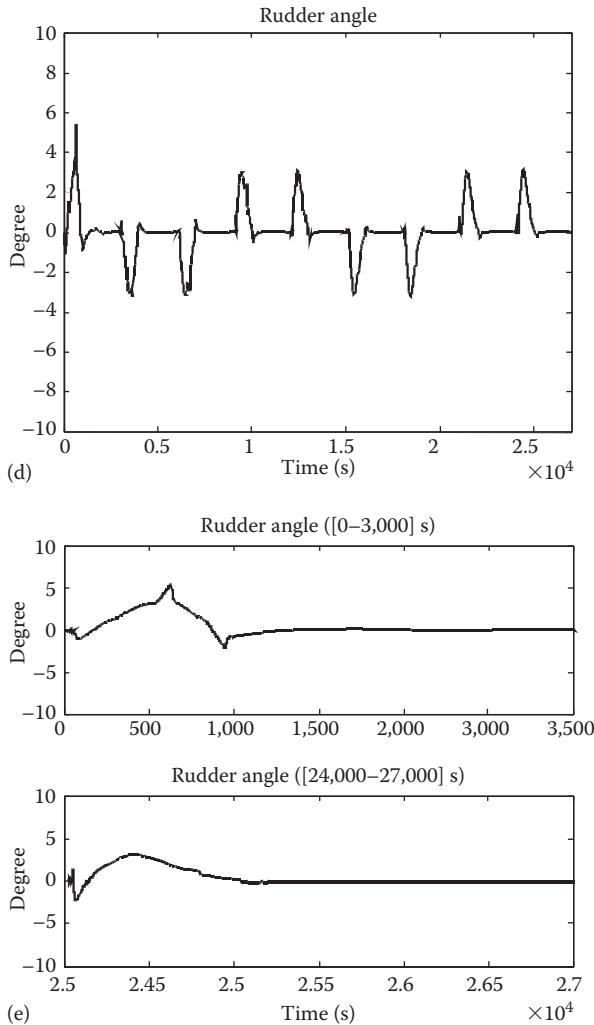


FIGURE 9.5 (continued) (d) rudder angle during the entire operating time, and (e) rudder angle in the starting and ending periods.

For the simulations with both the MLFC and FLC, the same nonlinear plant model is used as in Case 9.1 to emulate the ship's dynamics. In order to display how the MLFC learns to tune the fuzzy control rules, this example provides the simulation of a cargo ship operated under the wind disturbance for 39,000 s, which is a little bit longer than the duration used for the first two examples. In order to clearly show the plot of the wind disturbance in such a high frequency, only several periods of the disturbance during the initial time interval are plotted in [Figure 9.8](#), while the disturbance will repeat the same cycle of sine function during the entire simulation time. [Figures 9.9](#) and [9.10](#) illustrate the results obtained from this simulation, which show that the MLFC is quite successful

in generating the appropriate control rules for a good response since the ship heading tracks the reference model almost perfectly compared to the simulation results for the FLC. Especially, as can be clearly observed from [Figure 9.10](#) that during the initial time

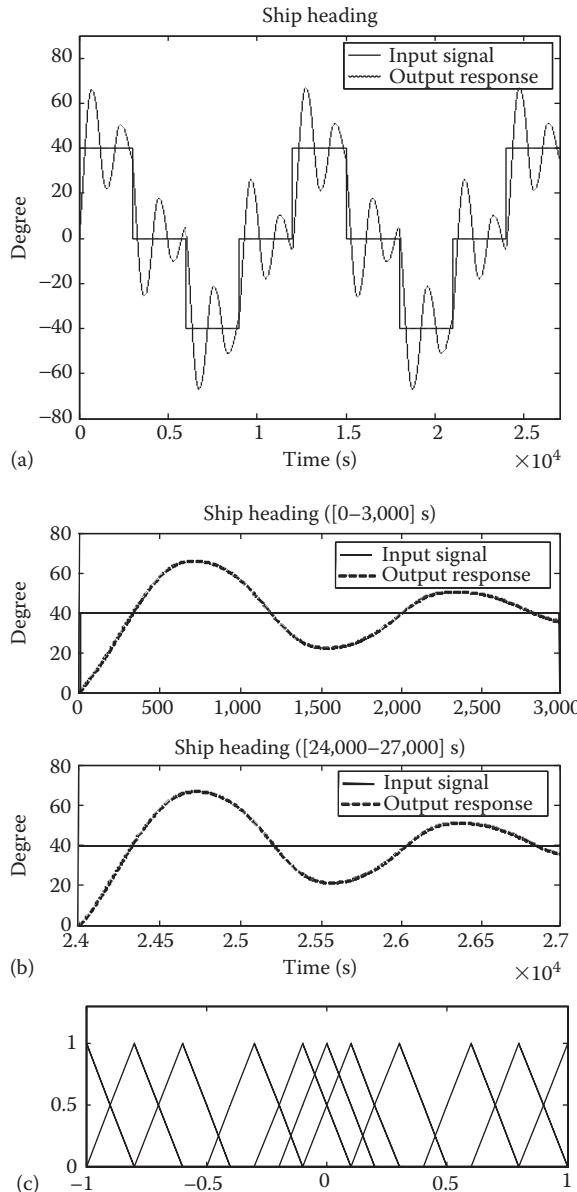


FIGURE 9.6 Simulation result for the FLC without ballast: (a) ship heading during the entire operating time, (b) ship heading in the starting and ending periods, (c) fuzzy MFs for output,

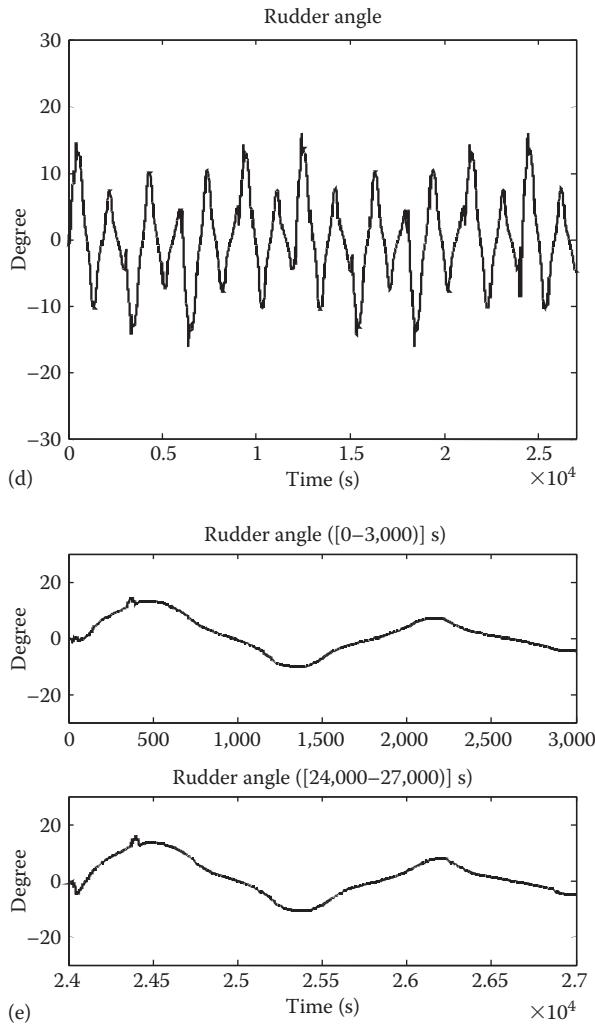


FIGURE 9.6 (continued) (d) rudder angle during the entire operating time, and (e) rudder angle in the starting and ending periods.

interval [0–3,000] s the MLFC shows a certain amount of oscillation and a relative longer settling time, while in the ending time interval [36,000–39,000] s, the MLFC can settle the system response quickly without any overshoot or steady-state error. Thus, the MLFC has the ability to improve the performance and converge fast to the desired trajectory under the existence of disturbance.

Case 9.4 With Sensor Noise

In the feedback control design, usually the system output is measured online and then different control schemes are implemented depending on the calculated system

error or other error functions. The existence of the sensor noise will enlarge the system error, thus deteriorating the control effort. This example considers the same nonlinear dynamic system as in Case 9.1, which is operated with sensor noise in the

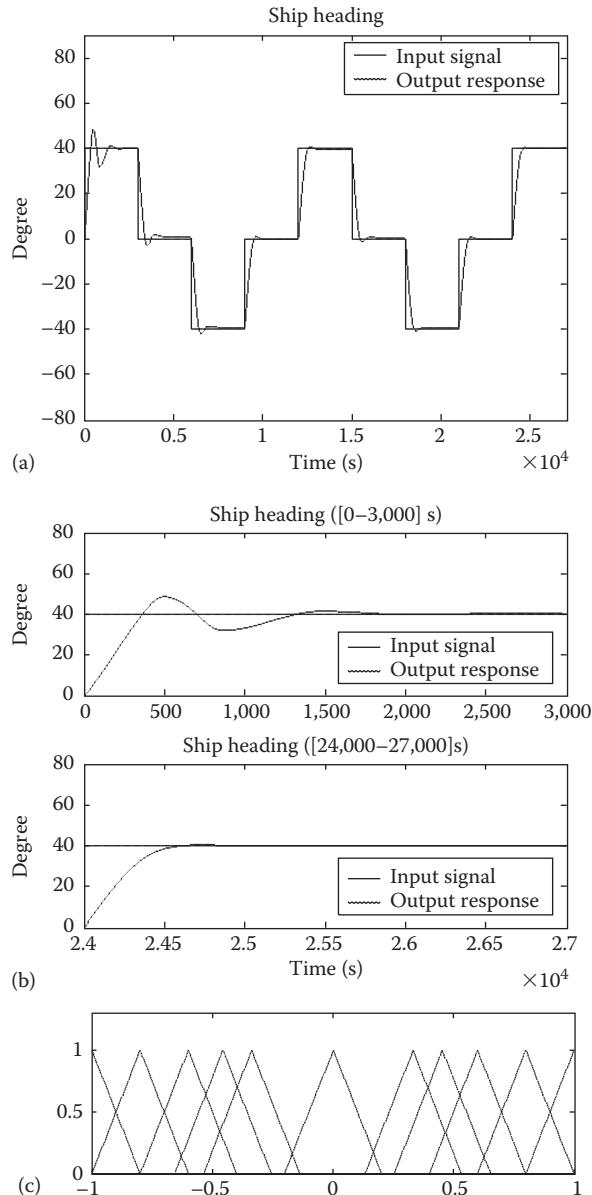


FIGURE 9.7 Simulation result for the MLFC without ballast: (a) ship heading during the entire operating time, (b) ship heading in the starting and ending periods, (c) fuzzy MFs for output,

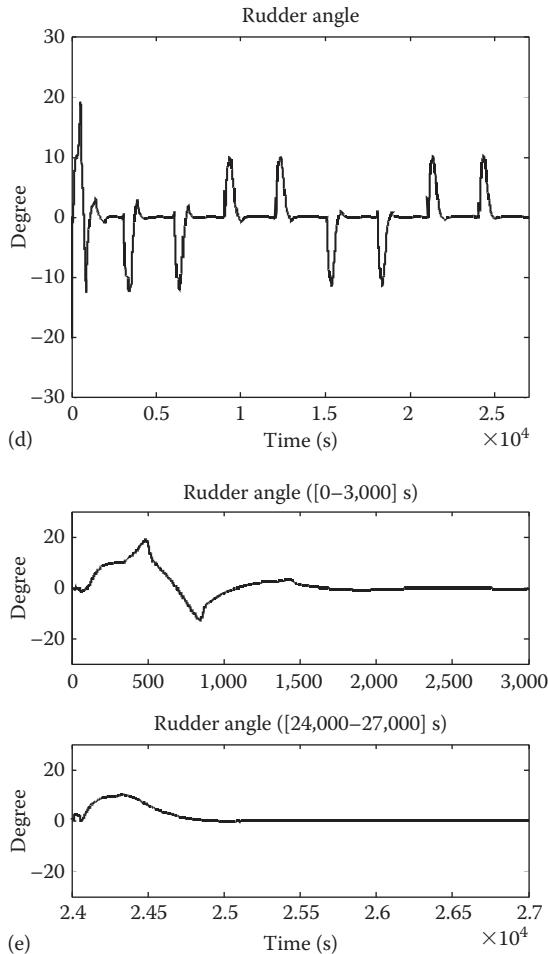


FIGURE 9.7 (continued) (d) rudder angle during the entire operating time, and (e) rudder angle in the starting and ending periods.

heading sensor. The sensor noise is added to the system output as the feedback signal as shown in [Figure 9.11](#).

$$s(t) = 0.01 \times (\pi / 180) \times (2 \times \text{rand} - 1) \quad (9.23)$$

To compare the performance of the MLFC and the FLC, simulations under the specified sensor noise were carried out and the output responses obtained are shown in [Figures 9.12](#) and [9.13](#). Both system responses converge to track the reference model with unnoticeable fluctuation due to the small magnitude of the sensor noise. However, the convergence rate of the FLC is significantly slower than that of the MLFC. As time goes by, the output response of the MLFC has a shorter settling time and a smaller overshoot for every coming step input consequence, which means

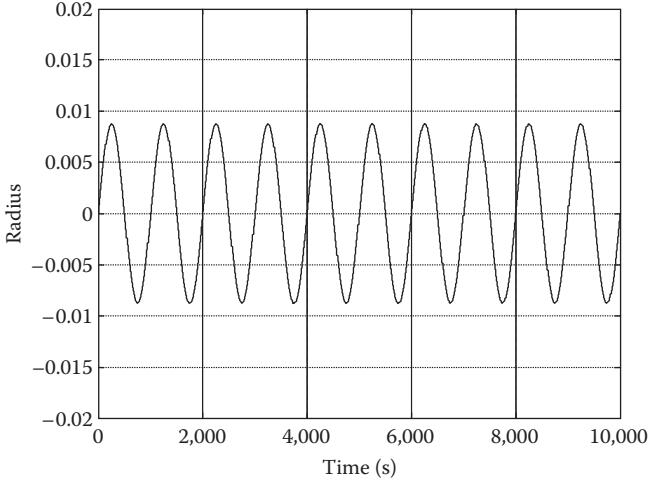


FIGURE 9.8 Wind disturbance at rudder.

that the MLFC has the ability to learn the system in the noisy environment and adjust the control parameters correspondingly. In the ending time interval, the output of MLFC can follow the reference input very quickly without any overshoot. Thus, the MLFC has the learning ability to tune the fuzzy rule location efficiently and then achieves the desired final tracking accuracy very quickly. Since the fuzzy control rules are fixed in the FLC method, the system performance in the ending time interval is the same as that in the initial time interval. Therefore, a judicious choice of initial fuzzy rules is very important in the FLC strategy.

9.3.2 FUZZY CRUISE CONTROL

From the simulation studies in Case 9.1, the good learning capability of the proposed MLFC in the presence of unexpected disturbance or sensor noise is illustrated. In this example, the MLFC will be further applied to a nonlinear system with parameter variations to demonstrate its superiority to the FLC. Here, a fuzzy cruise controller is developed to regulate a vehicle's speed $v(t)$ to a driver-specified value $v_d(t)$. In order to compare the convergence rates of the FLC and the MLFC, a nonconstant desired speed is adopted in this nonlinear system. Suppose that $v_d(t) = 65$ mph in the initial time interval $[0, 2]$ s, and then it increases linearly from 65 to 75 mph in the next 13 s. After that, it remains constant at the new value until $t = 35$ s. From $t = 35$ s to $t = 40$ s, it decreases linearly to the original value of 65 mph, and keeps that value for later time.

Consider an automobile of which dynamics is expressed as (Passino and Yurkovich, 1998)

$$\dot{v}(t) = \frac{1}{m} [-A_p(t) \cdot v^2(t) - d(t) + f(t)] \quad (9.24)$$

$$\dot{f}(t) = \frac{1}{\tau} [-f(t) + u(t)] \quad (9.25)$$

where

- $u(t)$ is the control action that can be both positive and negative ($u(t) > 0$ represents a throttle input and $u(t) < 0$ is a brake input)
- $m = 1300 \text{ kg}$ is the mass of the vehicle
- $\tau = 0.2 \text{ s}$ is the engine/brake time constant
- $d(t) = 100 \text{ N}$ is the friction force in the normal operation condition
- $A_\rho(t) = 0.3 \text{ N}$ is the aerodynamic drag in the normal operation condition
- $f(t)$ is the driving/braking force

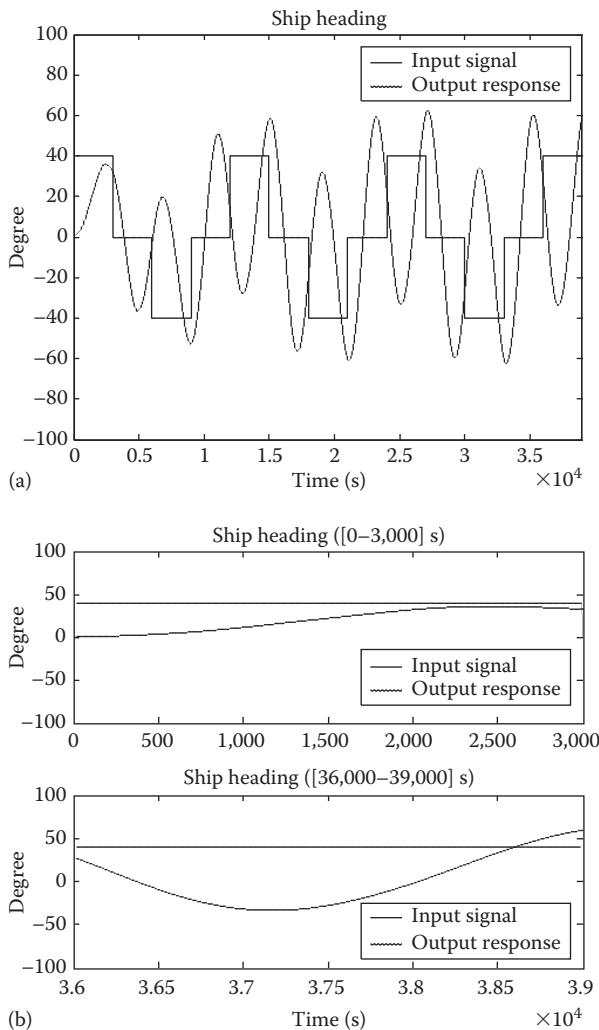


FIGURE 9.9 Simulation result for the FLC with wind disturbance: (a) ship heading during the entire operating time, (b) ship heading in the starting and ending periods,

(continued)

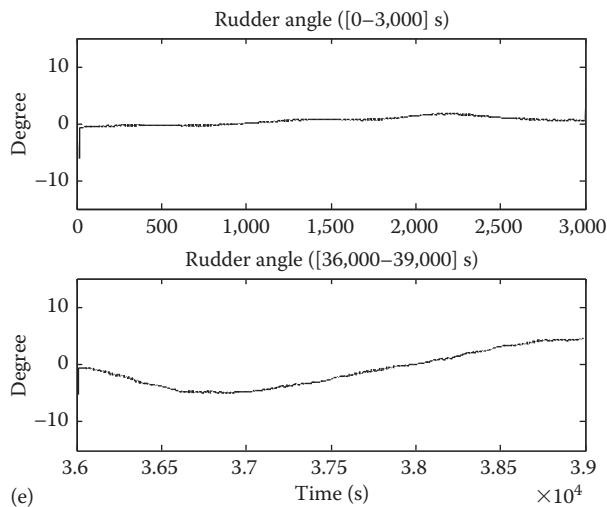
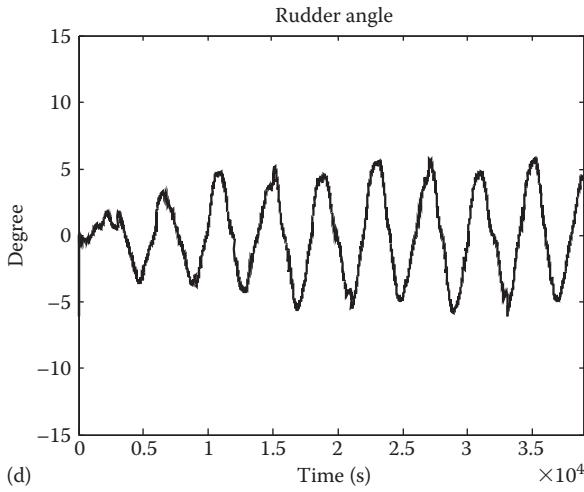
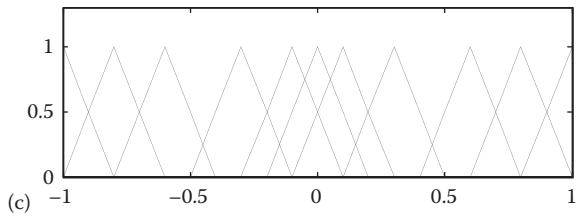


FIGURE 9.9 (continued) (c) fuzzy MFs for output, (d) rudder angle during the entire operating time, and (e) rudder angle in the starting and ending periods.

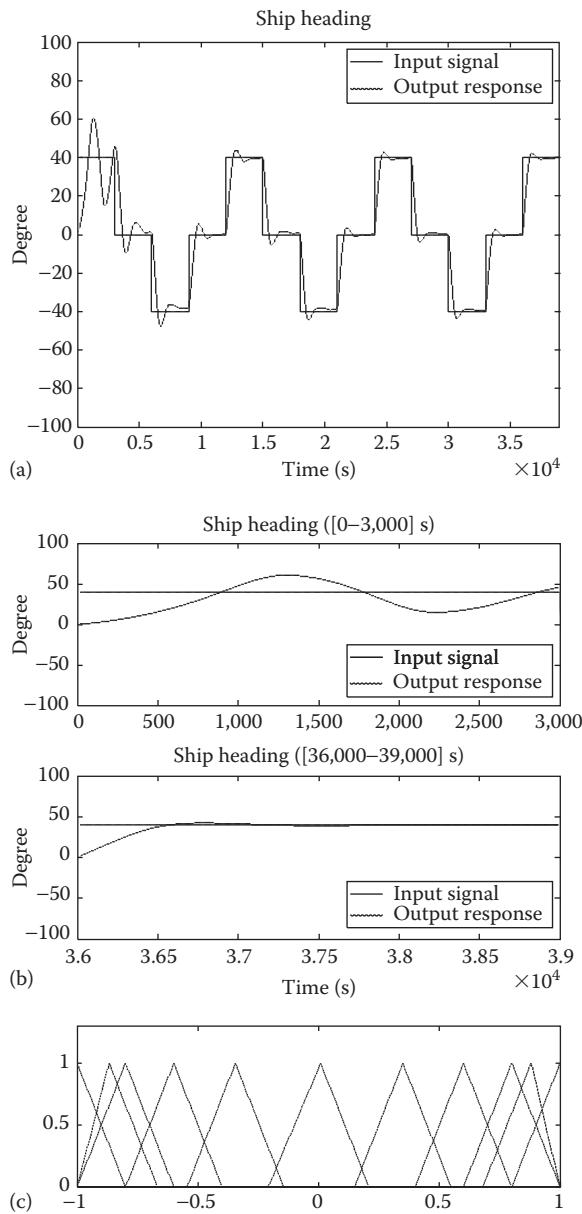


FIGURE 9.10 Simulation result for the MLFC with wind disturbance: (a) ship heading during the entire operating time (b) ship heading in the starting and ending periods, (c) fuzzy MFs for output,

(continued)

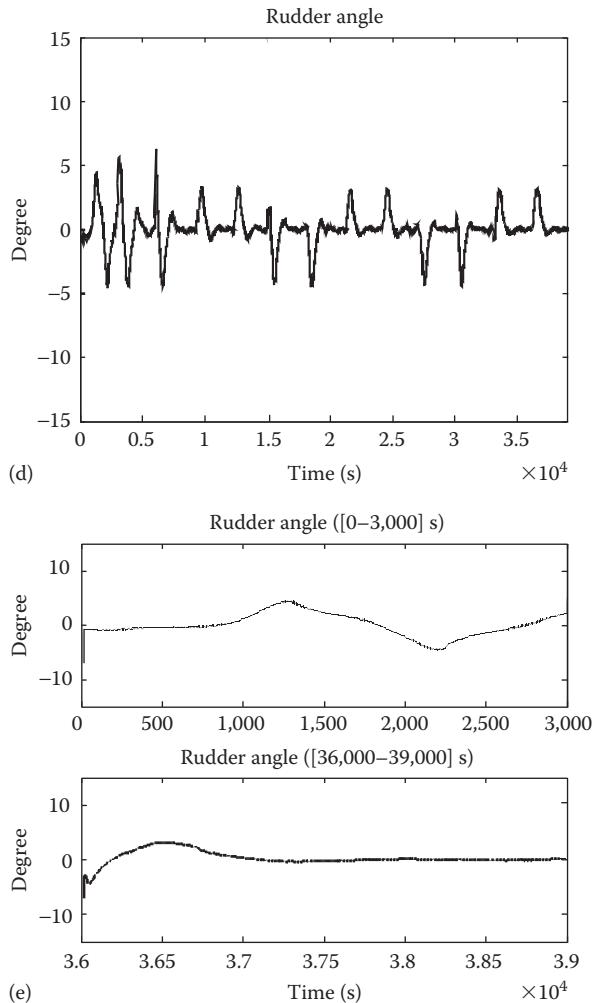


FIGURE 9.10 (continued) (d) rudder angle during the entire operating time, and (e) rudder angle in the starting and ending periods.

Furthermore, assume the input $u(t) \in [-1200 \text{ N}, 1200 \text{ N}]$ (i.e., that $u(t)$ is saturated at $\pm 1200 \text{ N}$).

The inputs to the fuzzy controller are the speed error and the change of the speed error expressed as

$$e(kT) = v_d(kT) - v(kT) \quad (9.26)$$

and

$$c(kT) = \frac{e(kT) - e(kT - T)}{T} \quad (9.27)$$

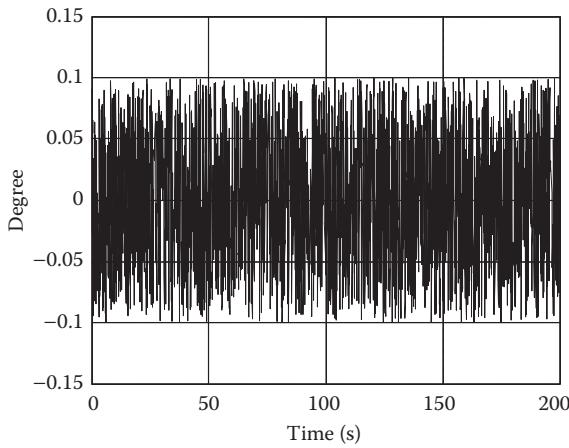


FIGURE 9.11 Sensor noise.

respectively, where

$v_d(kT)$ is the desired vehicle's speed

$v(kT)$ is the actual vehicle's speed

T is the sampling time for the controller

In the controller design, the scaling factors for the error, change in error, and the controller output are chosen as $g_e = 10$, $g_c = 0.5$, and $g_u = -1000$. The values are determined based on the range of each variable as described in detail in [Section 9.3.1](#). Note that the gain of the controller output g_u is chosen as a negative number, due to the fact that an increase in the control action $u(t)$ will intuitively increase the vehicle's speed $v(t)$, which is an opposite relationship as the one used in the simulation of the cargo ship steering.

Case 9.5 Normal Condition

To test the nominal performance of the nonlinear dynamic system for the FLC and the MLFC, simulations are first run for constant parameters with the frictional force $d(t) = 100$ N and the aerodynamic drag $A_p(t) = 0.3$ N ([Figures 9.14](#) and [9.15](#)). Due to the nonconstant desired speed during the simulation interval, the system outputs for both controllers show overshoots at the points of input signal change. The zero steady-state error of the FLC cannot be guaranteed at the end of simulation time. In contrast, since the MLFC has the ability to tune the fuzzy control rules whenever the output error or the change in error exists in the system, it has a relative smaller overshoot and near zero steady-state error, which proves its capability in tracking the nonconstant input signal and improving the system performance.

Case 9.6 With Time-Varying Friction Force

In many industrial processes, the involved parameters are not fixed and keep changing during the operation, which makes the ability of the MLFC in dealing with the

parameter variations more necessary. In this example, system response with a time-varying friction force is investigated to show how the MLFC deals with the parameter variations in the middle of plant operation. The corresponding friction force, which is mainly due to the poor surface condition of the road, is shown in [Figure 9.16](#).

As seen from Figure 9.16, the friction force is kept the same as in the normal condition, 100 N, only for the first 5 s. After that, it increases linearly to 700 N and

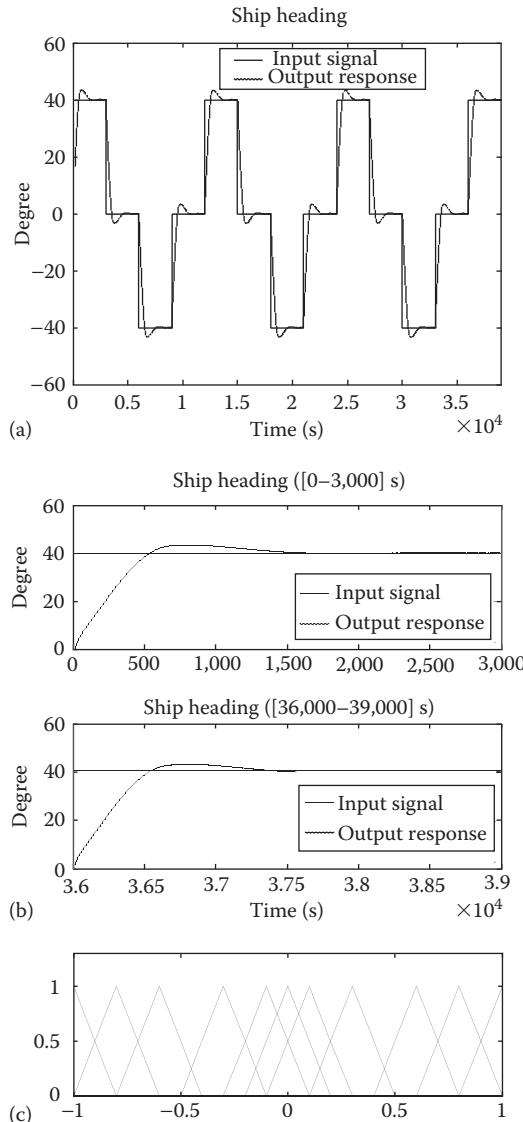


FIGURE 9.12 Simulation result for the FLC with sensor noise: (a) ship heading during the entire operating time, (b) ship heading in the starting and ending periods, (c) fuzzy MFs for output,

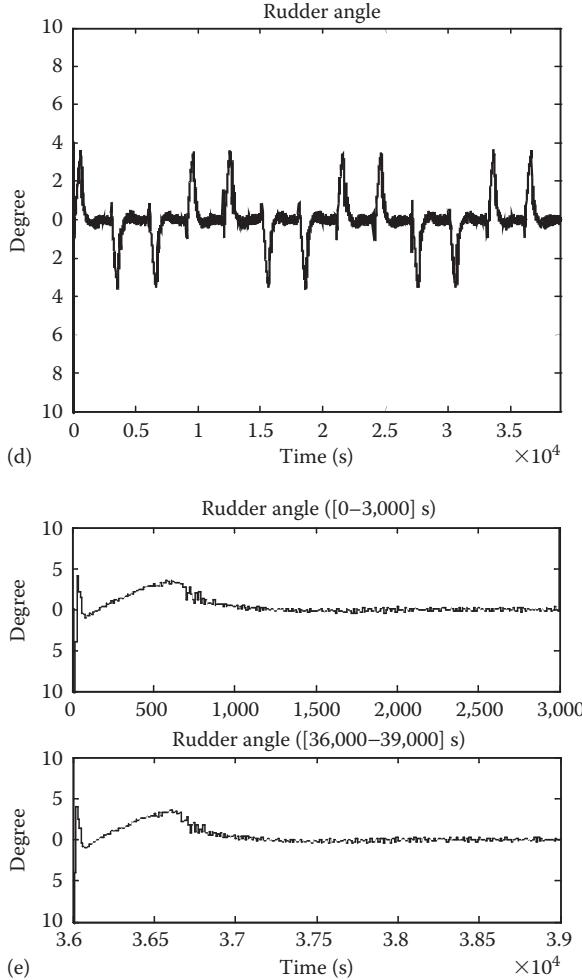


FIGURE 9.12 (continued) (d) rudder angle during the operating time, and (e) rudder angle in the starting and ending periods.

then remains constant at this new value. It will enlarge the error at the points of input signal change. When comparing Figures 9.14 and 9.17, it can be seen that the error signal for the FLC in this case reaches the maximum magnitude of about 1, which is much larger than the one in the normal condition, where the maximum magnitude is only about 0.3.

In order to suppress the undesirable effect of the time-varying parameters, the MLFC will relocate the centers of the output fuzzy MFs, which in turn minimize the error signal efficiently. As can be observed from Figure 9.18, the MLFC can drive this nonlinear plant with a time-varying parameter to follow the desired trajectory with a small overshoot and a very fast convergence rate. In contrast, the FLC produces large steady-state errors over the entire simulation time interval.

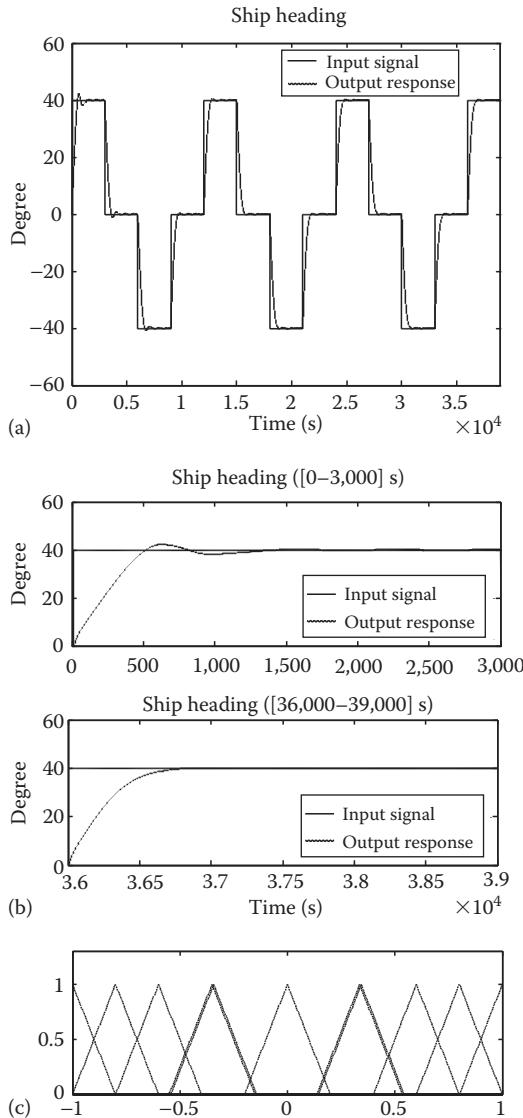


FIGURE 9.13 Simulation result for the MLFC with sensor noise: (a) ship heading during the entire operating time, (b) ship heading in the starting and ending periods, (c) fuzzy MFs for output,

Case 9.7 With Time-Varying Friction Force and Aerodynamic Drag

In this example, two time-varying parameters are applied to the system dynamics, and then a significant improvement in tracking capability achieved by the proposed MLFC strategy is demonstrated. The friction force is expressed as

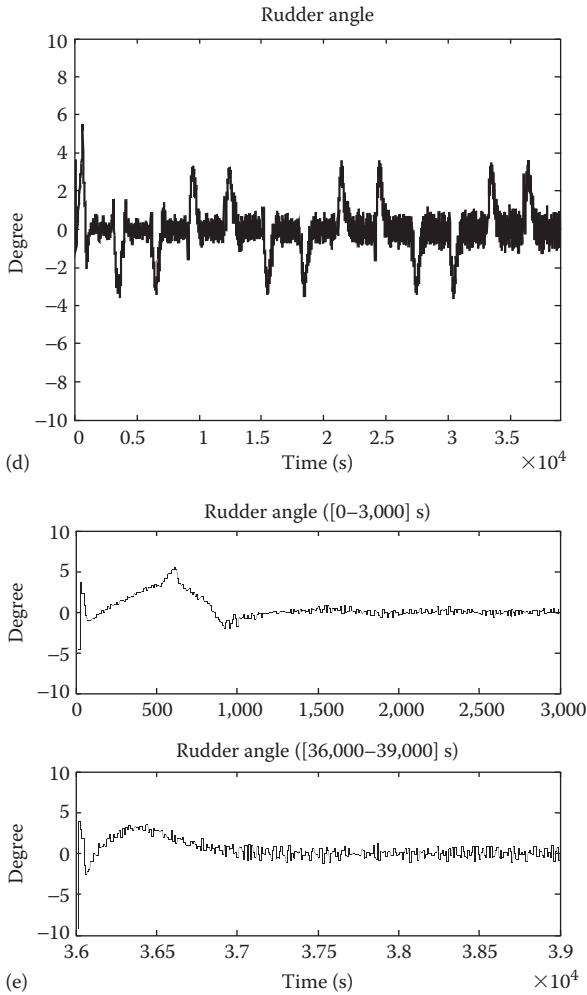


FIGURE 9.13 (continued) (d) rudder angle during the entire operating time, and (e) rudder angle in the starting and ending periods.

$$d(t) = \begin{cases} 100 & t \in [0, 5] \text{ s} \\ 100 + 30 \times (t - 5) & t \in [5, 25] \text{ s} \\ 700 & t \in [25, 50] \text{ s} \end{cases} \quad (9.28)$$

which is the same as in Case 9.6. The aerodynamic drag is

$$A_p(t) = \begin{cases} 0.5 & t \in [0, 25] \text{ s} \\ 0.5 - \frac{0.4}{15} \times (t - 25) \times [1 + \sin(5 \times t)] & t \in [25, 40] \text{ s} \\ 0.1 & t \in [40, 50] \text{ s} \end{cases} \quad (9.29)$$

These time-varying forces are graphically shown in [Figure 9.19](#).

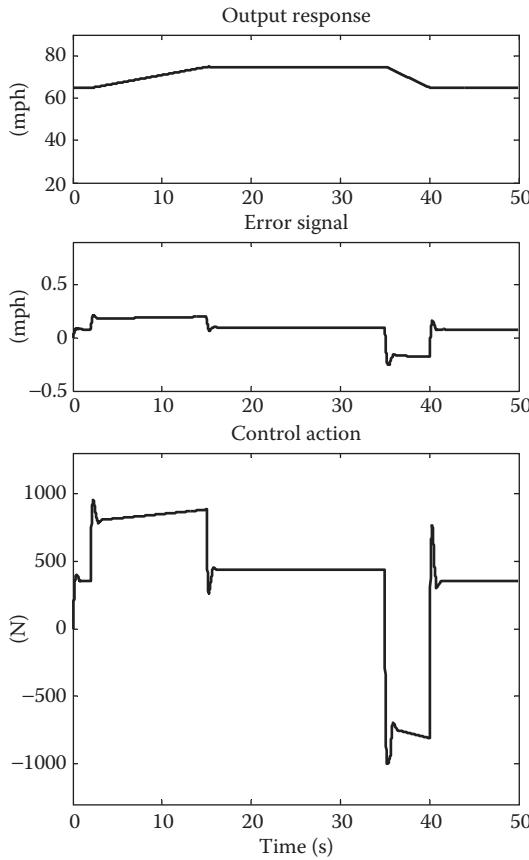


FIGURE 9.14 Simulation result for the FLC under normal condition.

Due to the co-effect of the two time-varying parameters, the system performance is really poor for the FLC algorithm, and the maximum error magnitude reaches around 6, as shown in Figure 9.20. For the MLFC, much smaller transient tracking error is achieved as shown in Figure 9.21 with magnitude only around 1, and almost zero final tracking error, which demonstrates the good learning capability of the proposed MLFC in dealing with severe parameter variations.

As demonstrated through this example, the proposed MLFC strategy can provide an excellent output tracking performance even with changes of input signal and parameters not converging to their nominal values.

9.3.3 WATER LEVEL CONTROL

In this section, a water level control problem for a surge tank is adopted to show the superiority of the MLFC under the condition of parameter variations (Passino and Yurkovich, 1998; Nounou and Passino, 1999, 2000). Consider a surge tank that is spherical-shaped as shown in Figure 9.22.

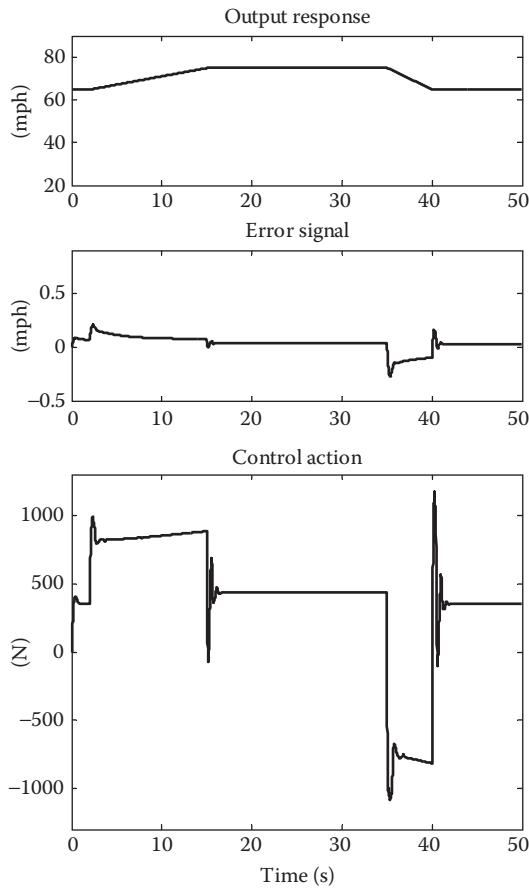


FIGURE 9.15 Simulation result for the MLFC under normal condition.

The differential equation representing this system is

$$\frac{dh(t)}{dt} = \frac{-c\sqrt{2gh(t)}}{A[h(t)]} + \frac{1}{A[h(t)]}u(t) \quad (9.30)$$

where

$u(t)$ is the input flow (m^3/s), which can be either positive or negative (it can both pull liquid out of the tank and put it in)

$h(t)$ is the liquid level (the output of the plant)

$A[h(t)]$ is the cross-sectional area of the tank

$g = 9.8 \text{ m/s}^2$ is gravitational acceleration

$c = 1$ is the known cross-sectional area of the output pipe

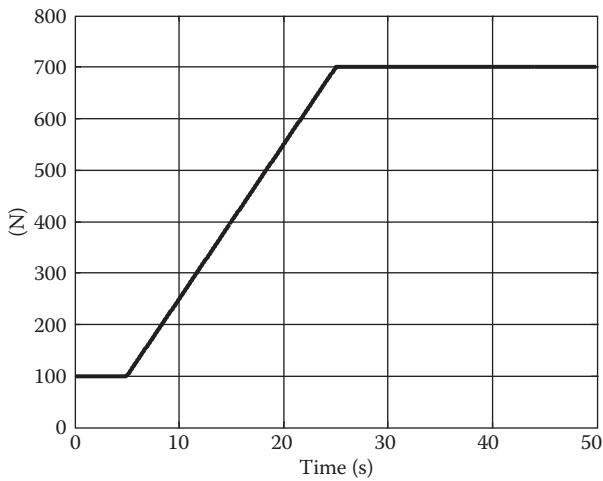


FIGURE 9.16 Time-varying friction force.

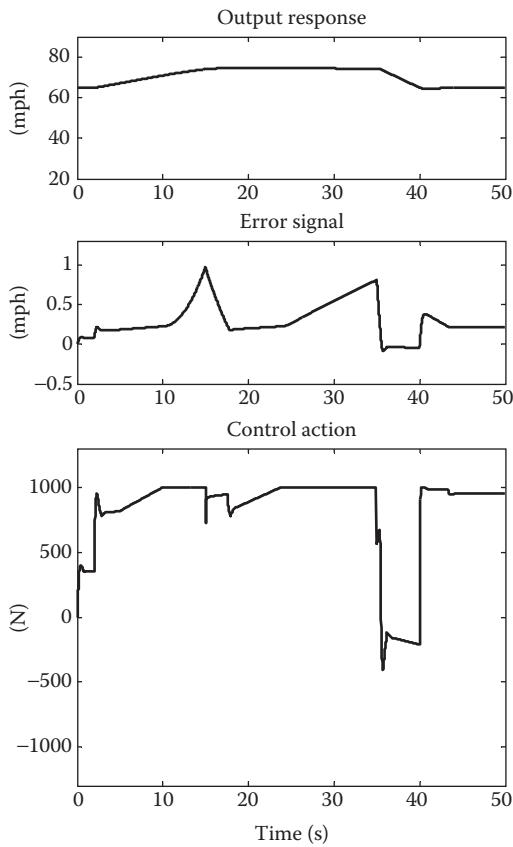


FIGURE 9.17 Simulation result for the FLC with time-varying friction force.

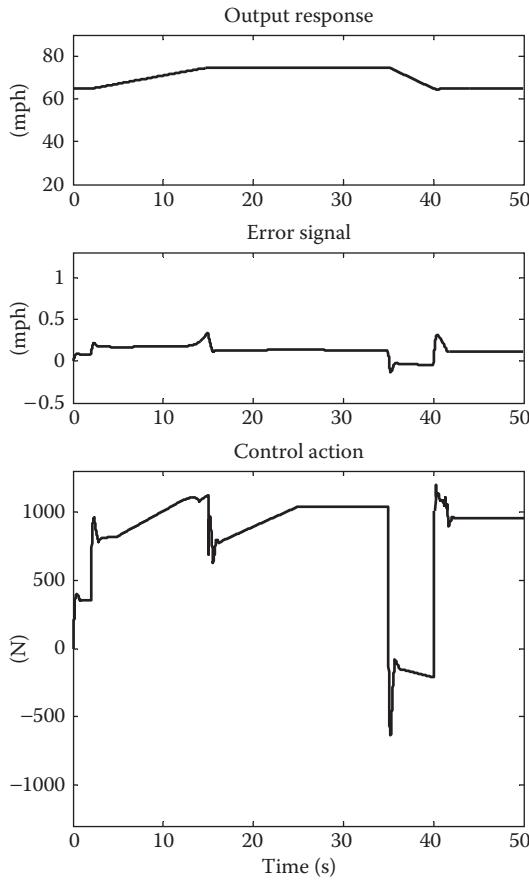


FIGURE 9.18 Simulation result for the MLFC with time-varying friction force.

Let $r(t)$ be the desired level of the liquid in the tank (the reference input). Assume $A[h(t)] = ah^2(t) + b$, where a is unknown but $a \in [a_1, a_2]$, for some fixed positive numbers a_1 and a_2 , and b is another unknown but $b \in [b_1, b_2]$, for some fixed positive numbers b_1 and b_2 . In this simulation example, assume $a_1 = 0.5$, $a_2 = 4$, $b_1 = 1$, and $b_2 = 3$.

The parameters $a = 1$ and $b = 2$ are chosen as the nominal system parameters. The first layer fuzzy control rules of the MLFC are chosen based on the system performance under the normal condition. In addition, as a physical constraint on the input, we assume the plant input saturates at $\pm 100 \text{ m}^3/\text{s}$ such that $-100 \leq u(t) \leq 100$.

The inputs to the fuzzy controller are the liquid level error and the change of the liquid level error is expressed as

$$e(kT) = r(kT) - h(kT) \quad (9.31)$$

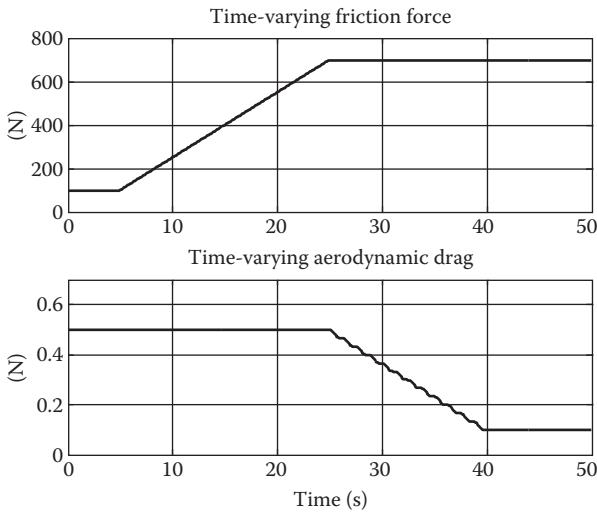


FIGURE 9.19 Time-varying parameters.

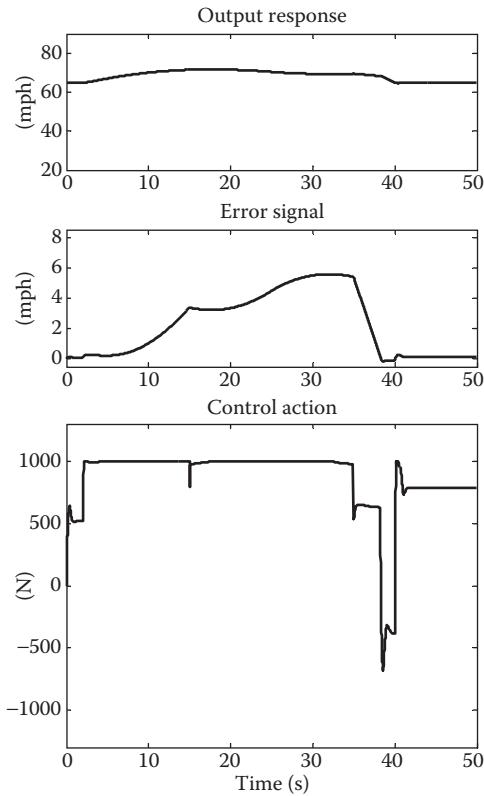


FIGURE 9.20 Simulation result for the FLC with time-varying friction force and aerodynamic drag.

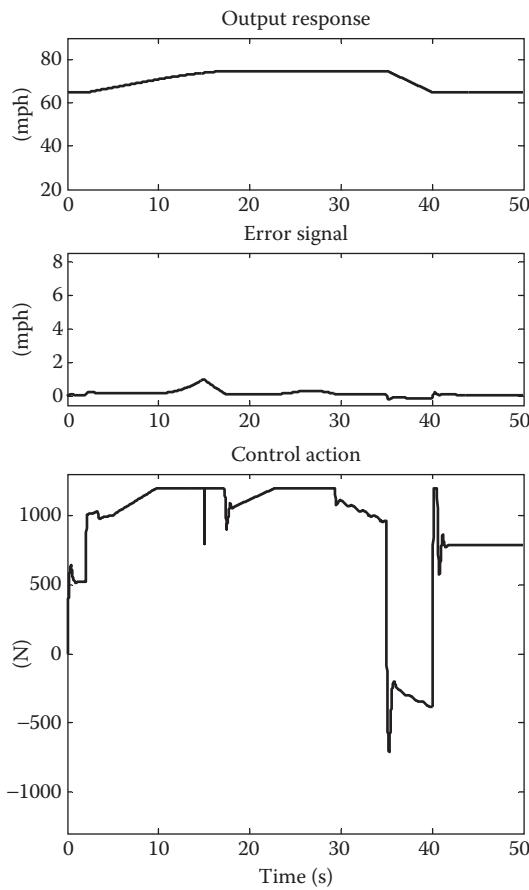


FIGURE 9.21 Simulation result for the MLFC with time-varying friction force and aerodynamic drag.

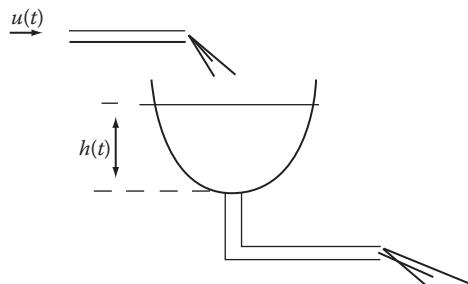


FIGURE 9.22 Surge tank.

and

$$c(kT) = \frac{e(kT) - e(kT - T)}{T} \quad (9.32)$$

respectively, where

$r(kT)$ is the desired liquid level in the tank

$h(kT)$ is the actual liquid level

T is the sampling time for the controller

The controller output is the input flow $u(t)$.

The gain factors for the error, change in error, and the controller output are chosen as $g_e = 1$, $g_c = 0.05$, and $g_u = -50$. The scaling factor of the controller output g_u is chosen to be a negative number, due to the fact that an increase in the control action $u(t)$ will intuitively increase the liquid level $h(t)$.

Case 9.8 Time-Varying Parameters with Step Consequence Input

Suppose that from $t = 30$ s to $t = 35$ s parameter a decreases from 1 to 0.5, and parameter b decreases linearly from 2 to 1. After that, from $t = 35$ s to $t = 45$ s, a increases linearly from 0.5 to 4 and b increases linearly from 1 to 3 and then they remain constant at these new values, as can be graphically shown in Figure 9.23. The fuzzy control rules of the FLC and the first layer fuzzy control rules of the MLFC are chosen based on the normal condition with the values $a = 1$ and $b = 2$ in the time interval $t \in [0, 30]$ s.

In order to compare the convergence rates of the FLC and the MLFC to parameter variations, a square wave input signal with an amplitude of 4 and a period of 20 s is considered for the system. The initial value of the actual liquid

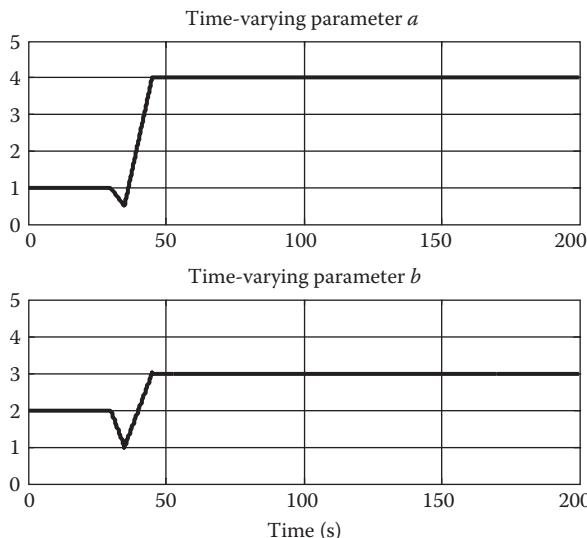


FIGURE 9.23 Time-varying parameters.

level for both systems is chosen to be $h(0) = 1$, which is different from the input value $r(0) = 3$. This choice tends to introduce more discrepancy into the system and will in turn show larger error in the steady state as can be shown in Figures 9.24 and 9.25.

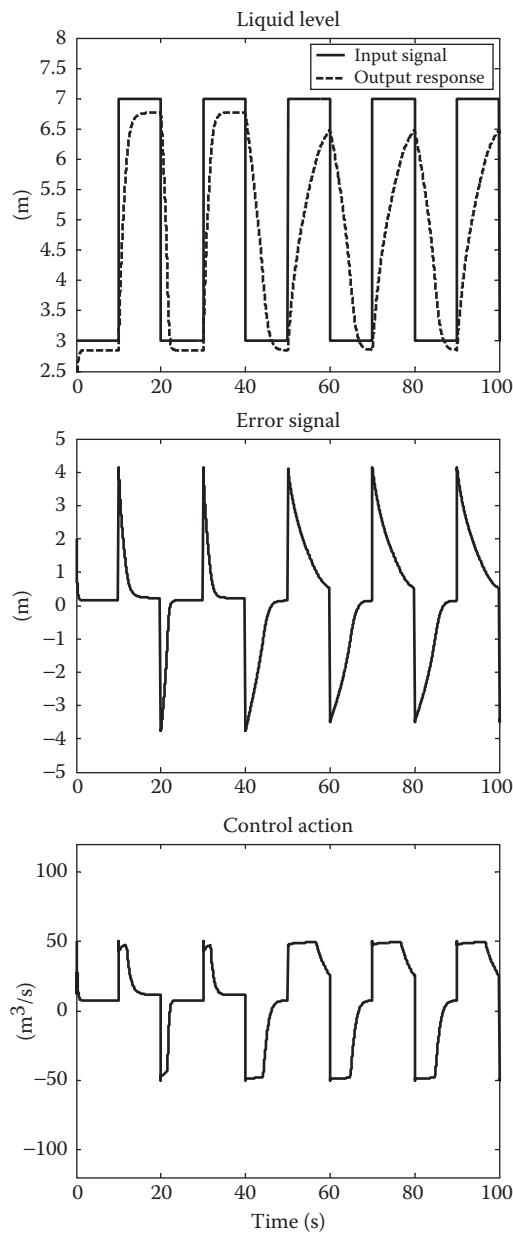


FIGURE 9.24 Simulation result for the FLC with time-varying parameters.

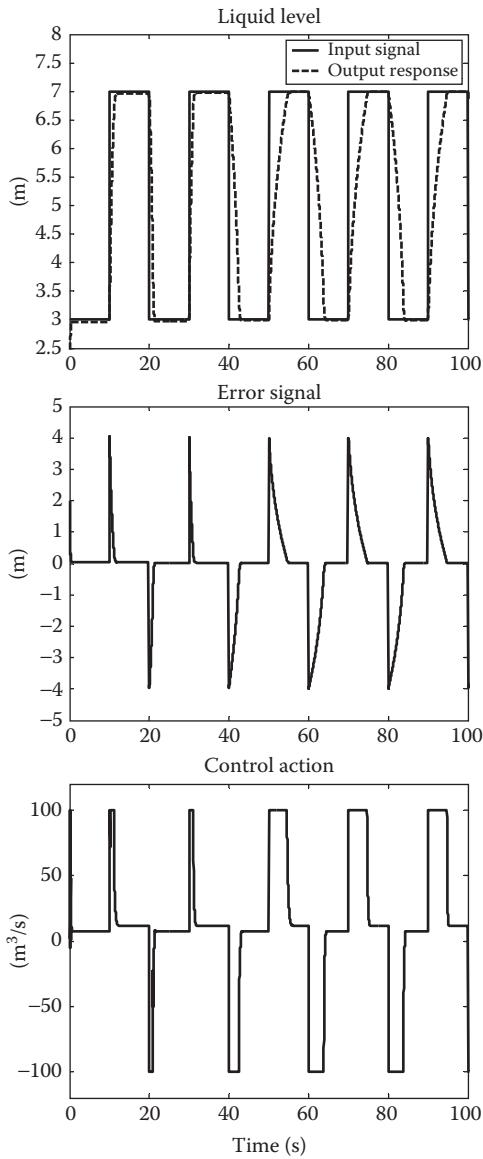


FIGURE 9.25 Simulation result for the MLFC with time-varying parameters.

In the normal operating condition with constant values of $a = 1$ and $b = 2$ at $t \in [0, 30]$ s, the MLFC converges to the reference input very quickly with near zero steady-state error. In contrast, the FLC has no capability to track the input signal and shows steady-state error of about 0.2 in magnitude.

In the time interval $[30, 100]$ s, the parameters a and b change as functions of time, and then keep constants $a = 4$ and $b = 3$, which are different from the

values in the normal condition. Compare Figures 9.24 and 9.25, the output signal of the FLC takes much longer time to settle than that of the MLFC. And also it is found that because of fixed fuzzy control rules, the control action for the FLC saturates at $u \in [-50, 50]$, which is determined by the gain factor $g_u = -50$. Comparatively, the control action of the MLFC saturates at $u \in [-100, 100]$ only due to the physical constraint on the input as addressed before. Thus, the MLFC is quite successful in generating the appropriate control rules for a good system response since the system exhibits a fast transient response with no steady-state error.

Note that the system response for the MLFC takes longer time to settle in the time interval [45, 100] s than the time interval [30, 45] s, because the parameters a and b are no longer kept the same as the values under the nominal condition. The control action $u(t)$ saturates at the boundary of ± 100 as shown in Figure 25c. If the range of the control action $u(t)$ is allowed to be larger, the settling time can be minimized and then the system performance can be improved.

Case 9.9 Time-Varying Parameters with Sinusoidal Input 1

In this example, a sinusoidal desired trajectory of

$$r(t) = 2 + 2 \times \sin(2\pi/5)t \quad (9.33)$$

is used to illustrate the tracking capability of the proposed MLFC.

The simulation results are shown in Figures 9.26 and 9.27. It can be seen that the system responses of the FLC and the MLFC are similar during the initial time interval of $t \in [0, 30]$ s. The error signals are about 0.5 in magnitude in both cases, since the fuzzy control rules are same in the initial time of the nominal operating condition.

As the parameters change during the time interval of [30, 100] s, the system responses show much difference. The error signal for the FLC increases to a magnitude of around 1, and cannot be minimized in the entire 70 s, which shows the incapability of the FLC in dealing with system uncertainty. In contrast, as shown in the last plot of Figure 9.27, the MLFC produces the longest spike of about 0.9 in magnitude at the points of the parameter change. As time progresses, the spikes resulting from subsequent input are reduced to the range of ± 0.5 m as a result of learning and will stay within the range, which is only one-half the value of the FLC case.

Case 9.10 Time-Varying Parameters with Sinusoidal Input 2

In this example, the input is the addition of two sine functions which is expressed as

$$r(t) = 2 + 2 \times \sin(2\pi/5)t + 2 \times \sin(\pi/20)t \quad (9.34)$$

The tracking errors for the FLC and the MLFC are given in Figures 9.28 and 9.29, respectively. The initial value of the liquid level for both algorithms is chosen to be $h(0) = 0.5$, which is different from the input value $r(0) = 3$. The objective is to

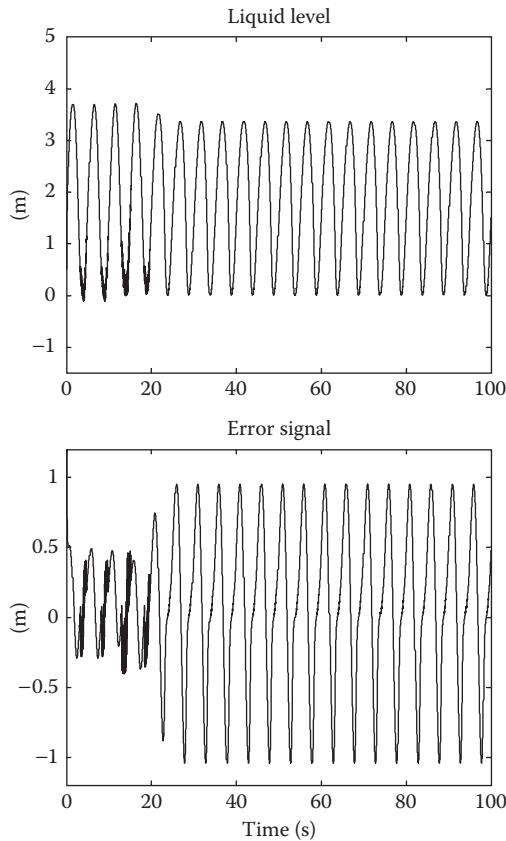


FIGURE 9.26 Simulation result for the FLC with time-varying parameters with sinusoidal input 1.

enlarge the error magnitude existing in the system as addressed before. The magnitude of error stays in the range of ± 2.0 m for the FLC. Comparatively, the output under the MLFC tracks the desired trajectory very well as shown in Figure 9.29. In fact, the tracking error mainly stays within ± 1.0 m, except for the appearance of spikes when the input signal reaches its maximum value.

In this simulation example, several input signals are tested for a nonlinear system with parameter variations. Based on all the observations, it can be concluded that the proposed MLFC is effective in achieving a prescribed transient performance and final tracking accuracy.

9.4 IMPLEMENTATION—FORCE CONTROL FOR GRINDING PROCESSES

In this section, the MLFC technique is implemented for a creep-feed grinding process. The grinding force is maintained at the maximum allowable level under varying depth

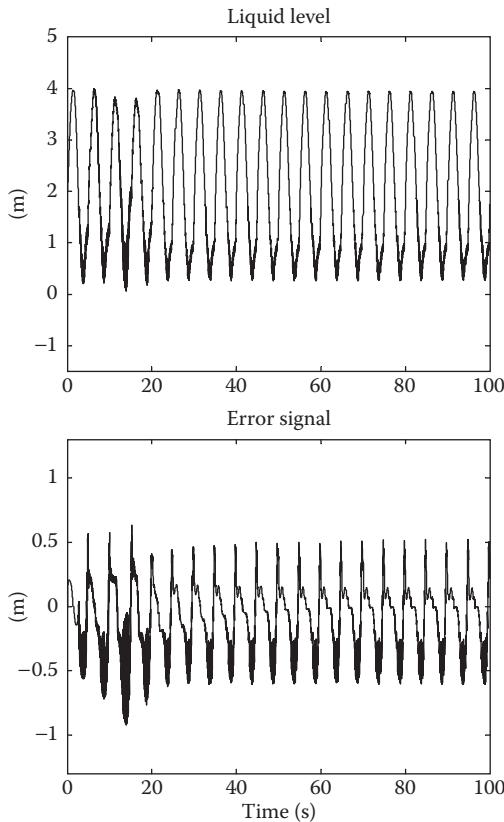


FIGURE 9.27 Simulation result for the MLFC with time-varying parameters with sinusoidal input 1.

of cut, so that the highest MRR is achieved with a good workpiece surface quality. The control rules are generated heuristically without any analytical model of the grinding process. Based on the real-time force measurement, the control parameters are adapted automatically within a stable range. A national instrument (NI) real-time control computer is used to implement in an open architecture control system for the grinding machine. Experimental results show that the cycle time has been reduced by up to 25% over those without force control and by 10%–20% compared with the conventional FLC, which indicates its effectiveness in improving the productivity of actual manufacturing processes. The effect of grinding wheel wear is also considered in the creep-feed grinding process, where the grinding force/power can be maintained around the specified value by the proposed MLFC as the wheel dulls gradually.

9.4.1 HARDWARE CONFIGURATION

Since most of the existing computer numerical control (CNC) controllers and supporting hardware have closed architecture designs, which make it difficult or

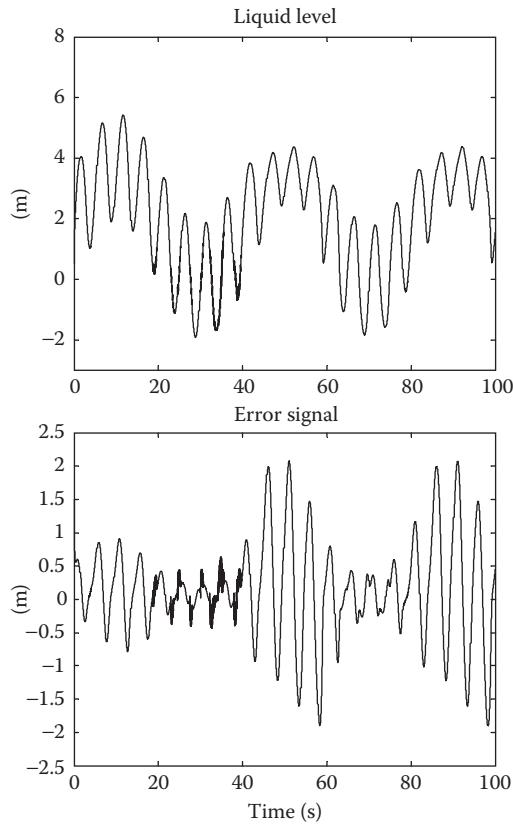


FIGURE 9.28 Simulation result for the FLC with time-varying parameters with sinusoidal input 2.

impossible to incorporate advanced control schemes to extend their capability, an open architecture controller has been designed to eliminate the actual implementation problem. It is capable of replacing or supplementing an existing CNC controller while leaving the original axis and spindle drive motors and supporting electronic interfaces intact. The open architecture controller described in this work was implemented on a three-axis MAZAK CNC machining center with an NI real-time controller, which is a compact, high-performance personal computer (PC) platform for modular instrumentation and data acquisition. A LabVIEW programming environment was used to perform the hierarchical fuzzy control calculation, which directly determines the proper feedrate of the grinding machine. The programs and user interfaces can be easily modified by a programmer to implement new control methods or extend the controller to interface with new CNC systems.

The communication line of the MAZAK control system was reverse-engineered to allow auxiliary controllers to be implemented (Rober and Shin, 1995). The M-32 CNC system interfaces with the axis/spindle controllers for the position signals and

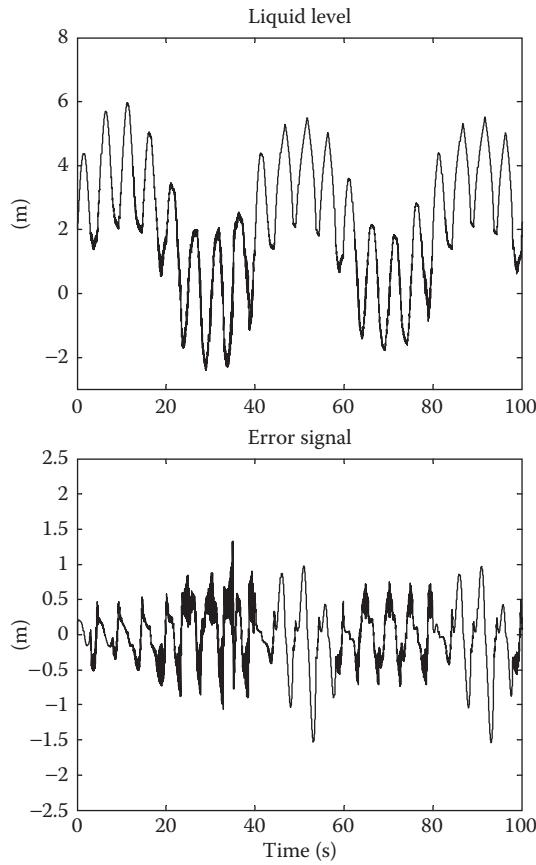


FIGURE 9.29 Simulation result for the MLFC with time-varying parameters with sinusoidal input 2.

the corresponding address via a 50-pin cable, while other peripheral functions such as lubrication, position limit detection by proximity sensors, and emergency stops are controlled by a programmable logic controller (PLC) system with the CNC controller. The existing CNC controller continues to monitor peripheral functions, but the PC-based system generates the required control commands for the axes and spindle. The existing PLC functions are kept intact, such as all emergency stop push buttons and emergency shutdown proximity sensors, thus maintaining the safety requirements of the system. The NI control computer overrides the reference positions for the X , Y , and Z axes and the spindle velocity. A diagram for the hardware interface for the digital communication is shown in Figure 9.30, where the data interface board is the most significant part in this control system. It serves as the digital electronic interface between the CNC system, the PC controller, the X , Y , and Z axes, and the spindle controller. It allows the PC to override the CNC signals to control all three axes of the machine plus the spindle simultaneously. With this

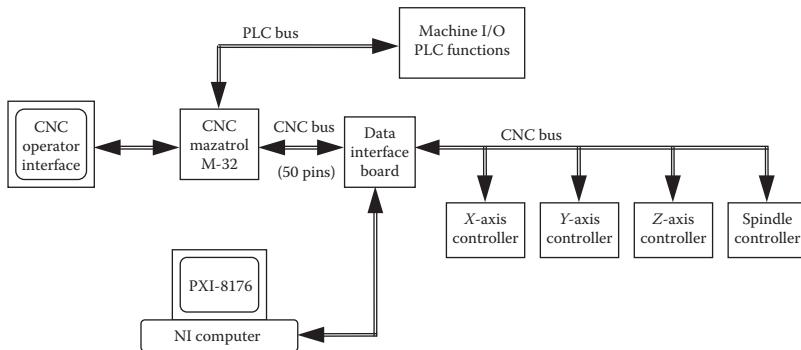


FIGURE 9.30 PC-based controller interface using digital communication.

arrangement, the open architecture controller can be easily adapted to the existing CNC machine.

The extension of the CNC bus interfaces the CNC controller with the axis/spindle controllers, via the PC-based control system. The bus contains the address lines that direct data to a specific axis/spindle controller, and the data lines that transfer the reference position or velocity commands. The feedback signals from the actual position/velocity sensors run through these same data lines to the CNC controller.

In this experiment, the CNC controller runs the address portion of the bus, while the NI control computer overrides the reference position and velocity signals from the CNC to the axis/spindle controllers. The feedback data for the position and velocity is sent from the axis/spindle controllers to the CNC uninterrupted, but the data was read by the NI computer through a specially designed interface board, which allows the NI computer to override the CNC signals to control all three axes of the machine and the spindle simultaneously.

An NI digital I/O board was used to read in the current machine position. Based on the online measured grinding force value, the new position is calculated based on the MLFC calculation and sent to the CNC machine. A constant sampling time of 3.55 ms is maintained to be consistent with that of the CNC of the machine. [Figure 9.31a](#) shows an overall structure of the experimental setup and [Figure 9.31b](#) is the picture for the grinding experimental setup.

9.4.2 MONITORING AND WORKPIECE SETUP

In the control experiment, the controlled variable is the grinding force. The force was measured by a three-axis Kistler type 9257B dynamometer with a Kistler 5004 dual mode amplifier, and was transferred to the NI control computer at the sampling interval of 3.55 ms. The forces during the grinding process are oscillatory in nature due to the random machining action of abrasive grits in the wheel, and therefore a second-order Butterworth low-pass filter (cutoff frequency: 0.01 Hz) was employed to obtain the static component of the grinding force.

Cutting force is directly related to feedrate in grinding processes. In most machining processes with existing CNC control systems, conservative feedrate is

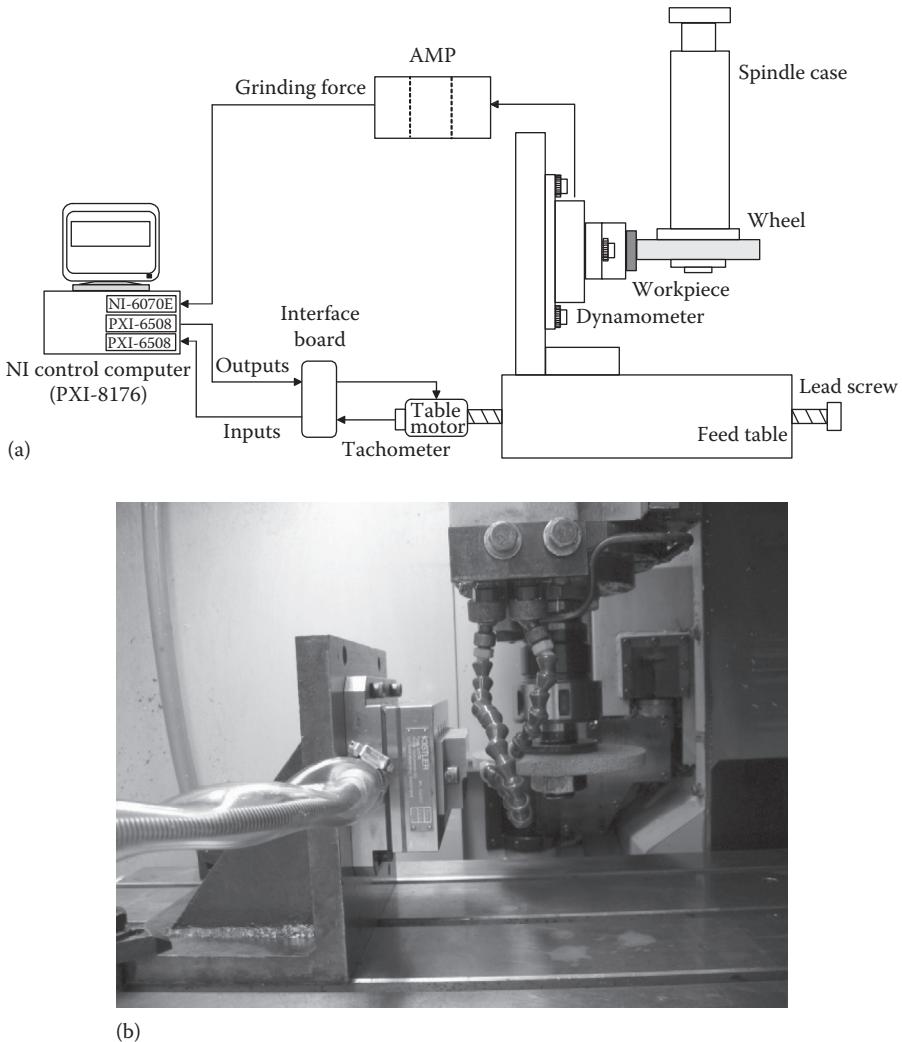


FIGURE 9.31 (a) Overall structure of the experimental setup and (b) picture for the creep-feed grinding experimental setup.

often used to keep the cutting forces within an acceptable range in order to avoid excessive cutting forces arising from the variation of workpiece geometry and wheel wear, which could damage the workpiece or spindle, cause high tool wear, or poor surface quality of the finished part. However, these conservative levels usually result in low productivity for machine tools. By introducing the MLFC into the control system, the operating condition can be varied according to working environment changes. The MLFC manipulates the machine tool's operating condition, providing further control ability in addition to those exclusively provided by the existing CNC-PLC controller. Usually, the CNC guides the normal sequence of the

tool/table position/velocity during machining, while in this case, the MLFC determines the proper feedrate to maintain a constant cutting force during the grinding operation. The desired force level is entered to the NI control computer via a graphical interface. When the force is smaller than the desired maximum possible level due to the uneven shape of the workpiece or during entry and exit periods, the feedrate is increased to enhance the cutting force and decrease the total cycle time, which in turn maximizes the MRR. On the other hand, if the grinding force is larger than the maximum possible value, the feedrate is decreased instantly to reduce the possibility of large spindle deflection, wheel breakage, or workpiece surface burn. Furthermore, the controller's parameters are adjusted only within the stable range, based on the stability analysis in [Section 9.2](#).

For safety considerations, the maximum feedrate is limited to 3 mm/s in the experiments. If the force levels exceed 100 N, the controller shuts down the system automatically to reduce the possibility of wheel damage. In this experimental study, the aluminum oxide grinding wheel WA60-I110.VC2 with a diameter of 163.32 mm was mounted on the grinding machine with a vertical spindle. The rotational speed of the wheel was fixed at 2500 rpm. The workpiece was mounted on a dynamometer, which was clamped to an angle plate mounted on the machine table. Down creep-feed grinding was chosen in this study.

9.4.3 EXPERIMENTAL IMPLEMENTATION RESULTS

The goal of the implementation of the proposed MLFC is to maintain a constant force level by controlling the feedrate of the creep-feed grinding under varying depth of cut. The control inputs are the error of the grinding force and the change of error. Three force control experiments were conducted under different shapes of the raw workpiece material. In all cases, the grinding force in the steady state was maintained at its maximum possible level by the proposed control scheme, achieving an improved MRR with a minimum cycle time.

In grinding processes, one main limitation to the MRR is workpiece burn (Amitay et al., 1981). The burning threshold corresponds to reaching a critical grinding temperature. At this critical temperature, the specific grinding energy can be determined empirically, as well as the corresponding grinding force (Malkin, 1978). In order to avoid the high-mechanical wheel wear and burning marks to achieve the desired quality and surface roughness of the workpiece, a certain boundary of the grinding force needs to be determined beforehand and the actual force should not exceed this limitation in the grinding experiment. In this study, two AISI 4140 steel blocks were prepared by heat treatment to Rockwell hardness (HR45N) of 56 and 64 RC, respectively.

In general, the value of grinding force depends on the hardness of workpiece material. With softer workpiece material, the normal force value is smaller under the same grinding condition (Younis et al., 1987; Saini, 1990), and the grain-workpiece deflections is smaller as well (Saini and Wager, 1985), which results in a larger actual depth of cut (Chiu and Malkin, 1993). The attritious wear and blunting rate of the wheel rise as the MRR increases (Brenner and Torrance, 1993), which leads to

TABLE 9.3
Correlation of the Grinding Conditions to Workpiece Burn (Hardness: 56 RC)

Nominal Feedrate (mm/s)	Depth of Cut (mm)	Width of Cut (mm)	Burn/No Burn	Force (N)
1.5	3	2	Burn	55
1	3	1	Burn	40
1	2.3	1.5	Burn	40
1	2.3	1	No burn	30

wheel dullness and increased grinding force value (Chiu and Malkin, 1993). When the wheel wear flat area gradually rises with the attritious wear and reaches a critical level, the workpiece will burn (Brenner and Torrance, 1993). The grinding wheel working with a harder workpiece has a better resharpening mechanism (Brenner and Torrance, 1993). On the other hand, the softer workpiece penetrates deeper into grinding wheel than the harder workpiece (Saini, 1990), which makes the abrasive grains of the wheel easily fill and clog the pores of the wheel. The rubbing force between the wheel wear flats and workpiece generates intense frictional heat in the cutting zone (Malkin and Cook, 1971), and virtually all this heat is conducted to the workpiece (Malkin and Anderson, 1974; Malkin, 1978). The clogged grinding wheel affects the cooling performance, so makes it more difficult to remove the heat promptly, which is another reason to cause the workpiece burn.

Based on the explanation above, two sets of experiments were run first without any control scheme in order to determine the upper bound of the filtered grinding force level. The effects of the grinding conditions on workpiece burn are listed in Tables 9.3 and 9.4 for the two workpiece specimens with different hardness as 56 and 64 RC. The resultant upper bounds of the filtered grinding force level were set to be 30 and 60 N.

TABLE 9.4
Correlation of the Grinding Conditions to Workpiece Burn (Hardness: 64 RC)

Nominal Feedrate (mm/s)	Depth of Cut (mm)	Width of Cut (mm)	Burn/No Burn	Force (N)
2	3	2	Burn	85
2	3	1	Burn	75
1	3.3	2	Burn	70
1	3	2	No burn	60

EXPERIMENT 9.1

In this case, the workpiece (hardness: 56 RC) has a flat surface and the depth of cut is set to 2.3 mm with width of cut at 1 mm as shown in Figure 9.32. Figure 9.33 shows the raw and filtered force signals without any force controller, where the table speed is fixed at 1 mm/s. Since the contact length of the grinding wheel to the workpiece increased continuously during the entry period as shown in Figure 9.32, a transient period of around 22 s can be observed where the grinding force increases from zero to a certain value as illustrated in the second graph in Figure 9.33. Also, because of the uneven distribution of the workpiece material along the surface and the possible wear of the wheel, the grinding force is not constant in the steady state even if the depth of cut is kept at the constant value of 2.3 mm.

A single-level FLC with fixed controller parameters (fuzzy rules) was also used for comparison with the MLFC scheme. Figure 9.34 shows a force response with the regular FLC, where only one layer of the fuzzy controller is applied and the fuzzy rules are fixed in the entire operation period. In this case, the transient time of the

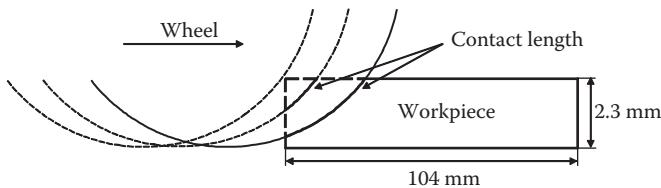


FIGURE 9.32 Illustration of the creep-feed grinding process in Experiment 9.1.

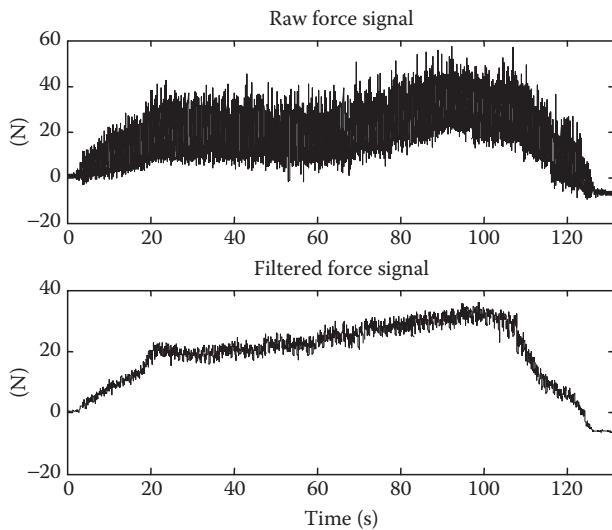


FIGURE 9.33 Force signals without force control in Experiment 9.1.

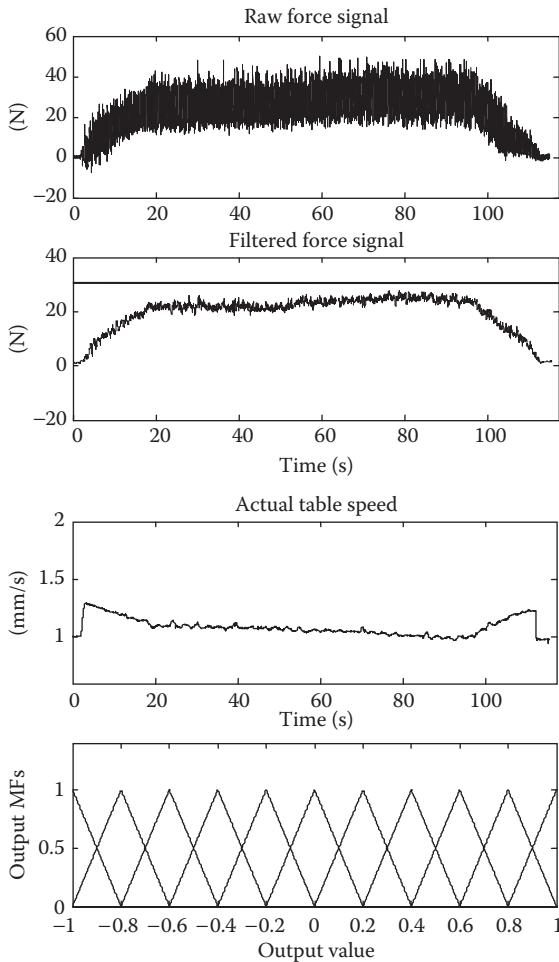


FIGURE 9.34 Force, control signal, and output MFs with the regular FLC in Experiment 9.1.

entry period is reduced to around 20 s. From the second graph in Figure 9.34, it can be seen that the regular FLC with fixed fuzzy rules also lacked the ability to maintain the constant grinding force in the steady-state region of the time interval [20–90] s. Experimental results for the MLFC system are presented in Figure 9.35. Compared to the grinding force profile with the regular FLC, the MLFC shows the capabilities to tune control parameters quickly, reduce the transient time to 15 s in the entry region, and maintain a constant level in the steady-state region of the entire grinding cycle. The total cycle time is also reduced from 130 s without any force control to 110 s, and the MRR is increased correspondingly. For the easy comparison of the control performance of the regular FLC with the MLFC, the reference force signal at 30 N is drawn in the plot of the filtered force signal in Figures 9.34 and 9.35.

The last plots of the Figures 9.34 and 9.35 illustrate the output MFs at the end of the experiments. In order to make the figures clear, only the output MFs

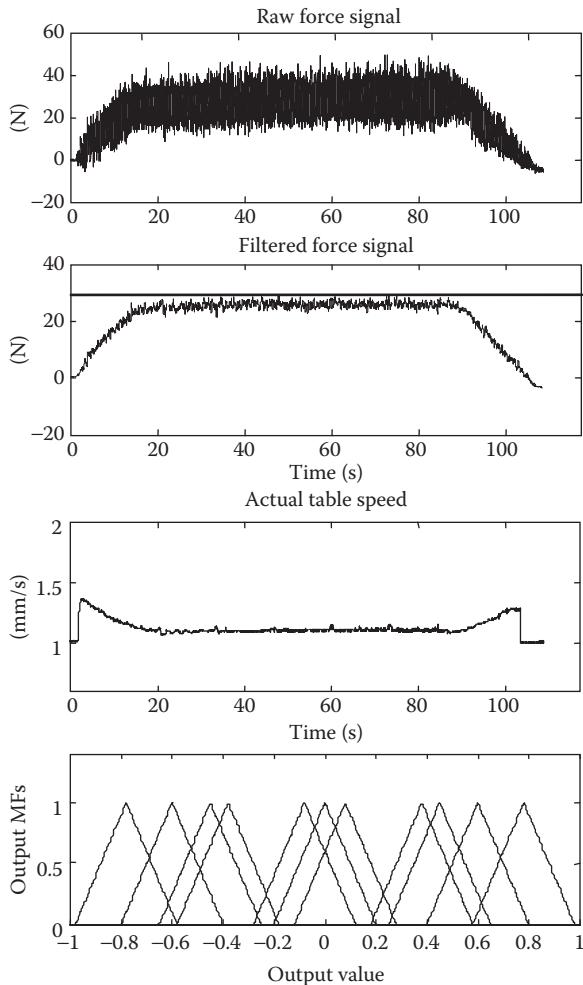


FIGURE 9.35 Force, control signal, and output MFs with the MLFC in Experiment 9.1.

corresponding to zero value of change in error are plotted. Each of the 11 triangles is the output MF in the normalized universe of control action. In the case of the regular FLC, the output MFs are evenly distributed and are fixed in the entire operation. On the other hand, the MLFC can tune the position of the output MFs with the embedded adaptation mechanism, and a better system performance is achieved as a result. ■

EXPERIMENT 9.2

Next a workpiece (hardness: 56 RC) with a sloped surface is considered as shown in Figure 9.36, and the width of cut is set to 2 mm. The grinding force without any force controller (with a constant feedrate at 1 mm/s) is shown in Figure 9.37, which has a

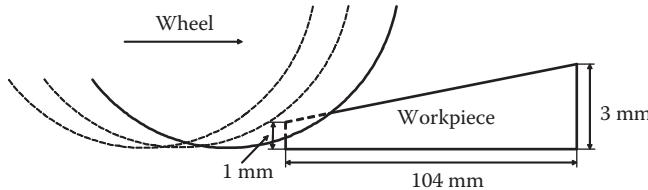


FIGURE 9.36 Illustration of the creep-feed grinding process in Experiment 9.2.

large variation from the entry point to the exiting point, since the depth of cut is increased continuously from 1 to 3 mm. [Figure 9.38](#) shows the control performance with the regular FLC, while [Figure 9.39](#) is the response with the MLFC. Again, the reference force signal at 30 N is drawn in the plot of the filtered force signal in Figures 9.38 and 9.39 for easy comparison.

In this experiment, the depth of cut changes continuously. In the beginning, when the depth of cut is small, a relatively large feedrate is calculated from the control algorithm. As the depth of cut increases, the feedrate is reduced to maintain the actual force value around the reference force level. A decreasing trend in the actual feedrate can be noticed in both the regular FLC and the MLFC cases due to increasing depth of cut. With the MLFC, the total grinding cycle is reduced by 25% over the case without any force controller and by 15% over the regular FLC case. Again, output MFs after the experiments are shown for comparison. ■

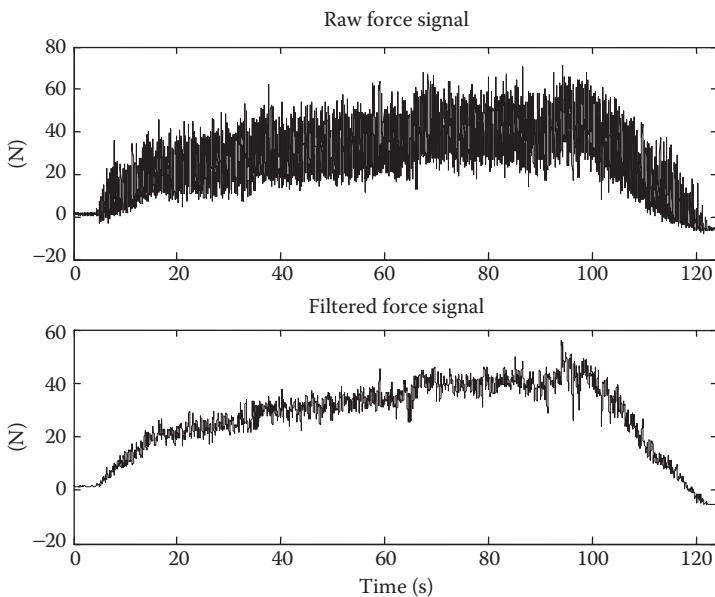


FIGURE 9.37 Force signals without force control in Experiment 9.2.

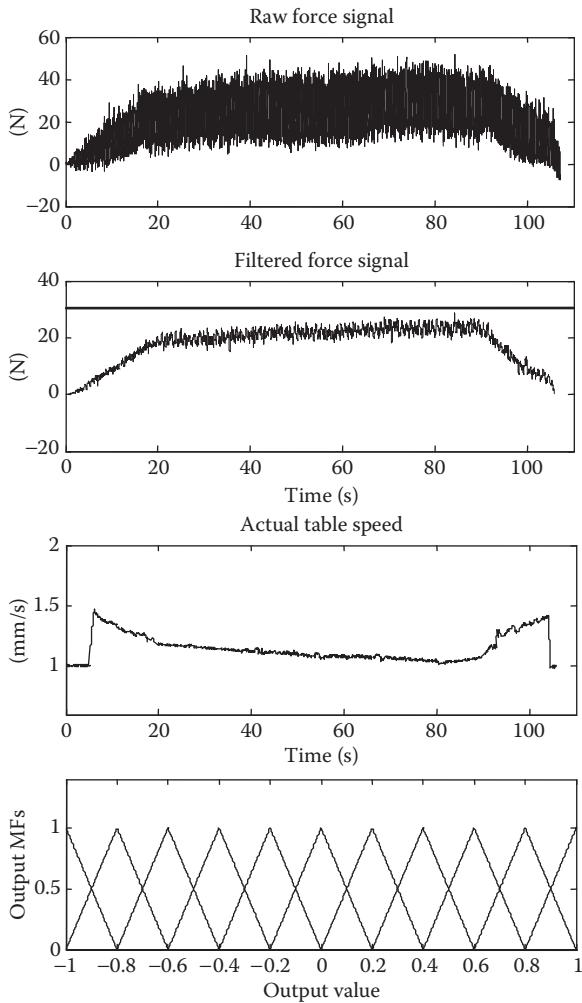


FIGURE 9.38 Force, control signal, and output MFs with the regular FLC in Experiment 9.2.

EXPERIMENT 9.3

In this experiment, the system performance of the MLFC is compared with the case when no controller is applied. A workpiece (hardness: 64 RC) with two-step changes in the depth of cut was used. The first change is from 1 to 3 mm, and the second change reduces the depth to 2 mm as shown in [Figure 9.40](#). This allows a thorough performance evaluation of the MLFC for both a sudden increase and decrease in the depth of cut. The width of cut used in this case was 2 mm.

In the case when no force control was implemented, the table speed was fixed at 1 mm/s as shown in the last plot in [Figure 9.41](#). As the depth of cut was increased, the

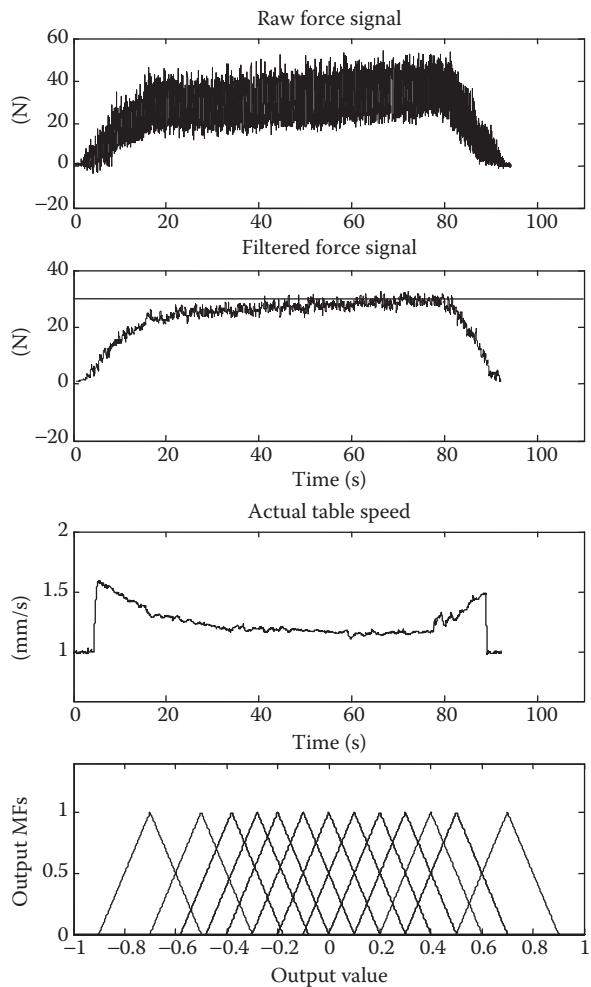


FIGURE 9.39 Force, control signal, and output MFs with the MLFC in Experiment 9.2.

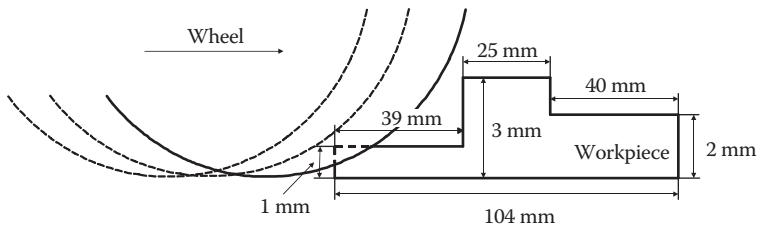


FIGURE 9.40 Illustration of the creep-feed grinding process in Experiment 9.3.

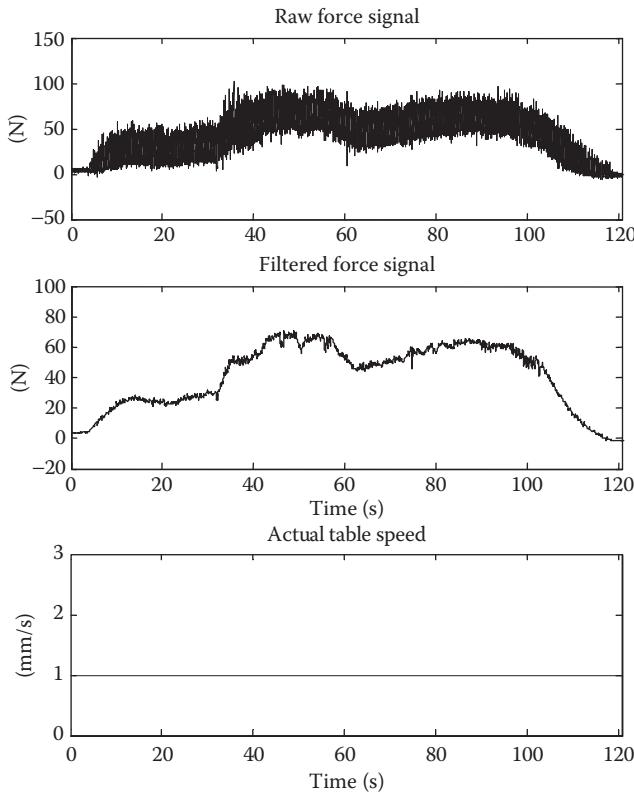


FIGURE 9.41 Force signals and control signal without the force controller in Experiment 9.3.

cutting force level increased substantially as expected. At the beginning of the grinding cycle, since the depth of cut in the first step is only 1 mm, the grinding force is pretty small, which is just around 20 N as shown in the second graph in Figure 9.41. The time period to finish this step is $37 - 5 = 32$ s. In contrast, The MLFC was able to adjust the force from 20 to 70 N in a very short amount of time ($20 - 5 = 15$ s), as illustrated in the second plot in Figure 9.42. In the middle part of the workpiece where the grinding force is roughly about its maximum value, the MLFC is able to reduce the feedrate correspondingly as shown in the last graph in Figure 9.42. In this case, the cycle time is reduced by about 25% with the MLFC while maintaining the force at the target level. ■

The three experimental results are summarized in Table 9.5. Experiment 9.1 is an extreme case since there is no variation in the depth of cut in the whole pass. The workpiece has a flat surface. Even in this case, the cycle time can be reduced from 130 to 110 s, by shortening the transient time in the entry period. While in real machining processes, the raw workpiece surface will not always be flat. It will have some slopes or some steps as in Experiments 9.2 and 9.3. When the depth of

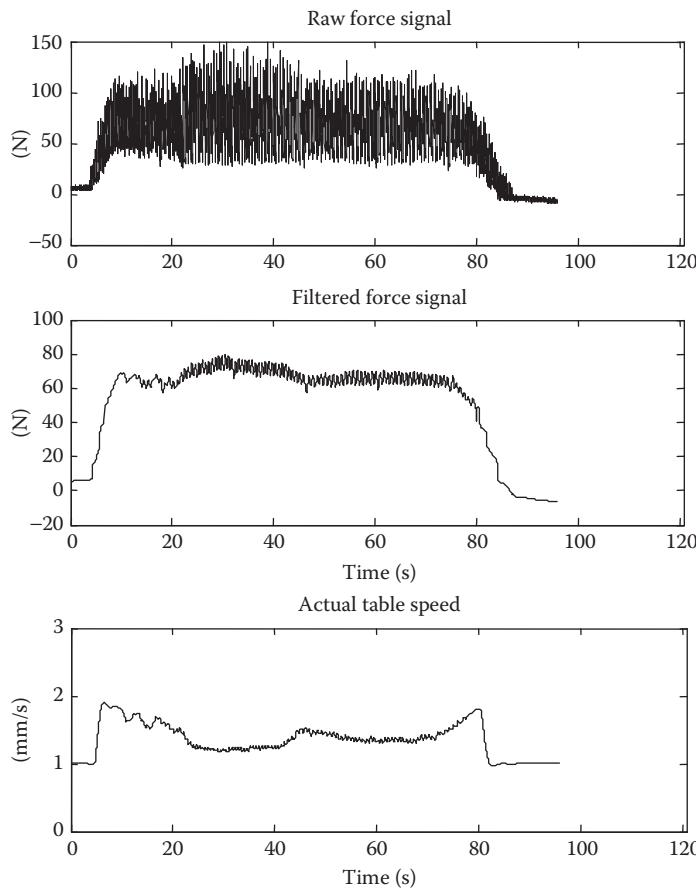


FIGURE 9.42 Force signals and control signal with the MLFC in Experiment 9.3.

TABLE 9.5
Summary on Experimental Results

	Experiment 9.1 (Flat Surface)		Experiment 9.2 (Sloped Surface)		Experiment 9.3 (Stepped Surface)	
	Cycle Time (s)	Cycle Time Reduction (%)	Cycle Time (s)	Cycle Time Reduction (%)	Cycle Time (s)	Cycle Time Reduction (%)
Without force controller	130	0	125	0	120	0
With regular FLC	117	6	110	15	N/A	N/A
With MLFC	110	15	94	25	90	25

cut is smaller, the feedrate can be increased to maintain the grinding force at the reference level, and the cycle time will be reduced. From the experimental results, the percentage of the cycle time reduction with an uneven workpiece surface will be more than that with a flat surface. As can be seen in [Table 9.5](#), the cycle time could be reduced by 25% with the MLFC over the case without any force controller for sloped and stepped surface, and by 15% for a flat surface.

9.4.4 WHEEL WEAR EXPERIMENTS

In this section, two sets of experiments are described to illustrate that the proposed control scheme is able to compensate for grinding wheel wear. A 300 mm long workpiece (hardness: 55 RC) was prepared so that the experiments could be performed for one whole pass in a creep-feed grinding condition. Normally, the wheel wear has a slowly time-varying effect. Thus, even in a fixed working condition, the grinding force/power will increase due to the deterioration of the working surface of the grinding wheel such as wear and loading. In the following experiments, the feedrate with the proposed MLFC will be shown to decrease at a small rate so that the grinding force/power is maintained around its nominal values. Since the length of dynamometer in [Figure 9.31a](#) is limited as 170 mm, a powermeter was used to control the feedrate in these experiments, where the effective grinding power was calibrated beforehand to have a proportional relationship to the tangential force.

EXPERIMENT 9.4

In this case, the workpiece depth of cut was set to 1 mm and width of cut at 1 mm as shown in Figure 9.43. The nominal feedrate was specified as 1 mm/s. [Figure 9.44](#) shows the raw and filtered power signals without any control scheme. As illustrated in the first two plots, due to the wheel wear effect, the effective grinding power increases gradually within one pass at the constant feedrate. [Figure 9.45](#) shows the grinding power response with the proposed fuzzy control scheme, where the grinding power is maintained around the constant level in the entire pass and the feedrate is reduced at a certain rate as shown in the last plot in Figure 9.45. Note there is a lower limit for the workpiece feedrate (e.g., 0.1 mm/s) in view of the chip formation mechanism of grains as stated in Furukawa and Ohishi (1984) and Kirk (1976). Once the calculated feedrate is smaller than the lower boundary, the grinding process needs to be stopped and the wheel needs to be redressed to restore its sharpness. ■

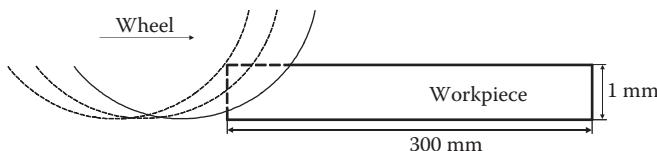


FIGURE 9.43 Illustration of the wheel wear experiment in Experiment 9.4.

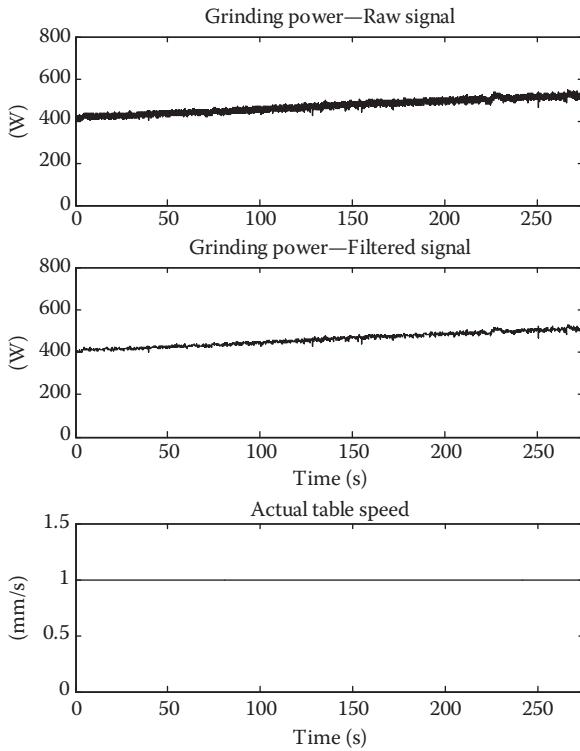


FIGURE 9.44 Power signals and control signal without any controller in Experiment 9.4.

EXPERIMENT 9.5

In this experiment, the depth of cut was specified as 0.7 mm and the width of cut was 1 mm as shown in [Figure 9.46](#). The nominal feedrate is 1.5 mm/s. In [Figure 9.47](#), when no control scheme is implemented, as the wheel wears, the grinding power increases gradually within one pass. In comparison, when the MLFC was implemented, the feedrate was reduced and the grinding power was maintained around its nominal value as shown in [Figure 9.48](#). ■

9.5 SIMULATION AND IMPLEMENTATION—FORCE CONTROL FOR MILLING PROCESSES

This section presents the application of the proposed MLFC to force control of milling processes. As shown in the simulation examples and experimental implementations, an improved system performance can be achieved under different machining conditions with parameter variations and process uncertainties compared with other conventional/intelligent control methods. The workpiece feedrate could

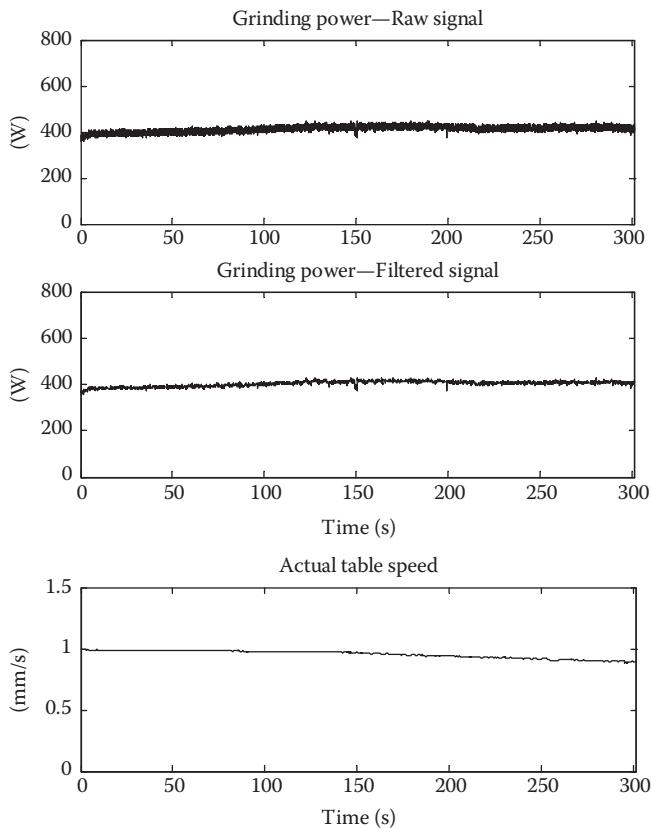


FIGURE 9.45 Power signals and control signal with the MLFC in Experiment 9.4.

be adjusted in real time. While maintaining the desired constant force level, the overall machining productivity was improved, and a nonoscillating system response with short settling time could be obtained. The successful implementations demonstrate the feasibility of the proposed adaptive fuzzy controller in controlling manufacturing processes in real industrial applications.

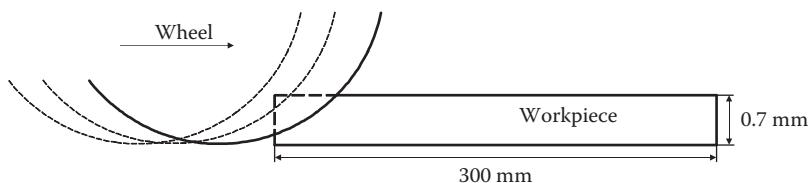


FIGURE 9.46 Illustration of the wheel wear experiment in Experiment 9.5.

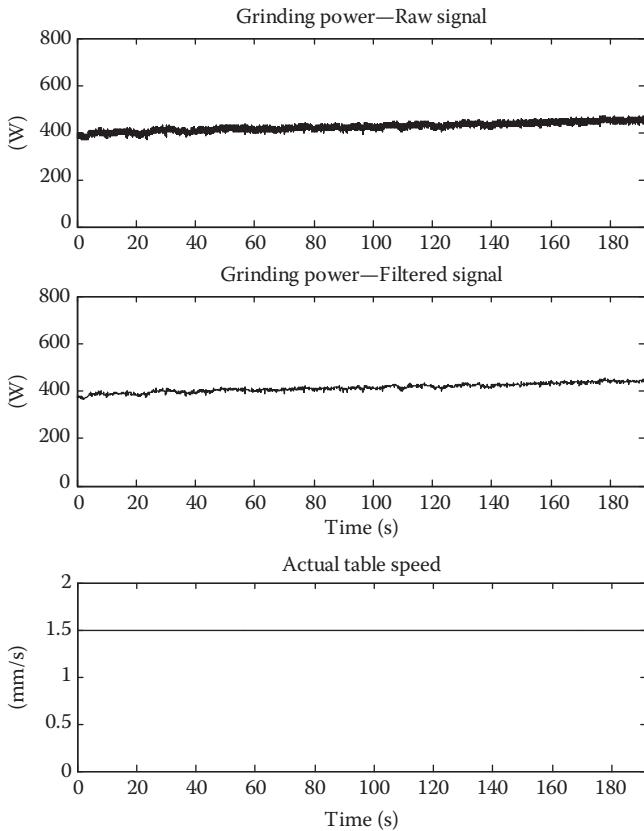


FIGURE 9.47 Power signals and control signal without any controller in Experiment 9.5.

9.5.1 SIMULATION EXAMPLES

In this section, two benchmarking simulation results are presented to demonstrate the performance of the proposed MLFC against some previous adaptive control schemes.

SIMULATION 9.1 Compared with a Direct Adaptive Controller

This simulation example considers an end milling process with step changes in axial depth of cut shown in Ma and Altintas (1990). The ZOH equivalent transfer function of the milling machine feed-drive was identified as a second-order system:

$$\frac{u(z)}{v(z)} = \frac{0.951807(z^{-1} + 0.0020946z^{-2})}{1 - 0.0461992z^{-1}} \quad (9.35)$$

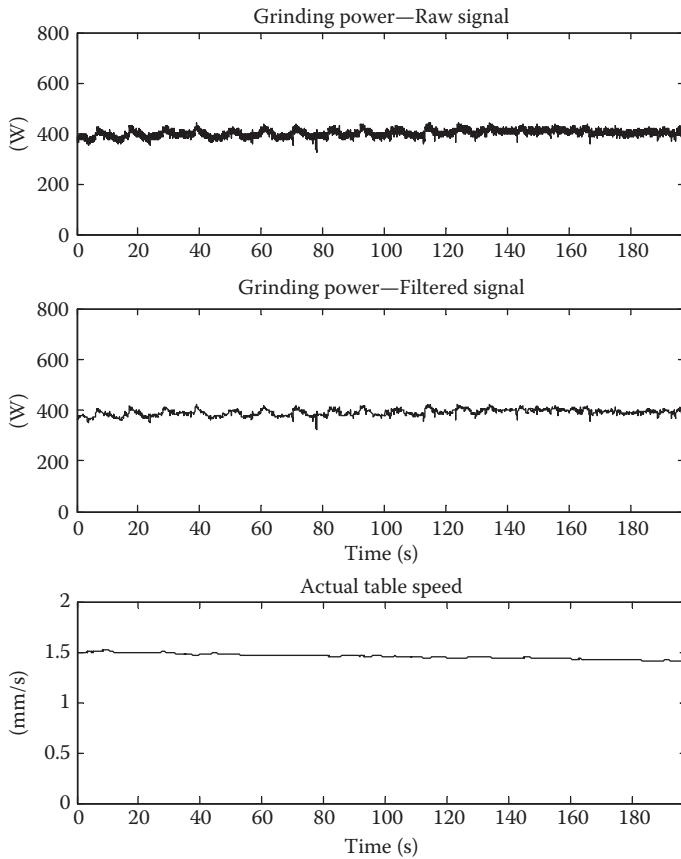


FIGURE 9.48 Power signals and control signal with the MLFC in Experiment 9.5.

where

$u(z)$ is the actual feed delivered by the feed-drive unit

$v(z)$ is the feed command to the CNC system

The chatter-free milling process was formulated as a time-varying first-order dynamic system (Ma and Altintas, 1990):

$$\frac{f(z)}{c(z)} = \frac{pz^{-1}}{1 - qz^{-1}} \text{ where } \mu = \frac{k_s \cdot c_1(b) \cdot a}{k_x}, \quad p = k_x \frac{\mu}{1 + \mu}, \quad q = \frac{\mu}{1 + \mu} \quad (9.36)$$

where

$f(z)$ is the maximum cutting force

$c(z)$ is the feedrate

k_s is a constant whose value depends on workpiece material and tool geometry

$c_1(b)$ is a parameter that depends on the number of cutting teeth, geometry of the cutter, and width of cut b
 a is the axial depth of cut
 k_x is the static stiffness of the cutter

The relation between u and c is assumed to be a fixed constant of 0.0491. In this simulation, the desired reference force level is given by $r(k) = 1,500(1 - e^{-k/5})$ N, where $k_s = 1,350$ N mm $^{-2}$, $c_1(b) = 1.3$, $b = 25.4$ mm, and $k_x = 12,100$ N mm $^{-2}$, while $a = 2.54$ mm for a distance of 22.2 mm, then 5.08 mm for 38.1 mm, 3.81 mm for 25.4 mm, and 5.08 mm for 15.9 mm, successively. For safety consideration, the feed command is bounded by $v_{\min} = 0$ counts s $^{-1}$, $v_{\max} = 8000$ counts s $^{-1}$, where 1 count = 0.00127 mm. The sampling period is chosen as $T_c = 38.7$ ms.

The proposed MLFC was executed to regulate the cutting force at the desired level. The actual cutting force is compared with the desired value and the control law is formulated by fuzzy inferencing calculation to provide the required feed commands to the CNC unit of the machine tool. The corresponding simulation result is shown in Figure 9.49. The upper plot shows the desired cutting force and the actual one, and the lower plot is the required feed command calculated by the MLFC. The result indicates that even with large changes in workpiece geometry, the proposed MLFC has the ability to regulate the cutting force at the desired value in a short transient time period without any oscillation.

In order to compare the system performance, a direct adaptive control approach by Altintas (1994) was implemented for the same milling process. The closed-loop

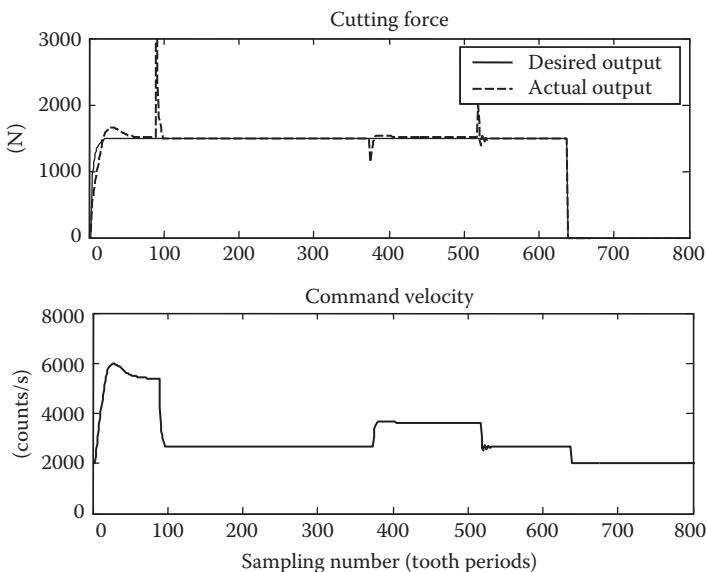


FIGURE 9.49 Simulation result of the MLFC in Simulation 9.1.

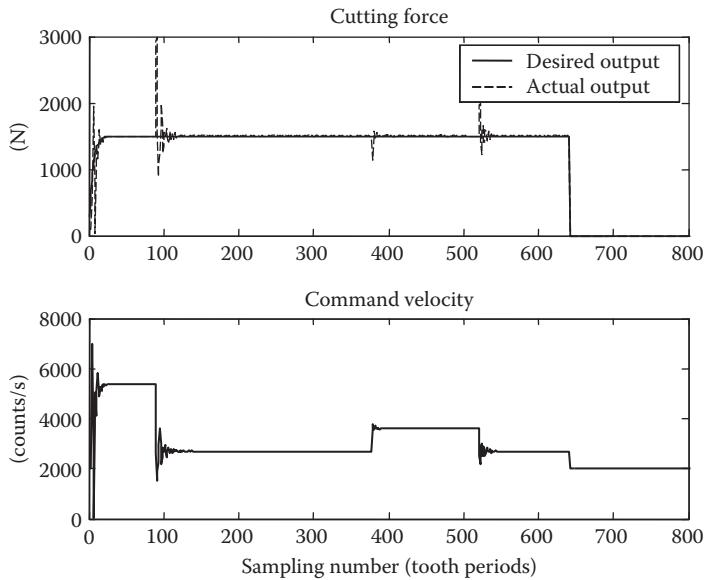


FIGURE 9.50 Simulation result of the direct adaptive controller in Simulation 9.1.

milling system was designed to work in the same manner as its open-loop behavior. The parameters of the adaptive controller were directly identified using the standard recursive least-square algorithm from the milling process. The required feedrate command to the CNC unit was calculated and sent out at each sampling period. The simulation response is shown in Figure 9.50. By comparing Figure 9.50 with Figure 9.49, it can be seen that the proposed fuzzy control method shows better system performance in shortening the transient time and reducing the output oscillation. Figure 9.51 shows the fuzzy controller surface between the control inputs and output. ■

SIMULATION 9.2 Compared with an Extended Model Reference Adaptive Controller

In this simulation example, the system performance of the MLFC is demonstrated for an end-milling process with a nonminimum phase zero (Rober and Shin, 1996):

$$\frac{f(z)}{u(z)} = \frac{z^{-1}(237.8 + 181.7z^{-1})}{1 - 1.39z^{-1} + 0.445z^{-2}} \quad (9.37)$$

which has the discrete zero at -0.76 and the discrete poles at 0.89 and 0.50 . $f(z)$ is the peak cutting force, and $u(z)$ represents the input signal. In the simulation, a workpiece with two-step changes in axial depth of cut is considered as shown in Figure 9.52. The first and third portions (depth of cut = 10 mm) represent the nominal

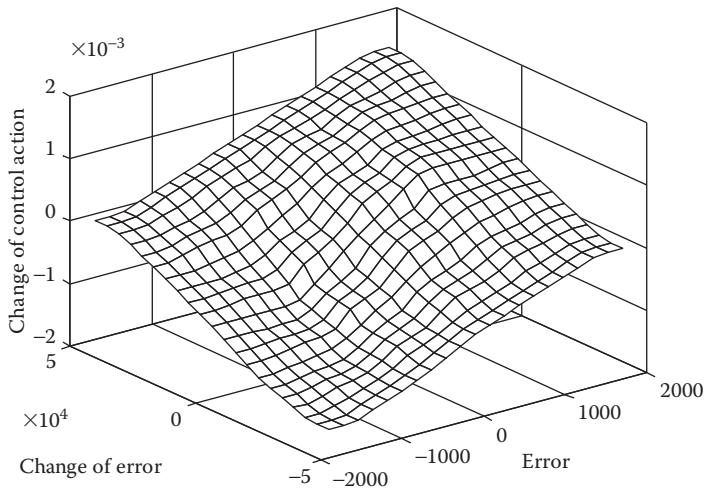


FIGURE 9.51 Fuzzy controller surface in Simulation 9.1.

plant in Equation 9.37, and the middle portion (depth of cut = 20 mm) doubles the plant gain.

The simulation example is intended to demonstrate a high-speed milling machine operation in a roughing stage. The desired force level is set to have a magnitude at 1500 N, since a high MRR can cause excessive vibration or even machine tool damage. On the other hand, the actual cutting force during milling processes is oscillatory in nature. Even with a constant input velocity, the peak force still fluctuates, due to spindle vibration, nonhomogeneity of material hardness, and dynamometer dynamics. Thus, a 5% noise signal is added to the output signal to simulate the random component of the online force measurement.

Figure 9.53a shows the simulation result of the MLFC. The vertical dotted lines denote changes in axial depth of cut. The reference force signal at 1500 N is drawn in Figure 9.53a for clear illustration. In the presence of drastic changes in depth of cut, the resultant system remains stable, the cutting force is maintained around 1500 N and the input oscillation is small. The extended model reference adaptive control (MRAC) method (Rober and Shin, 1996) incorporates zero-phase error tracking control into the standard MRAC system. The method employs a modified recursive least-squares estimation algorithm for online parameter identification. Figure 9.53b presents

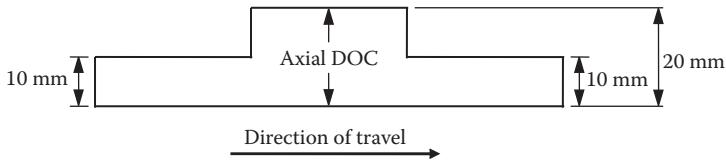


FIGURE 9.52 Cross section of workpiece in the simulation.

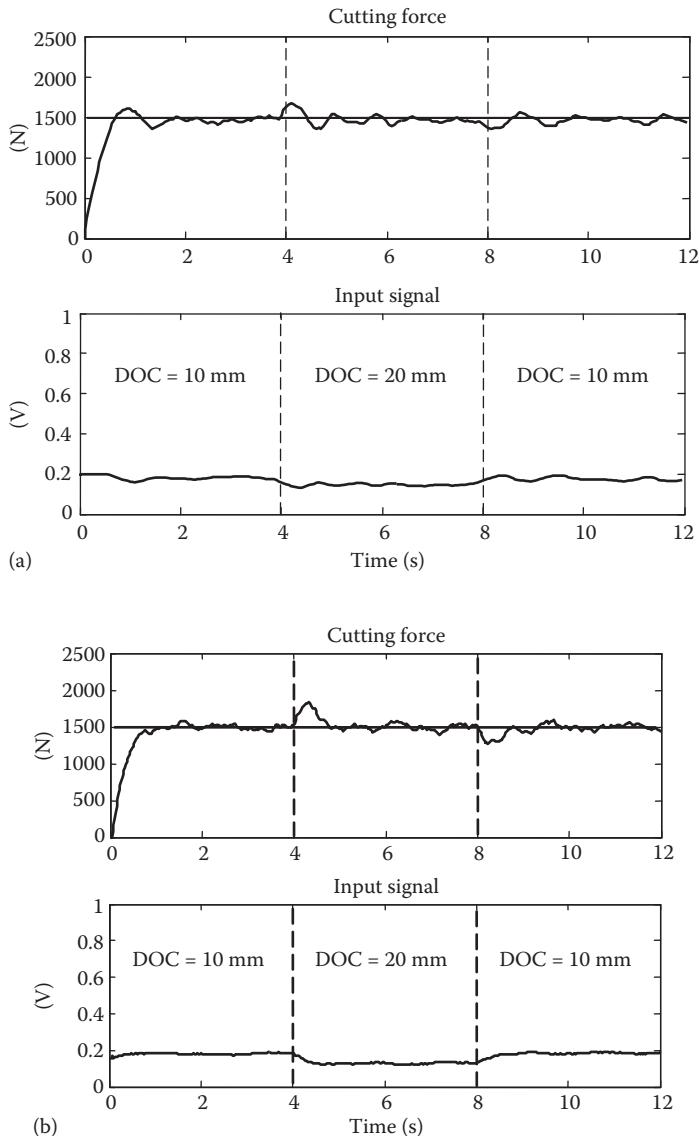


FIGURE 9.53 (a) Simulation result of the MLFC and (b) simulation result of the extended MRAC in Simulation 9.2.

simulation results for the extended MRAC with a weighting factor of 8000. The result shows similar characteristics as the case with the proposed MLFC in terms of providing good output tracking with small input oscillation. Both closed-loop systems remain stable in the presence of a near unstable zero. However, a satisfactory system performance of the extended MRAC depends on a judicious choice of the initial parameter

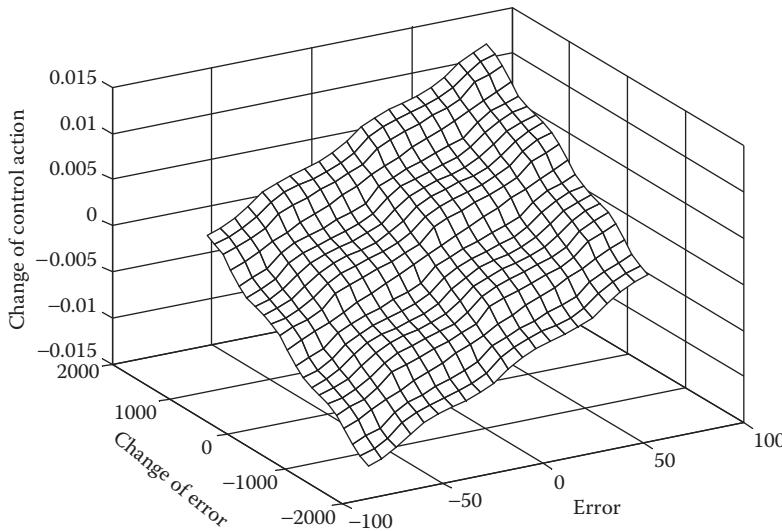


FIGURE 9.54 Fuzzy controller surface in Simulation 9.2.

estimation and weighting factor, and the computational burden is greater than the proposed MLFC for online operation. Figure 9.54 illustrates the fuzzy controller surface between the control inputs and outputs. ■

9.5.2 EXPERIMENTAL SETUP—HARDWARE CONFIGURATION

In this study, actual experiments were carried out on the CNC milling machine, which was controlled by digital signals with a NI real-time controller. Figure 9.55 shows the overall structure of the experimental setup for milling force control and Figure 9.56 is the picture for the end-milling experimental setup. The workpiece was mounted to the dynamometer, which was then mounted to the controlled machine table. The cutting force signal was read in through an analog I/O board

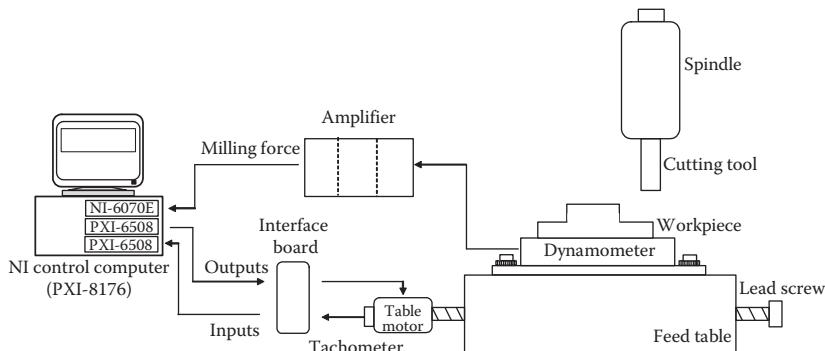


FIGURE 9.55 Experimental setup for milling force control.

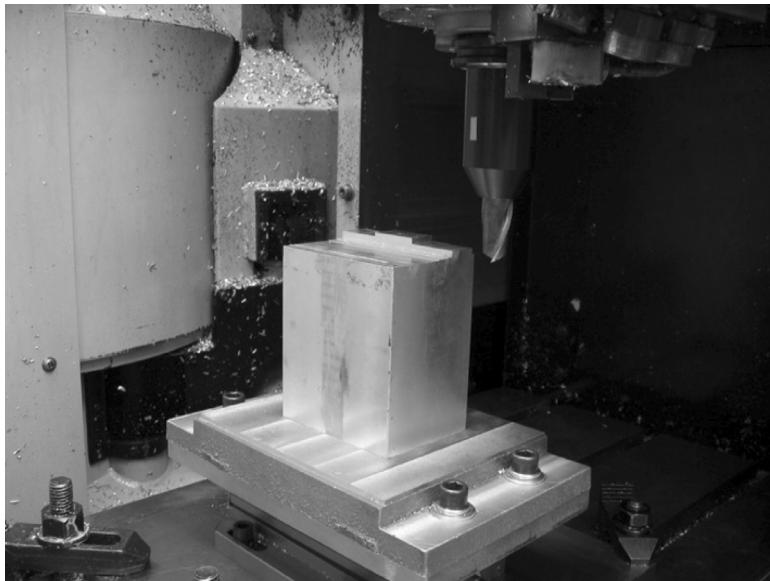


FIGURE 9.56 Picture for the end-milling experimental setup.

from the dynamometer. The force level during the end-milling process is fluctuating in nature. Therefore, the control computer searched for the maximum cutting force within each cutter rotation and used this peak level for control calculation. The required position was calculated from the MLFC and was sent to the milling machine at the next sampling time. A constant sampling time of 3.55 ms was maintained to be consistent with that of the CNC of the machine. A 3/4 in. diameter 10 flute end-mill was used, running at a constant rotational speed of 3000 rpm. The workpiece material was aluminum 7075T6, and the reference force was 500 N.

9.5.3 EXPERIMENTAL IMPLEMENTATION RESULTS

The proposed MLFC was implemented to maintain a constant cutting force level by controlling the feedrate of an end-milling process with depth of cut variation. Two sets of experiments were conducted with different cross sections of raw workpiece parts. In both cases, the milling force in the steady state was maintained at its maximum possible level, and an improved MRR was achieved with a minimum cycle time.

EXPERIMENT 9.6

In this case, the workpiece with a sloped surface was considered, the cross section of which is shown in [Figure 9.57](#). The width of cut was fixed at 2.54 mm. [Figure 9.58](#) shows the cutting force variation without any force controller, where the actual table

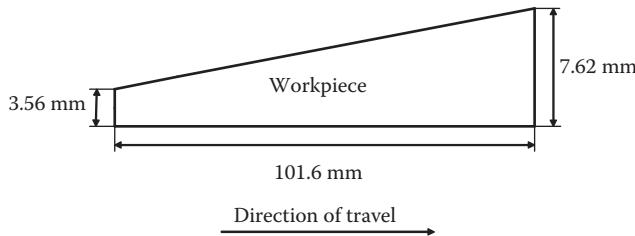


FIGURE 9.57 Cross section of workpiece in the end-milling process in Experiment 9.6.

speed was fixed at 1.27 mm/s (3 in./min). As the depth of cut changed from 3.56 mm to 7.62 mm, the force signal increased continuously from the entry point at around 400 N to the exiting point at 600 N. Next, a single-level FLC with fixed control parameters was used for system performance comparison. [Figure 9.59](#) shows the

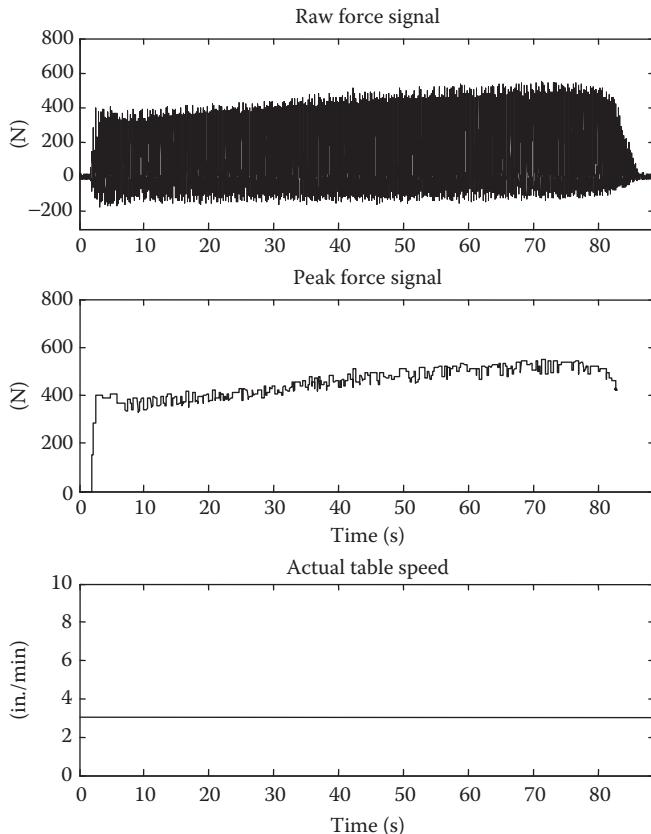


FIGURE 9.58 Force signals and control signal without any force controller in Experiment 9.6.

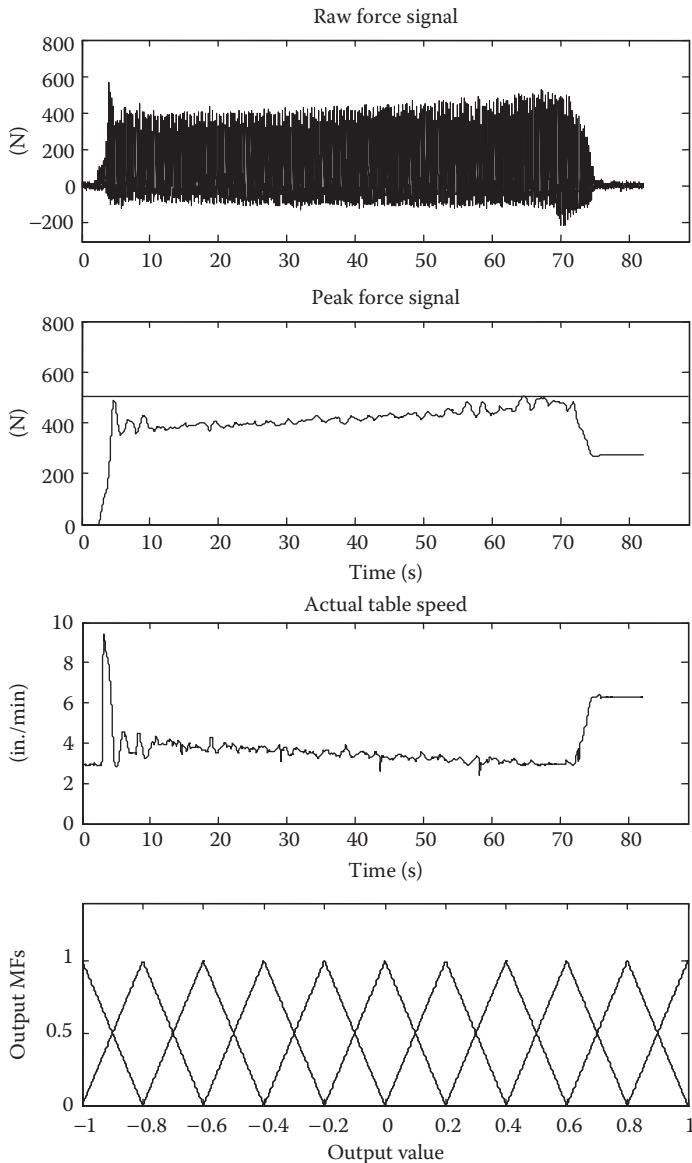


FIGURE 9.59 Force signals, control signal, and output MFs with regular FLC in Experiment 9.6.

force responses with this regular FLC. In this case, only one layer fuzzy controller was applied and the fuzzy rules were fixed in the entire operation period. From the second graph in Figure 9.59, it can be seen that the regular FLC with fixed fuzzy rules also lacks the ability to maintain the constant cutting force for the varying

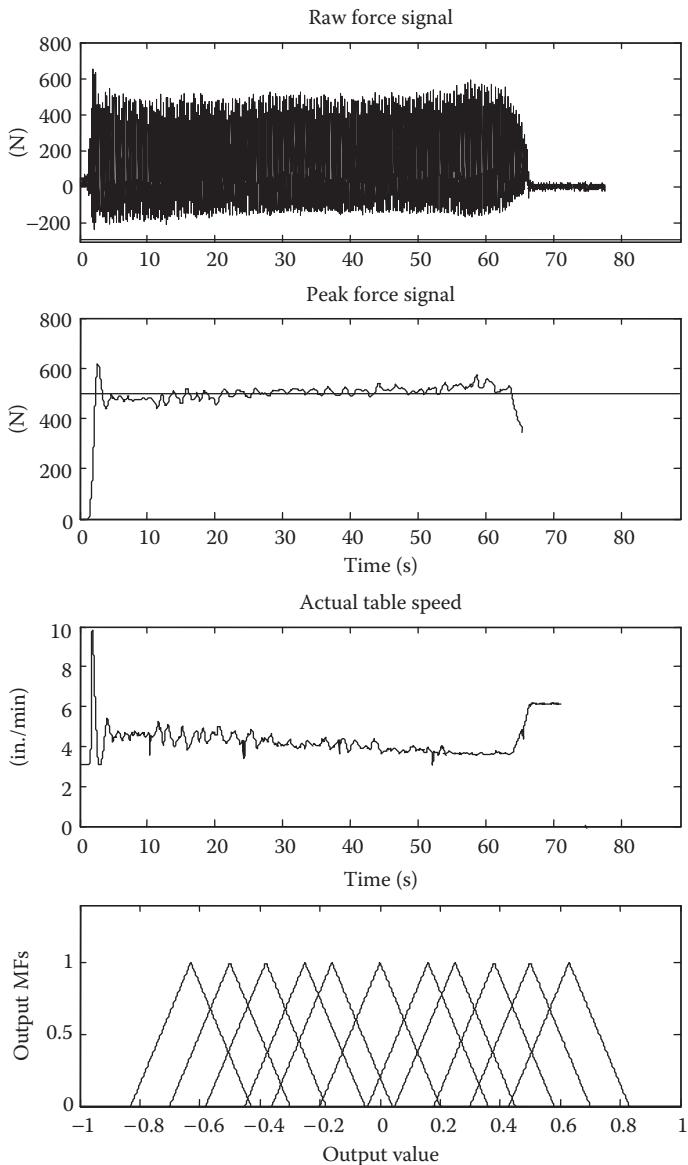


FIGURE 9.60 Force signals, control signal, and output MFs with MLFC in Experiment 9.6.

depth of cut. Finally, the experimental result for the MLFC system is presented in Figure 9.60, which shows its capabilities to tune control parameters quickly and maintain a constant force level by varying the table speed. The total cycle time is reduced from 85 s without any force controller to 72 s with regular FLC, and then

to 65 s with the MLFC, where the MRR is increased as a consequence. For easy comparison of the control performance, the reference force signal at 500 N is drawn in both plots of the peak force signal in Figures 9.59 and 9.60 in the cases with the regular FLC and MLFC.

The last plots of Figures 9.59 and 9.60 illustrate the output MFs at the end of the experiments. In order to make the figures clear, only the output MFs corresponding to zero value of change in error are plotted. Each of the 11 triangles is the output MF in the normalized universe of the control action. In the case with regular FLC, the output MFs are evenly distributed and are fixed in the entire operation process. On the other hand, the MLFC can tune the position of the output MFs with the embedded online adaptation mechanism and a better system performance can be achieved as a result. ■

EXPERIMENT 9.7

This experiment was conducted using a workpiece with two step changes in the axial depth of cut as shown in Figure 9.61. The first change is from 2.54 to 7.62 mm, and the second change reduces the depth to 2.54 mm. This allows a thorough performance evaluation of the MLFC for both a sudden increase and decrease in the depth of cut. The width of cut was fixed at 2.54 mm as in Experiment 9.6.

In the case when no force controller was implemented, as the depth of cut increased, the cutting force level increased substantially as expected. At the beginning of the operation cycle, since the depth of cut in the first step is only 2.54 mm, the milling force is pretty small, which is just around 200 N as shown in the second graph in Figure 9.62. The time period to finish this step is $28 - 2 = 26$ s. In contrast, the MLFC was able to adjust the force from 200 to 500 N in a very short time period ($20 - 2 = 18$ s), as illustrated in the second plot in Figure 9.64. In the middle part of the workpiece where the milling force is roughly about its maximum value, the MLFC was able to decrease the feedrate correspondingly. In this experiment, the cycle time was reduced from around 85 s with no force controller to around 56 s with the MLFC, which is about 34% reduction. The reference force signals at 500 N are drawn in the

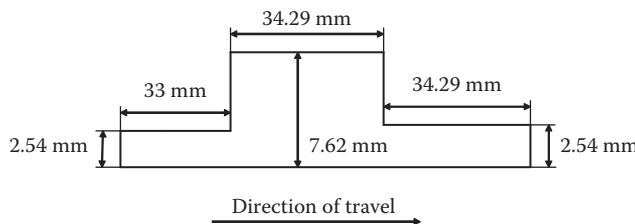


FIGURE 9.61 Cross section of workpiece in the end-milling process in Experiment 9.7.

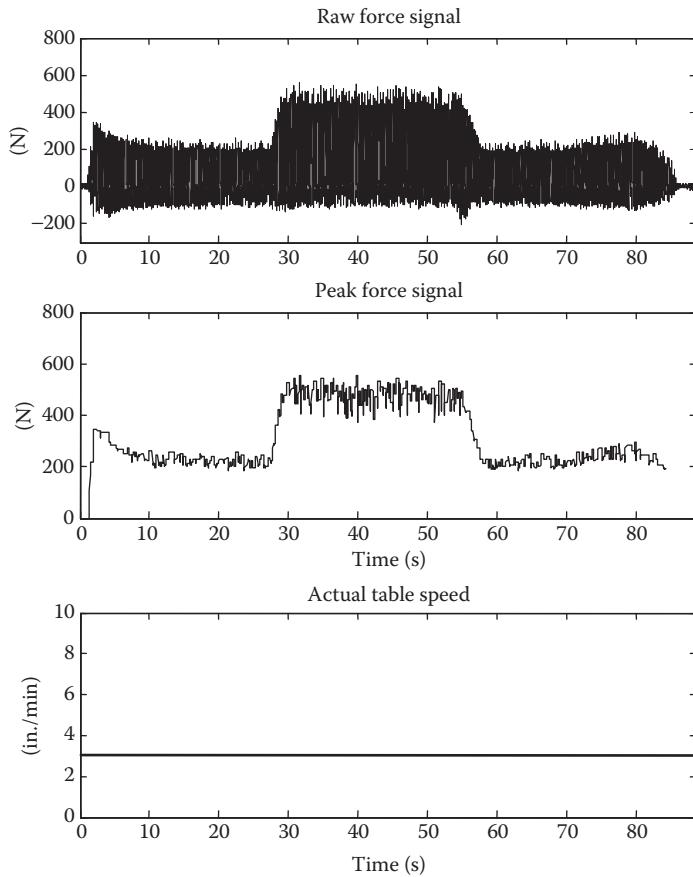


FIGURE 9.62 Force signals and control signal without any force controller in Experiment 9.7.

plot of the peak force signal in Figures 9.63 and 9.64 for easy comparison, and the output MFs after the experiments are shown in the last plots. ■

The two experimental results are summarized in Table 9.6. In Experiment 9.6, a workpiece with sloped surface was considered, where the total cycle time was reduced by 15% with a regular FLC, and 24% with the proposed MLFC. In Experiment 9.7, when the workpiece has step changes in the axial depth of cut, the cycle time was reduced by 22% with a regular FLC, and by 34% with the MLFC. These experiments are designed to simulate the actual machining processes in real industry, where the raw workpiece surface will not always be flat. It will have some slopes or some step changes as designed in the experiments. When the depth of cut is small, the feedrate can be increased to maintain the milling force at

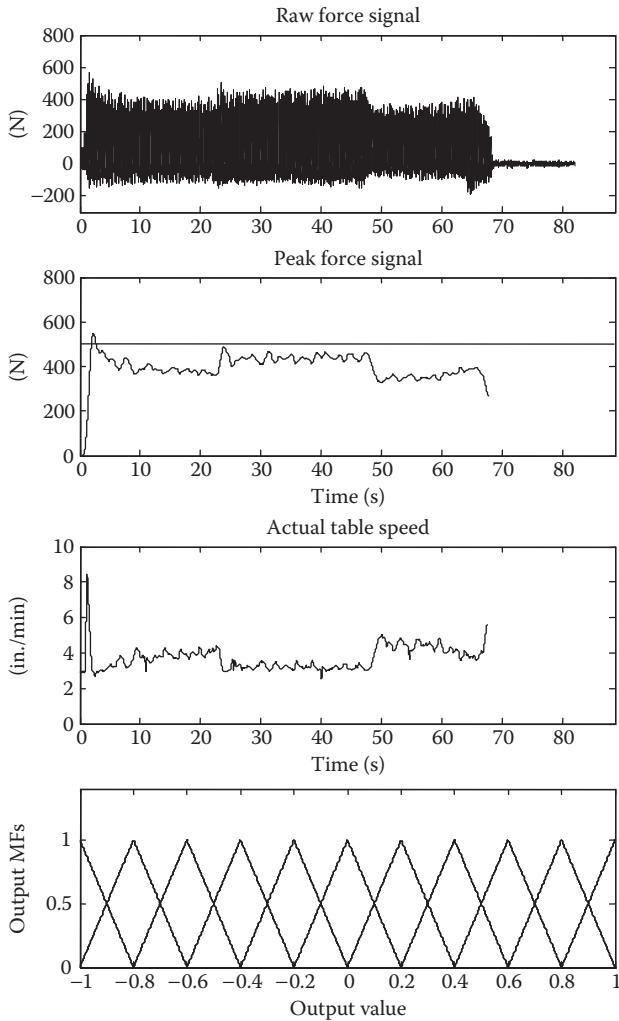


FIGURE 9.63 Force signals, control signal, and output MFs with regular FLC in Experiment 9.7.

the reference level, and the cycle time can be reduced as a result. The comparison results in Table 9.6 illustrates that the proposed MLFC method is an effective ways of reducing the total cycle time and increasing the MRR.

9.6 CONCLUSION

In this chapter, an MLFC system is proposed, which can adaptively compensate for external disturbances and time-varying parameters. An adaptive fuzzy controller is designed to tune the parameters of the first layer fuzzy controller based on the system performance. It is shown that a limited number of fuzzy rules are adequate to control

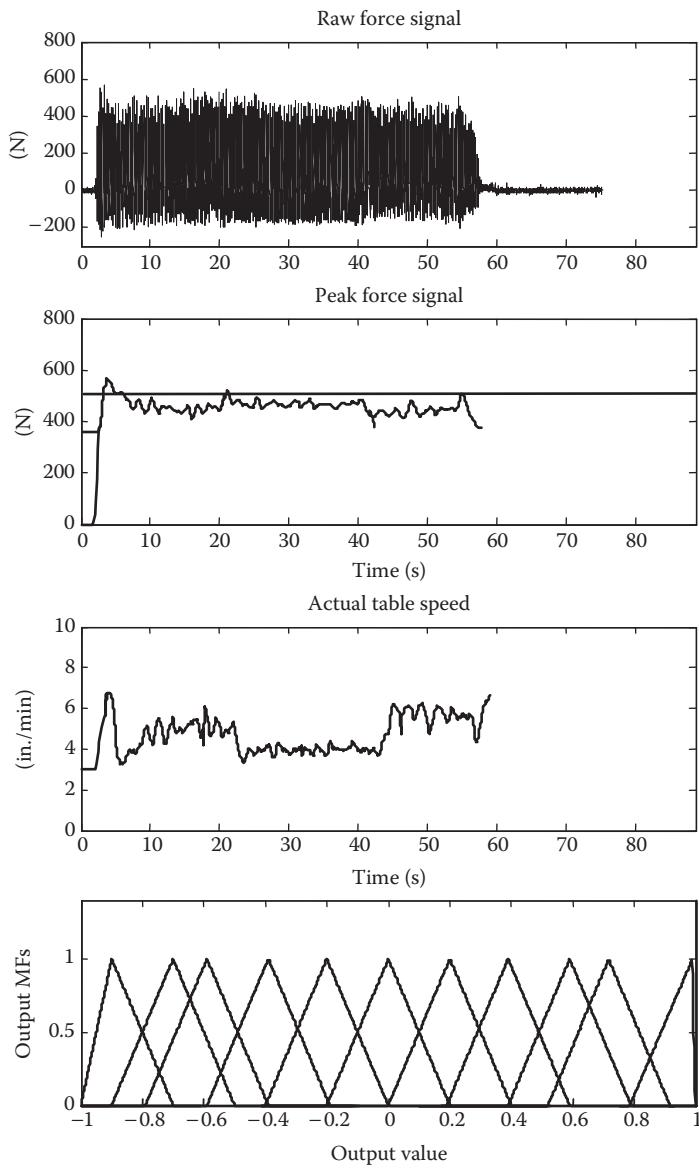


FIGURE 9.64 Force signals, control signal, and output MFs with MLFC in Experiment 9.7.

the dynamics of unknown systems and provide significant computation savings. The stability of the proposed control algorithm is guaranteed in the sense of input–output passivity. A series of simulations and experiments for nonlinear systems demonstrate the effectiveness of the control structure in dealing with systems with uncertainties and disturbances.

TABLE 9.6
Summary on Experimental Results

	Experiment 9.6 (Sloped Surface)		Experiment 9.7 (Stepped Surface)	
	Cycle Time (s)	Cycle Time Reduction (%)	Cycle Time (s)	Cycle Time Reduction (%)
Without force controller	85	0	85	0
With regular FLC	72	15	66	22
With MLFC	65	24	56	34

REFERENCES

- Altintas, Y., Direct adaptive control of end milling process, *International Journal of Machine Tools and Manufacture*, 34(4): 461–472, 1994.
- Amitay, G., Malkin, S., and Koren, Y., Adaptive control optimization of grinding, *Transactions of the ASME, Journal of Engineering for Industry*, 103: 103–108, 1981.
- Aracil, J. and Gordillo, F., *Stability Issues in Fuzzy Control*, Physica-Verlag, Heidelberg, Germany, 2000.
- Brenner, N. and Torrance, A.A., Wheel sharpness measurement for force prediction in grinding, *Wear*, 160: 317–323, 1993.
- Calcev, G., Some remarks in the stability of Mamdani fuzzy control systems, *IEEE Transactions on Fuzzy Systems*, 6(3): 436–442, 1998.
- Chiu, N. and Malkin, S., Computer simulation for cylindrical plunge grinding, *Annals of the CIRP*, 42(1): 383–387, 1993.
- Farinwata, S.S., Filev, D., and Langari, R., *Fuzzy Control Synthesis and Analysis*, John Wiley & Sons, Ltd., Chichester, United Kingdom, 2000.
- Furukawa, Y. and Ohishi, S., Adaptive control of creep feed grinding to avoid workpiece burn, *Proceedings of the 5th International Conference on Production Engineering*, Tokyo, Japan, pp. 64–69, 1984.
- Haber, R.E., Haber, R.H., Alique, A., and Ros, S., Hierarchical fuzzy control of the milling process with a self-tuning algorithm, *Proceedings of the 2000 IEEE International Symposium on Intelligent Control*, Rio Patras, Greece, pp. 115–120, 2000.
- Hsu, P.-L. and Fann, W.-R., Fuzzy adaptive control of machining processes with a self-learning algorithm, *ASME Journal of Manufacturing Science and Engineering*, 118: 522–530, 1996.
- Khalil, H.K., *Nonlinear Systems*, Prentice Hall, Upper Saddle River, NJ, 2002.
- Kirk, G.K., Wheel sharpness measurement for force prediction in grinding, *Wear*, 160: 317–323, 1976.
- Layne, J.R. and Passino, K.M., Fuzzy model reference learning control for cargo ship steering, *IEEE Control Systems Magazine*, 13(6): 23–34, 1993.
- Lee, C.C., Fuzzy logic in control system: Fuzzy logic controller—Part I, *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2): 404–418, 1990.
- Linkens, D.A. and Abbot, M.F., Self-organising fuzzy logic control and the selection of its scaling factors, *Transaction of the Institute of Measurement and Control*, 14(3): 114–125, 1992.
- Ma, C.C.H. and Altintas, Y., Direct adaptive cutting force control of milling processes, *Automatica*, 26(5): 899–902, 1990.

- Malkin, S., Burning limits for surface and cylindrical grinding of steels, *Annals of CIRP*, 27(1): 233–236, 1978.
- Malkin, S. and Anderson, R.B., Thermal aspects of grinding: Part I—Energy partition, *ASME Transactions: Journal of Engineering for Industry*, 96: 1177–1183, 1974.
- Malkin, S. and Cook, N.H., The wear of grinding wheels, Part I—Attritious wear, *ASME Transactions: Journal of Engineering for Industry*, 93: 1120–1128, 1971.
- Nounou, H.N. and Passino, K.M., Fuzzy model predictive control: Techniques, stability issues, and examples, *Proceedings of the IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, Cambridge, MA, pp. 423–428, 1999.
- Nounou, H.N. and Passino, K.M., Stable auto-tuning of the adaptation gain for indirect adaptive control, *Proceedings of the American Control Conference*, Chicago, IL, pp. 2159–2163, 2000.
- Passino, K.M. and Yurkovich, S., *Fuzzy Control*, Addison-Wesley, Reading, MA, 1998.
- Procyk, T.J. and Mamdani, E.H., A linguistic self-organizing process controller, *Automatica*, 15(1): 15–30, 1979.
- Rober, S. and Shin, Y.C., Modeling and control of CNC machines using a PC-based open architecture controller, *Mechatronics*, 5(4): 401–420, 1995.
- Rober, S.J. and Shin, Y.C., Control of cutting force for end milling processes using an extended model reference adaptive control scheme, *Transactions of the ASME, Journal of Manufacturing Science and Engineering*, 118: 339–347, 1996.
- Saini, D.P., Wheel hardness and local elastic deflections in grinding, *International Journal of Machine Tools and Manufacture*, 30(4): 637–649, 1990.
- Saini, D.P. and Wager, J.G., Local contact deflections and forces in grinding, *Annals of the CIRP*, 34(1): 281–285, 1985.
- Slotine, J.-J.E., *Applied Nonlinear Control*, Prentice Hall, Englewood Cliffs, NJ, 1991.
- Xu, C. and Shin, Y.C., Design of a multi-level fuzzy controller for nonlinear systems and stability analysis, *IEEE Transactions on Fuzzy Systems*, 13(6): 761–778, 2005.
- Ying, H., Practical design of nonlinear fuzzy controllers with stability analysis for regulating processes with unknown mathematical models, *Automatica*, 30(7): 1185–1195, 1994.
- Younis, M., Sadek, M.M. and El-Wardani, T., A new approach to development of a grinding force model, *Transactions of the ASME, Journal of Engineering for Industry*, 109: 306–313, 1987.

10 Intelligent Control for MISO Nonlinear Systems

In this chapter, an adaptive fuzzy controller is developed for general multi-input single-output (MISO) nonlinear systems with time-varying characteristics and unknown system uncertainties. The system dynamics is modeled in fuzzy domain and a fuzzy inverse model is automatically constructed, which has the ability of uniquely determining the inverse relationship for each input–output pair (Xu and Shin, 2007). Based on this inverse model, a multilevel fuzzy controller (MLFC) is designed based on the fuzzy inverse model, which has an embedded hierarchical structure for each control variable to compensate for the time-varying parameters and external disturbance. The stability of the MLFC-MISO controller is proved using input–output passivity theory. Finally, two simulation examples are presented to illustrate the effectiveness of this controller for the multivariable system under various system conditions with plant parameter variations and system uncertainties.

10.1 MLFC-MISO CONTROL SYSTEM STRUCTURE

Many real-world systems are complex and ill-defined with unknown plant variations and disturbances. Classical control techniques need precise system models and parameters and often encounter a major difficulty if the accurate knowledge of the plant is unavailable. New methods for designing an effective controller for such complex systems without precise models are thus necessary. Intelligent controllers based on neural networks and fuzzy logic are attractive alternative approaches since they do not require accurate mathematical expressions of the plant. They provide the advantage of easy design over the classical control methods and have been implemented for various complex and nonlinear single systems. Furthermore, the successful applications of fuzzy control techniques for single variable systems fueled their extension and application to the control of multivariable systems. However, in most cases, multivariable fuzzy controllers are designed based on human knowledge, which is a rough description of system inputs/outputs relationships. An inaccuracy in rule table construction is unavoidable as the complexity of the system increases, especially in the presence of high nonlinearity and strong interactions. The complicated interaction effects among each variable may lead to an incorrect fuzzy inferencing result, thereby reducing system control performance. Additionally, due to the inherent complexity of multivariable systems, multiple combinations of

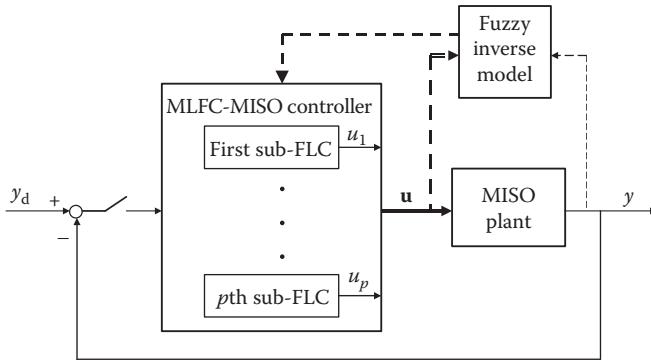


FIGURE 10.1 Block diagram of the MLFC-MISO control system.

system inputs can lead to a same system output. In other words, a same output may be the result of multiple choices of input combinations. As a result in the control process, it is difficult to uniquely determine the appropriate control actions based on the current system performance.

In this chapter, an MLFC is introduced for a general MISO system where the control rules are determined based on a system fuzzy inverse model. In the design procedure, a fuzzy forward model is first used to approximate the MISO nonlinear system dynamics and then the inverse relationship of the system input–output pair is uniquely determined. The MLFC-MISO controller is constructed with an adaptation mechanism to tune the control parameters online.

Figure 10.1 is the closed-loop control structure for general MISO nonlinear systems, whose inputs are $\mathbf{u} = [u_1, \dots, u_p]^T \in \mathbb{R}^p$ and the scalar output is y . The MISO system is modeled in fuzzy domain and its inverse model is systematically formulated based on the procedure explained in Chapter 4, which is able to uniquely determine the inverse relationship for each input–output pair of this MISO system. The fuzzy inferencing calculation result from the fuzzy inverse model will be used to initialize the parameters of the MLFC-MISO controller. The big box in Figure 10.1 in the middle represents the MLFC-MISO controller, which is composed of p subfuzzy logic controllers (FLCs) for each control variable $u_r(k)$, $r = 1, \dots, p$. They work in parallel and the multiple control actions $\mathbf{u} = [u_1, \dots, u_p]^T$ are obtained and used to control this MISO system at the same time.

10.1.1 CONTROL PARAMETERS INITIALIZATION

Consider a MISO system, where the forward model provides a unique mapping. In other words, for a given set of inputs, there is only one solution for the output. However, in the inverse relationship, the same system output can result from different combinations of system inputs. It is difficult to model the inverse relationship directly using the input–output training data. In this section, a procedure of constructing the fuzzy inverse model for the MISO system, which is used to initialize the parameters of the MLFC-MISO controller, is presented. Consider a nonlinear MISO dynamic system with p inputs in the form of input–output expression:

$$\begin{aligned}
y(k) &= f[y(k-1), y(k-2), \dots, y(k-n+1), y(k-n), \\
&\quad u_1(k-1), u_1(k-2), \dots, u_1(k-m_1+1), u_1(k-m_1), \\
&\quad \dots \\
&\quad u_p(k-1), u_p(k-2), \dots, u_p(k-m_p+1), u_p(k-m_p)] \quad (10.1)
\end{aligned}$$

where

u_1, \dots, u_p are the p inputs to the system

y is the scalar output

$f(\cdot)$ is a smooth function representing the system's nonlinear dynamics

m_1, \dots, m_p are the maximum delay terms for each input variable

n is the delay term for the system output

After shifting one sampling time forward,

$$\begin{aligned}
y(k+1) &= f[y(k), y(k-1), \dots, y(k-n+2), y(k-n+1), \\
&\quad u_1(k), u_1(k-1), \dots, u_1(k-m_1+2), u_1(k-m_1+1), \\
&\quad \dots \\
&\quad u_p(k), u_p(k-1), \dots, u_p(k-m_p+2), u_p(k-m_p+1)] \quad (10.2)
\end{aligned}$$

one can obtain

$$\begin{aligned}
\Delta y(k+1) &= y(k+1) - y(k) \\
&= f[y(k), y(k-1), \dots, y(k-n+2), y(k-n+1), \\
&\quad u_1(k), u_1(k-1), \dots, u_1(k-m_1+2), u_1(k-m_1+1), \\
&\quad \dots \\
&\quad u_p(k), u_p(k-1), \dots, u_p(k-m_p+2), u_p(k-m_p+1)] - y(k) \\
&= f_1[y(k), y(k-1), \dots, y(k-n+2), y(k-n+1), \\
&\quad u_1(k), u_1(k-1), \dots, u_1(k-m_1+2), u_1(k-m_1+1), \\
&\quad \dots \\
&\quad u_p(k), u_p(k-1), \dots, u_p(k-m_p+2), u_p(k-m_p+1)] \quad (10.3)
\end{aligned}$$

Denote

$\Delta y(k) = y(k) - y(k-1)$, $\Delta y(k-1) = y(k-1) - y(k-2)$, \dots , $\Delta u_1(k) = u_1(k) - u_1(k-1)$, \dots , $\Delta u_p(k) = u_p(k) - u_p(k-1)$. Then the above equation can be represented as

$$\begin{aligned}
\Delta y(k+1) &= f_2[\Delta y(k), \Delta y(k-1), \dots, \Delta y(k-n+3), \Delta y(k-n+2), \\
&\quad \Delta u_1(k), \Delta u_1(k-1), \dots, \Delta u_1(k-m_1+3), \Delta u_1(k-m_1+2), \\
&\quad \dots \\
&\quad \Delta u_p(k), \Delta u_p(k-1), \dots, \Delta u_p(k-m_p+3), \Delta u_p(k-m_p+2)] \quad (10.4)
\end{aligned}$$

Suppose all the input and output variables are measurable, the MISO system (Equation 10.4) can be represented by the following fuzzy model as

$$\begin{aligned}
 R^i: & \text{IF } y(k) = A^i \text{ AND } \Delta y(k) = B^i \text{ AND } \dots \text{ AND } \Delta y(k-n+2) = B_{(n-2)}^i \text{ AND} \\
 & u_1(k) = A_1^i \text{ AND } \Delta u_1(k) = D_1^i \text{ AND } \dots \text{ AND } \Delta u_1(k-m_1+2) = D_{1(m_1-2)}^i \text{ AND} \\
 & \quad \dots \\
 & u_p(k) = A_p^i \text{ AND } \Delta u_p(k) = D_p^i \text{ AND } \dots \text{ AND } \Delta u_p(k-m_p+2) = D_{p(m_p-2)}^i \text{ AND}, \\
 \text{THEN } & \Delta y(k+1) = c^i
 \end{aligned} \tag{10.5}$$

where

R^i ($i = 1, 2, \dots, l$) represents the i th rule of the fuzzy model, l is the number of fuzzy rules

$A^i, B^i, \dots, B_{(n-2)}^i, A_1^i, D_1^i, \dots, D_{1(m_1-2)}^i, A_p^i, D_p^i, \dots, D_{p(m_p-2)}^i$ are triangular membership functions (MFs)

c^i represents a singleton MF

Define the center of the triangular MF D_r^i at position d_r^i , such that $\ker(D_r^i) = d_r^i$, where $\ker(D_r^i) = \{d_r^i | \mu_{D_r^i}[u_r(k)] = 1\}$, $r = 1, 2, \dots, p$. Assume the l triangular MF D_r^i for the system input $u_r(k)$ in each rule ($i = 1, 2, \dots, l$) overlaps adjacently as $\sum_{i=1}^l \mu_{D_r^i}[u_r(k)] = 1$.

For simplification, a multidimensional fuzzy set \mathbf{Y}^i is defined for the vector $\mathbf{y}(k) = [y(k), \Delta y(k), \Delta y(k-1), \dots, \Delta y(k-n+3), \Delta y(k-n+2)]^T$ by applying the min operator in the Cartesian product space $\mathbf{Y}^i = A^i \times B^i \times \dots \times B_{(n-2)}^i$ with the corresponding membership degree as

$$\begin{aligned}
 \mu_{\mathbf{Y}^i}[\mathbf{y}(k)] &= \mu_{A^i \times B^i \times \dots \times B_{(n-2)}^i}(y(k), \Delta y(k), \dots, \Delta y(k-n+2)) \\
 &= \min(\mu_{A^i}[y(k)], \mu_{B^i}[\Delta y(k)], \dots, \mu_{B_{(n-2)}^i}[\Delta y(k-n+2)]) \\
 &= \mu_{A^i}[y(k)] \wedge \mu_{B^i}[\Delta y(k)] \wedge \dots \wedge \mu_{B_{(n-2)}^i}[\Delta y(k-n+2)]
 \end{aligned} \tag{10.6}$$

Similarly, a multidimensional fuzzy set \mathbf{U}_1^i is defined for the vector $\mathbf{u}_1(k) = [u_1(k), \Delta u_1(k-1), \dots, \Delta u_1(k-m_1+3), \Delta u_1(k-m_1+2)]^T$ by applying the min operator in the Cartesian product space $\mathbf{U}_1^i = A_1^i \times D_1^i \times \dots \times D_{1(m_1-2)}^i$ with the corresponding membership degree as

$$\begin{aligned}
 \mu_{U_1^i}[\mathbf{u}_1(k)] &= \mu_{A_1^i \times D_1^i \times \dots \times D_{1(m_1-2)}^i}(u_1(k), \Delta u_1(k-1), \dots, \Delta u_1(k-m_1+2)) \\
 &= \min(\mu_{A_1^i}[u_1(k)], \mu_{D_1^i}[\Delta u_1(k-1)], \dots, \mu_{D_{1(m_1-2)}^i}[\Delta u_1(k-m_1+2)]) \\
 &= \mu_{A_1^i}[u_1(k)] \wedge \mu_{D_1^i}[\Delta u_1(k-1)] \wedge \dots \wedge \mu_{D_{1(m_1-2)}^i}[\Delta u_1(k-m_1+2)]
 \end{aligned} \tag{10.7}$$

A multidimensional fuzzy set \mathbf{U}_p^i is defined for the vector $\mathbf{u}_p(k) = [u_p(k), \Delta u_p(k-1), \dots, \Delta u_p(k-m_p+3), \Delta u_p(k-m_p+2)]^T$ by applying the min operator in the Cartesian

product space $\mathbf{U}_p^i = A_p^i \times D_p^i \times \cdots \times D_{p(m_p-2)}^i$ with the corresponding membership degree as

$$\begin{aligned}\mu_{U_p^i}[\mathbf{u}_p(k)] &= \mu_{A_p^i \times D_p^i \times \cdots \times D_{p(m_p-2)}^i}(u_p(k), \Delta u_p(k-1), \dots, \Delta u_p(k-m_p+2)) \\ &= \min(\mu_{A_p^i}[u_p(k)], \mu_{D_p^i}[\Delta u_p(k-1)], \dots, \mu_{D_{p(m_p-2)}^i}[\Delta u_p(k-m_p+2)]) \\ &= \mu_{A_p^i}[u_p(k)] \wedge \mu_{D_p^i}[\Delta u_p(k-1)] \wedge \cdots \wedge \mu_{D_{p(m_p-2)}^i}[\Delta u_p(k-m_p+2)]\end{aligned}\quad (10.8)$$

And a multidimensional fuzzy set \mathbf{X}^i is defined for the vector $\mathbf{x}(k) = [\mathbf{y}(k), \mathbf{u}_1(k), \dots, \mathbf{u}_p(k)]^T$ by applying the min operator in the Cartesian product space $\mathbf{X}^i = \mathbf{Y}^i \times \mathbf{U}_1^i \times \cdots \times \mathbf{U}_p^i$ with the corresponding membership degree as

$$\begin{aligned}\mu_{\mathbf{X}^i}[\mathbf{x}(k)] &= \mu_{\mathbf{Y}^i \times \mathbf{U}_1^i \times \cdots \times \mathbf{U}_p^i}(\mathbf{y}(k), \mathbf{u}_1(k), \dots, \mathbf{u}_p(k)) \\ &= \min(\mu_{\mathbf{Y}^i}[\mathbf{y}(k)], \mu_{\mathbf{U}_1^i}[\mathbf{u}_1(k)], \dots, \mu_{\mathbf{U}_p^i}[\mathbf{u}_p(k)]) \\ &= \mu_{\mathbf{Y}^i}[\mathbf{y}(k)] \wedge \mu_{\mathbf{U}_1^i}[\mathbf{u}_1(k)] \wedge \cdots \wedge \mu_{\mathbf{U}_p^i}[\mathbf{u}_p(k)]\end{aligned}\quad (10.9)$$

Therefore, the fuzzy rules for the system dynamics can be simplified as

$$\begin{aligned}R^i: \text{IF } \mathbf{x}(k) = \mathbf{X}^i \text{ AND } \\ \Delta u_1(k) = D_1^i \text{ AND } \Delta u_2(k) = D_2^i \text{ AND } \dots \text{ AND } \Delta u_p(k) = D_p^i, \\ \text{THEN } \Delta y(k+1) = c^i\end{aligned}\quad (10.10)$$

As derived in Chapter 4, define a series of trapezoidal MF $\overline{D}_1^i, \overline{D}_2^i, \dots, \overline{D}_p^i$ for the p input variables and each individual membership degree is denoted as $\mu_{\overline{D}_r^i}[u_r(k)] = \mu_{\mathbf{X}^i}[\mathbf{x}(k)] \wedge \mu_{D_r^i}[u_r(k)]$. The height of the fuzzy MF is $\mu_{\mathbf{X}^i}[\mathbf{x}(k)]$, which can be calculated from Equation 10.9. The i th fuzzy rule can be expressed as

$$\begin{aligned}R^i: \text{IF } \Delta u_1(k) = \overline{D}_1^i \text{ AND } \Delta u_2(k) = \overline{D}_2^i \text{ AND } \dots \text{ AND } \Delta u_p(k) = \overline{D}_p^i, \\ \text{THEN } \Delta y(k+1) = c^i\end{aligned}\quad (10.11)$$

THEOREM 10.1

Suppose at any given time instant, the state vector is measurable as $\mathbf{x}(k)$. The desired system output is known as y_d and then the desired output for the fuzzy forward system (Equation 10.11) is $\Delta y_d(k+1) = y_d - y(k)$. The fuzzy forward model (Equation 10.11) of the dynamic MISO nonlinear system is invertible if and only if in each rule, $d_r^m \geq d_r^n \rightarrow c^m \geq c^n$ (or $d_r^m < d_r^n \rightarrow c^m < c^n$), $\forall r \in [1, p], \forall m, n, m \neq n$. The fuzzy inverse model for each system input $\Delta u_r(k)$ is constructed as

$$\begin{aligned}
\text{IR}^i: \text{IF } \Delta y(k+1) = C^i, \text{ THEN } \Delta u_1(k) = \bar{d}_1^i \\
\text{IF } \Delta y(k+1) = C^i, \text{ THEN } \Delta u_2(k) = \bar{d}_2^i \\
&\vdots \\
\text{IF } \Delta y(k+1) = C^i, \text{ THEN } \Delta u_p(k) = \bar{d}_p^i
\end{aligned} \tag{10.12}$$

where IR^i ($i = 1, 2, \dots, l$) represents the i th rule of the fuzzy inverse model. Denote the output MF vector as $\mathbf{d}^i = [\bar{d}_1^i, \dots, \bar{d}_p^i]^T$, and then the fuzzy inverse model can be expressed in a vector form as

$$\text{IR}^i: \text{IF } \Delta y(k+1) = C^i, \text{ THEN } \Delta \mathbf{u}(k) = \bar{\mathbf{d}}^i \tag{10.13}$$

Denotation

C^i : A triangular MF for output $\Delta y(k+1)$, whose center is at position c^i , such that $\ker(C^i) = c^i$ and $\ker(C^i) = \{c^i | \mu_{C^i}[\Delta y(k+1)] = 1\}$. The l triangular MFs C^i in each rule ($i = 1, 2, \dots, l$) overlap adjacently to each other, such that $\sum_{i=1}^l \mu_{C^i}[\Delta y(k+1)] = 1$. Therefore, at any sampling time, a crisp input can fire maximum two fuzzy sets. Suppose the desired output $\Delta y_d(k+1)$ fires two adjacent fuzzy sets C^j and C^{j+1} in the rules IR^j and IR^{j+1} . The singleton output MFs \bar{d}_r^j and \bar{d}_r^{j+1} is determined in the following four cases:

- When $\mu_{\mathbf{X}^j}(\mathbf{x}) > \mu_{C^j}(\Delta y_d)$ AND $\mu_{\mathbf{X}^{j+1}}(\mathbf{x}) > \mu_{C^{j+1}}(\Delta y_d)$

$$\Rightarrow \bar{d}_r^j = d_r^j \text{ AND } \bar{d}_r^{j+1} = d_r^{j+1}$$

- When $\mu_{\mathbf{X}^j}(\mathbf{x}) > \mu_{C^j}(\Delta y_d)$ and $\mu_{\mathbf{X}^{j+1}}(\mathbf{x}) < \mu_{C^{j+1}}(\Delta y_d)$

$$\Rightarrow \bar{d}_r^j = d_r^j \frac{\mu_{\mathbf{X}^{j+1}}(\mathbf{x})}{\mu_{C^{j+1}}(\Delta y_d)} \text{ AND } \bar{d}_r^{j+1} = d_r^{j+1} \frac{\mu_{C^{j+1}}(\Delta y_d) - \mu_{C^j}(\Delta y_d) \mu_{\mathbf{X}^{j+1}}(\mathbf{x})}{[\mu_{C^{j+1}}(\Delta y_d)]^2}$$

- When $\mu_{\mathbf{X}^j}(\mathbf{x}) < \mu_{C^j}(\Delta y_d)$ and $\mu_{\mathbf{X}^{j+1}}(\mathbf{x}) > \mu_{C^{j+1}}(\Delta y_d)$

$$\Rightarrow \bar{d}_r^j = d_r^j \frac{1}{\mu_{C^j}(\Delta y_d)} \text{ AND } \bar{d}_r^{j+1} = (d_r^{j+1} - d_r^j) \frac{\mu_{\mathbf{X}^j}(\mathbf{x})}{\mu_{C^j}(\Delta y_d)};$$

- When $\mu_{\mathbf{X}^j}(\mathbf{x}) < \mu_{C^j}(\Delta y_d)$ and $\mu_{\mathbf{X}^{j+1}}(\mathbf{x}) < \mu_{C^{j+1}}(\Delta y_d)$:

$$\Rightarrow \text{undetermined}$$

From the derivation procedure described in Chapter 4, the fuzzy inverse model for the changes of the system inputs $[\Delta u_1(k), \dots, \Delta u_p(k)]^T$ can be automatically constructed based on the fuzzy forward model. In other words, once the desired change in the system output $\Delta y_d(k+1)$ is given and the current input and output variables are measurable, the required changes of the system inputs $[\Delta u_1(k), \dots, \Delta u_p(k)]^T$ can be obtained simultaneously by the fuzzy inferencing calculation. These values will be used to initialize the parameters of the MLFC-MISO controller, which is explained as follows.

In most control problems, the desired system trajectory is specified beforehand. In order to move the current system output $y(k)$ to the desired value y_d at the next sampling time, it is desirable that the change of the system output at the next sampling time be equal to the difference of the desired system output from the current value, that is, $\Delta y_d(k+1) = y_d - y(k)$. Feeding this value into the fuzzy inverse model, the required changes of the system inputs $[\Delta u_1(k), \dots, \Delta u_p(k)]^T$ will be obtained by the fuzzy inferencing calculation. Suppose at current operating time, the system output is y , in order to obtain the desired output y_d at the next sampling time, the fuzzy inverse model is formulated and the required changes of the system input vector $[\Delta u_1, \dots, \Delta u_p]^T$ is obtained. Define the sensitivity factor λ_r as the ratio of the change of the system input Δu_r over the change of the system output $\Delta y_d = y_d - y$ as

$$\lambda_r = \frac{\Delta u_r}{\Delta y_d} = \frac{\Delta u_r}{y_d - y}, \quad r = 1, 2, \dots, p \quad (10.14)$$

The sign of this ratio λ_r represents the increase/decrease relationship of each system input set to the system output at the current operating time, which is used to initialize the fuzzy control rule tables. As described above, if the sensitivity factor λ_r is a negative number, which means an increase in the system input u_r results in a decrease in the system output y , the fuzzy control rule base will be constructed as in Table 9.1. On the other hand, if the sensitivity factor λ_r is a positive number, which means an increase in the system input u_r makes the system output y increase, the fuzzy control rule base will be established as the counterpart as in Table 9.1, where each entry in the table will have its negative value.

The absolute value of the sensitivity factor λ_r is the required change of the system input Δu_r to a unit change of the system output Δy_d . This value will be used as the scaling factor to adjust the importance degree for each control action u_r of the MLFC-MISO controller, which is explained in the next section. ■

10.1.2 FUZZY ADAPTIVE PD-PI CONTROLLER

Conventional PID (proportional-integral-derivative) controllers can be easily designed and the implementation cost is low. However, they are mostly effective for linear systems and difficult to achieve the desired performance in the entire operation range for complex nonlinear systems. Thus, fuzzy PID controllers have been

developed in recent years (Mizumoto, 1992; Chen, 1996; Mann and Hu, 1999; Farinwata et al., 2000). In the current MLFC-MISO controller design, each subcontroller is designed to be a fuzzy PID controller. The inputs to the controllers are the error $\tilde{e}(k)$, the change of the error $\tilde{c}(k)$, and the integral of the error $\tilde{b}(k)$, which are scaled by multiplying suitable scaling factors into their normalized regions as

$$\tilde{e}(k) = e(k) \cdot GE = [y_d(k) - y(k)] \cdot GE \quad (10.15)$$

$$\tilde{c}(k) = c(k) \cdot GC = [e(k) - e(k-1)] \cdot GC \quad (10.16)$$

$$\tilde{b}(k) = b(k) \cdot GI = \sum_{i=1}^k e(i) \cdot GI \quad (10.17)$$

where

$y_d(k)$ is the desired output at the time instant k

$y(k)$ is the actual output

$e(k)$ is the actual error

$c(k)$ is the actual change of the error

$b(k)$ is the actual integral of the error

GE is the scaling factor of the error

GC is the scaling factor of the change of error

GI is the scaling factor of the integral of the error

The PID fuzzy control rules are in the form of

$$R: \text{IF } e_1 \text{ is } E_k \text{ AND } e_2 \text{ is } E_l \text{ AND } e_3 \text{ is } E_q, \text{ THEN } u_r \text{ is } U_{rn(k,l,q)} \quad (10.18)$$

where

e_1 is the error $\tilde{e}(k)$

e_2 is the change of the error $\tilde{c}(k)$

e_3 is the integral of the error $\tilde{b}(k)$

u_r is the control action $\tilde{u}_r(k)$

E_k , E_l , and E_q are the linguistic variables of e_1 , e_2 , and e_3 , respectively

$U_{rn(k,l,q)}$ is the linguistic variable of u_r with $n(k, l, q)$ representing any function whose value at k , l , and q are integers (Ying, 1994; Calcev, 1998; Aracil and Gordillo, 2000; Farinwata et al., 2000)

The defuzzified control action $u_r(k)$ is calculated based on the three inputs using the fuzzy inferencing mechanism

$$u_r(k) = f_r(e(k), c(k), b(k)) \quad (10.19)$$

where $f_r(\cdot)$ is a nonlinear input-output mapping determined by the fuzzy control rules. The output of the fuzzy PID controller is constructed as shown in [Figure 10.2](#).

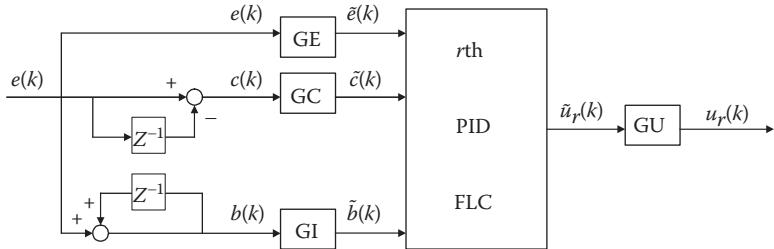


FIGURE 10.2 *rth* Fuzzy PID controller.

In Figure 10.2, in order to obtain the fuzzy relation R for the fuzzy PID controller, the fuzzy relations are aggregated as

$$R(e_1, e_2, e_3; u_r) = \bigcup_{i=1}^l \left\{ e_1^i \cap e_2^i \cap e_3^i \cap u_r^i \right\} \quad (10.20)$$

where

$$i = 1, 2, \dots, l$$

l is the total number of fuzzy rules

\bigcup is the union operator

\cap is the intersection operator

Suppose E_1 , E_2 , and E_3 are the fuzzy input spaces of e_1 , e_2 , and e_3 , respectively, and U_r is the fuzzy output space of u_r . Given the current fuzzy inputs e'_1 , e'_2 , and e'_3 , the new fuzzy output u'_r is obtained by the compositional rule of inferencing:

$$u'_r = e'_1 \circ e'_2 \circ e'_3 \circ R(e_1, e_2, e_3; u_r) \quad (10.21)$$

where \circ is the composition operator of fuzzy relations. Suppose the number of MFs assigned for each input variable is N , and then a total of N^3 fuzzy rules will be required to cover $N \times N \times N$ possible combinations of the input fuzzy sets. The rule base with three inputs becomes rather large as N increases. Because of the multidimensionality of the fuzzy relation R , the compositional fuzzy inferencing is difficult to perform and the computational time will be too long for real-time implementation. To overcome these difficulties, it is proposed to break up the inferencing of the multidimensional rule base into parts, so that the fuzzy inferencing is easier to perform. The fuzzy PID structure is analyzed and a fuzzy PD-PI controller is formed by the following fuzzy derivation:

$$\begin{aligned}
u'_r &= (e'_1, e'_2, e'_3) \circ \bigcup_{i=1}^l R(e_1^i, e_2^i, e_3^i; u_r^i) \\
&= (e'_1 \cap e'_2 \cap e'_3) \circ \bigcup_{E_1, E_2, E_3, U_r} (R_1, \dots, R_l) \\
&= \bigcup_{E_1, E_2, E_3} \left\{ (e'_1 \cap e'_2 \cap e'_3) \cap \left[\bigcup_{E_1, E_2, E_3, U_r} (R_1, \dots, R_l) \right] \right\} \\
&= \bigcup_{E_1, E_2, E_3, U_r} \bigcup_{E_1, E_2, E_3} [(e'_1 \cap e'_2 \cap e'_3) \cap R_1], \dots, [(e'_1 \cap e'_2 \cap e'_3) \cap R_l] \\
&= \bigcup_{E_1, E_2, E_3, U_r} \{[(e'_1, e'_2, e'_3) \circ R_1], \dots, [(e'_1, e'_2, e'_3) \circ R_l]\} \\
&= \bigcup_{E_1, E_2, E_3, U_r} (e'_1, e'_2, e'_3) \circ R(e_1^i, e_2^i, e_3^i; u_r^i) \\
&= \bigcup_{E_1, E_2, E_3, U_r} \bigcup_{E_1, E_2, E_3} [(e'_1 \cap e'_2 \cap e'_3) \cap (e_1^i \cap e_2^i \cap e_3^i \cap u_r^i)] \\
&= \bigcup_{E_1, E_2, E_3, U_r} \bigcup_{E_1, E_2, E_3} [(e'_1 \cap e'_1 \cap e'_2 \cap e'_3) \cap (e_1^i \cap e_1^i \cap e_2^i \cap e_3^i \cap u_r^i)] \\
&= \bigcup_{E_1, E_2, E_3, U_r} \bigcup_{E_1, E_2} \left([(e'_1 \cap e'_2) \cap (e_1^i \cap e_2^i) \cap u_r^i] \cap \left\{ \bigcup_{E_1, E_3} [(e'_1 \cap e'_3) \cap (e_1^i \cap e_3^i) \cap u_r^i] \right\} \right) \\
&= \bigcup_{E_1, E_2, E_3, U_r} \bigcup_{E_1, E_2} \left\{ [(e'_1 \cap e'_2) \cap (e_1^i \cap e_2^i) \cap u_r^i] \cap [(e'_1 \cap e'_3) \circ R(E_1^i, e_3^i; u_r^i)] \right\} \\
&= \bigcup_{E_1, E_2, E_3, U_r} \left\{ [(e'_1 \cap e'_2) \circ R(e_1^i, e_2^i; u_r^i)] \cap [(e'_1 \cap e'_3) \circ R(e_1^i, e_3^i; u_r^i)] \right\} \\
&= \left\{ \bigcup_{E_1, E_2, U_r} [(e'_1, e'_2) \circ R(e_1^i, e_2^i; u_r^i)] \right\} \cap \left\{ \bigcup_{E_1, E_3, U_r} [(e'_1, e'_3) \circ R(e_1^i, e_3^i; u_r^i)] \right\} \quad (10.22)
\end{aligned}$$

The resultant fuzzy PD-PI controller is constructed as shown in Figure 10.3, which is composed of two fuzzy rule bases. One is for the *P* (proportional) and *D* (derivative) inputs, and the other is for the *P* (proportional) and *I* (integral) inputs. The min operator in Figure 10.3 is defined in fuzzy domain, which represents the fuzzy intersection operation. The fuzzy outputs of the two fuzzy rule bases are compared, and their intersection is defuzzified and applied to the controlled system.

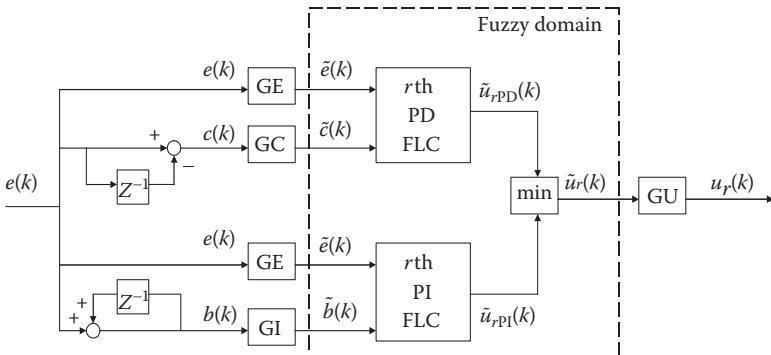


FIGURE 10.3 *rth* Fuzzy PD-PI controller.

In order to deal with the intrinsic uncertainties and time-varying parameters in complex nonlinear plants, each subfuzzy controller is designed to have a hierarchical structure with the upper level modifying the lower level fuzzy controller to achieve the desired system output. The detailed design method is explained in Xu and Shin (2005). The fuzzy rules in both levels are generated based on the fuzzy inverse model of the systems.

The first layer fuzzy PD control rules are in the form of

$$R: \text{IF } e_1 \text{ is } E_k \text{ AND } e_2 \text{ is } E_l, \text{ THEN } u_{r1} \text{ is } U_{rm(k,l)} \quad (10.23)$$

where

u_{r1} is the control action $\tilde{u}_{r\text{PD}}(k)$

$U_{rm(k,l)}$ is the linguistic variable of u_{r1}

The fuzzy control law of the first layer fuzzy PD controller is shown as

$$\tilde{u}_{r\text{PD}} = \frac{\sum_{k,l} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)] \cdot U_{rm(k,l)} \}}{\sum_{k,l} [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)]} \quad (10.24)$$

with

$$\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2) = \min[\mu_{E_k}(e_1), \mu_{E_l}(e_2)] \quad (10.25)$$

In order to compensate for the model uncertainties and the time-varying parameters, a self-organizing fuzzy PD controller is designed and embedded as the second layer fuzzy PD controller. The corresponding fuzzy control rules are

$$R: \text{IF } e_1 \text{ is } E_k \text{ AND } e_2 \text{ is } E_l, \text{ THEN } C_r \text{ is } C_{rm(k,l)} \quad (10.26)$$

where

C_r is the change of the center of the output MFs of the first layer fuzzy PD controller

$C_{rm(k,l)}$ is the linguistic variable of C_r

After the fuzzy rule adaptation, the centers of output MFs of the first layer fuzzy PD controller should be tuned as

$$U_{rm(k,l)} \rightarrow U_{rm(k,l)} + \frac{\sum_{k,l} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)] \cdot C_{rm(k,l)} \}}{\sum_{k,l} [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)]} \quad (10.27)$$

Similarly, the fuzzy rules of the PI controller are in the form of

$$R: \text{IF } e_1 \text{ is } E_k \text{ AND } e_3 \text{ is } E_q, \text{ THEN } u_{r2} \text{ is } U_{rn(k,q)} \quad (10.28)$$

where

- u_{r2} is the control action $\tilde{u}_{r\text{PI}}(k)$
- $U_{rn(k,q)}$ is the linguistic variable of u_{r2}

The fuzzy PI controller shares the same structure as the fuzzy PD controller, which has a hierarchical structure of two levels. The fuzzy control law of the first layer fuzzy PI controller is

$$\tilde{u}_{r\text{PI}} = \frac{\sum_{k,q} \left\{ \left[\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3) \right] \cdot U_{rn(k,q)} \right\}}{\sum_{k,q} \left[\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3) \right]} \quad (10.29)$$

And the centers of output MFs of the first layer fuzzy PI controller are tuned as

$$U_{rn(k,q)} \rightarrow U_{rn(k,q)} + \frac{\sum_{k,q} \left\{ \left[\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3) \right] \cdot C_{rm(k,q)} \right\}}{\sum_{k,q} \left[\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3) \right]} \quad (10.30)$$

With this adaptation mechanism, each adaptive fuzzy controller has the ability to tune the output MFs of the first layer fuzzy controller based on the system performance, in order to correct the imprecision of the first layer fuzzy control rules and compensate for the time-varying parameters.

As stated in Section 10.1.1, the absolute value of the sensitivity factor λ_r represents the required change of the system input Δu_r to a unit change of the system output Δy_d . It determines the importance degree of each control action in obtaining the desired change of the system output. Thus, the r th control action $u_r(k)$ from the r th sub-FLC needs to be multiplied by the absolute value of the sensitivity factor λ_r to obtain the final control action as $|\lambda_r| \cdot u_r(k)$. The defuzzified output $u_r(k)$ of the fuzzy PD–PI controller is

$$\begin{aligned} u_r(k) &= \tilde{u}_r(k) \cdot \text{GU} \\ &= \min(\tilde{u}_{r\text{PD}}(k), \tilde{u}_{r\text{PI}}(k)) \cdot \text{GU} \\ &= \min(\tilde{u}_{r\text{PD}}(k) \cdot \text{GU}, \tilde{u}_{r\text{PI}}(k) \cdot \text{GU}) \\ &= \min(u_{r\text{PD}}(k), u_{r\text{PI}}(k)) \end{aligned} \quad (10.31)$$

The p control actions can be written in vector form as $|\boldsymbol{\lambda}| \cdot \mathbf{u}(k)$, where $|\boldsymbol{\lambda}| = [|\lambda|_1, \dots, |\lambda|_p]$, $\mathbf{u}(k) = \begin{bmatrix} u_1(k) \\ \vdots \\ u_p(k) \end{bmatrix} = \begin{bmatrix} \min(u_{1\text{PD}}(k), u_{1\text{PI}}(k)) \\ \vdots \\ \min(u_{p\text{PD}}(k), u_{p\text{PI}}(k)) \end{bmatrix}$. The block diagram of each adaptive subfuzzy controller is shown in [Figure 10.4](#).

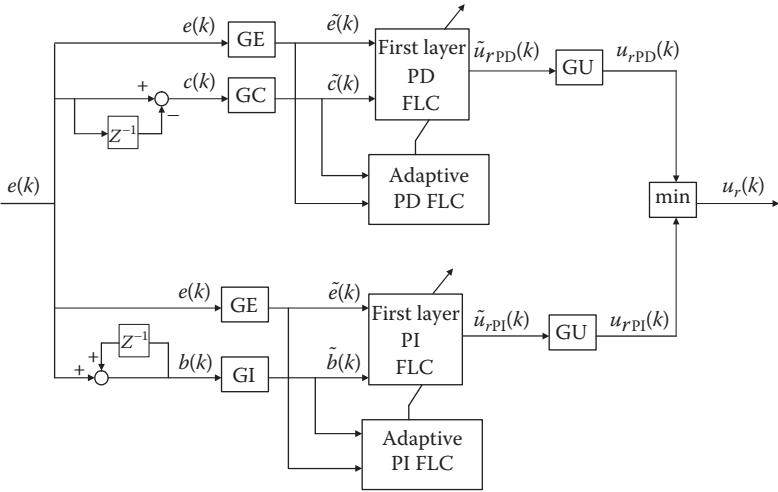


FIGURE 10.4 *r*th Adaptive fuzzy PD-PI controller.

Note that since all the variables are normalized within the consideration range, in the defuzzification step, the defuzzified control action will not exceed the operating range. In the implementation of control, if one control action hits the boundary of constraint, that is, saturation, it will stay right at the boundary point and the other variables are used to control the system.

10.2 STABILITY ANALYSIS

From Equation 10.31, the crisp output of the *r*th subfuzzy controller is

$$\begin{aligned}
 & |\lambda_r| \cdot u_r(k) \\
 &= |\lambda_r| \cdot \min(u_{rPD}(k), u_{rPI}(k)) \\
 &= |\lambda_r| \cdot \min(\tilde{u}_{rPD}(k) \cdot GU, \tilde{u}_{rPI}(k) \cdot GU) \\
 &= |\lambda_r| \cdot \min\left(\frac{\sum_{k,l} \{[\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)] \cdot U_{rn(k,l)}\}}{\sum_{k,l} [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)]} \cdot GU,\right. \\
 &\quad \left.\frac{\sum_{k,q} \{[\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)] \cdot U_{rn(k,q)}\}}{\sum_{k,q} [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)]} \cdot GU\right)
 \end{aligned}$$

By the adaptation mechanism,

$$\begin{aligned}
|\lambda_r| \cdot u_r(k) &= |\lambda_r| \cdot \min \left(\frac{\sum_{k,l} \left\{ [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)] \cdot \left(U_{rn(k,l)} + \frac{\sum_{k,l} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)] \cdot C_{rm(k,l)} \}}{\sum_{k,l} [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)]} \cdot GU \right) \right\}}{\sum_{k,l} [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)]} \cdot GU, \right. \\
&\quad \left. \sum_{k,q} \left\{ [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)] \cdot \left(U_{rn(k,q)} + \frac{\sum_{k,q} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)] \cdot C_{rm(k,q)} \}}{\sum_{k,q} [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)]} \cdot GU \right) \right\} \right. \\
&\quad \left. \sum_{k,q} [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)] \cdot GU \right) \\
&= |\lambda_r| \cdot \min \left(\frac{\sum_{k,l} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)] \cdot U_{rn(k,l)} \}}{\sum_{k,l} [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)]} \cdot GU + \frac{\sum_{k,l} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)] \cdot C_{rm(k,l)} \}}{\sum_{k,l} [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)]} \cdot GU^2, \right. \\
&\quad \left. \frac{\sum_{k,q} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)] \cdot U_{rn(k,q)} \}}{\sum_{k,q} [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)]} \cdot GU + \frac{\sum_{k,q} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)] \cdot C_{rm(k,q)} \}}{\sum_{k,q} [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)]} \cdot GU^2 \right) \\
&= \min(\Phi_{r1PD}(e_1, e_2) + \Phi_{r2PD}(e_1, e_2), \Phi_{r1PI}(e_1, e_3) + \Phi_{r2PI}(e_1, e_3)) \\
&= \min(\Phi_{rPD}(e_1, e_2), \Phi_{rPI}(e_1, e_3))
\end{aligned} \tag{10.32}$$

where

$$\Phi_{rPD}(e_1, e_2) = \Phi_{r1PD}(e_1, e_2) + \Phi_{r2PD}(e_1, e_2)$$

$$\Phi_{rPI}(e_1, e_3) = \Phi_{r1PI}(e_1, e_3) + \Phi_{r2PI}(e_1, e_3)$$

$$\Phi_{r1PD}(e_1, e_2) = |\lambda_r| \cdot \frac{\sum_{k,l} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)] \cdot U_{rn(k,l)} \}}{\sum_{k,l} [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)]} \cdot GU$$

$$\Phi_{r2PD}(e_1, e_2) = |\lambda_r| \cdot \frac{\sum_{k,l} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)] \cdot C_{rm(k,l)} \}}{\sum_{k,l} [\mu_{E_k}(e_1) \wedge \mu_{E_l}(e_2)]} \cdot GU^2$$

$$\Phi_{r1PI}(e_1, e_3) = |\lambda_r| \cdot \frac{\sum_{k,q} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)] \cdot U_{rn(k,q)} \}}{\sum_{k,q} [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)]} \cdot GU$$

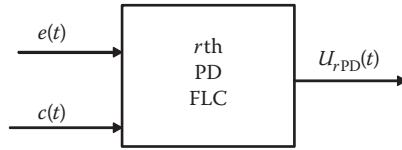
$$\Phi_{r2PI}(e_1, e_3) = |\lambda_r| \cdot \frac{\sum_{k,q} \{ [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)] \cdot C_{rm(k,q)} \}}{\sum_{k,q} [\mu_{E_k}(e_1) \wedge \mu_{E_q}(e_3)]} \cdot GU^2$$

It is observed that $\Phi_{r1PD}(e_1, e_2)$ represents the first layer fuzzy PD controller, while $\Phi_{r2PD}(e_1, e_2)$ is the self-organizing fuzzy PD controller. In continuous case of the MLFC-MISO controller design, e_1 represents the error $e(t)$, e_2 represents the change of the error $c(t)$, and $\Phi_{rPD}(e_1, e_2)$ is the r th PD control action. Thus, the r th PD fuzzy control system can be defined as

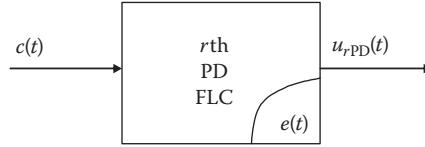
$$\dot{e}_1 = e_2 \quad (10.33)$$

$$\Phi_{rPD}(e_1, e_2) = \Phi_{r1PD}(e_1, e_2) + \Phi_{r2PD}(e_1, e_2) \quad (10.34)$$

As proved in Section 9.2, the r th PD fuzzy controller is input–output passive stable in continuous domain, whose inputs are the error $e(t)$ and the change of the error $c(t)$, the output is the r th PD fuzzy control action $u_{rPD}(t) = \Phi_{rPD}(e_1, e_2)$, as shown below:



As presented in Section 9.2, if the error signal $e(t)$ is defined as the state variable, the r th PD fuzzy controller can be further presented as an equivalent expression:



The definition of passivity results in

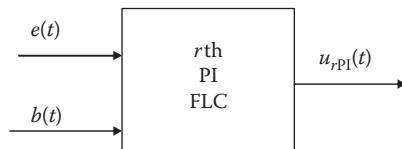
$$\int_0^t c(\tau) \cdot u_{rPD}(\tau) d\tau \geq V[e(t)] - V[e(0)] \quad (10.35)$$

Similarly, $\Phi_{r1PI}(e_1, e_3)$ and $\Phi_{r2PI}(e_1, e_3)$ represent the corresponding first layer and the adaptive layer of the fuzzy PI controller, respectively. In continuous domain, e_1 represents the error $e(t)$, e_3 represents the integral of the error $b(t)$, and $\Phi_{rPI}(e_1, e_3)$ is the r th PI control action. Thus, the r th PI fuzzy control system can be defined as

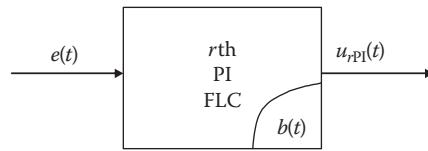
$$\dot{e}_3 = e_1 \quad (10.36)$$

$$\Phi_{rPI}(e_1, e_3) = \Phi_{r1PI}(e_1, e_3) + \Phi_{r2PI}(e_1, e_3) \quad (10.37)$$

And then the r th PI fuzzy controller is input–output passive stable in continuous domain, whose inputs are the error $e(t)$ and the integral of the error $b(t)$, the output is the r th PI fuzzy control action $u_{rPI}(t) = \Phi_{rPI}(e_1, e_3)$, as shown below:



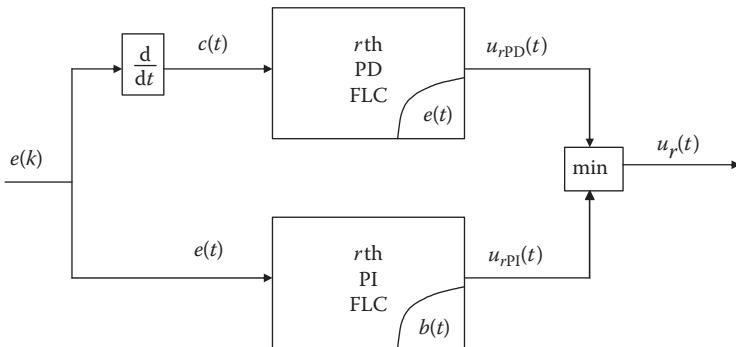
If the integral of the error $b(t)$ is defined as the state variable, the r th PI fuzzy controller can be further presented as an equivalent expression:



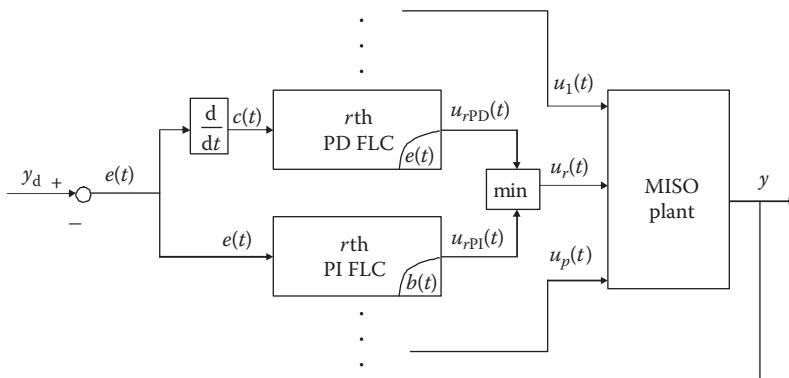
The definition of passivity results in

$$\int_0^t e(\tau) \cdot u_{r\text{PI}}(\tau) d\tau \geq V[b(t)] - V[b(0)] \quad (10.38)$$

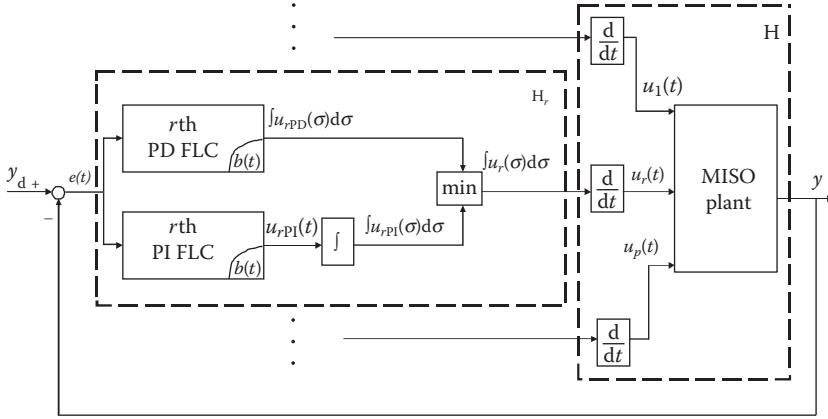
And then the r th adaptive fuzzy PD–PI controller in Figure 10.4 can be simplified as the figure below and is passive stable:



The closed-loop fuzzy control system for this MISO plant is shown below:



By exchanging the two functional blocks of $\frac{d}{dt}$ and the r th PD FLC, adding two additional functional blocks of \int and $\frac{d}{dt}$ for the r th PI FLC, the closed-loop fuzzy control system can be further transformed into the figure below:



Furthermore, specify H_r as the r th subcontroller and the combination of $\frac{d}{dt}$ and the open-loop plant as the augmented plant H . From the derivation in [Section 10.2](#), if two passive subsystems H_r and H are interconnected in a well-defined negative feedback connection, the closed-loop system is still stable, where H_r represents the controller and H is the system to be controlled. Thus, the requirement of the closed-loop system stability is the input–output passivity of the augmented system. As long as the open-loop plant satisfies the relaxed-passivity condition (Equation 10.39), the resultant MLFC-MISO closed-loop system is input–output passive stable if

$$V[\mathbf{x}(t_f)] - V[\mathbf{x}(t_0)] \leq \int_{t_0}^{t_f} \left[\int u_r(\sigma) d\sigma \right]^T y(\sigma) d\sigma, \quad \forall \mathbf{x}, \forall u_r \quad (10.39)$$

for each $u_r - y$ input–output pair in continuous domain. The passivity of the plant can be checked by the inequality equation (Equation 10.40) using the fuzzy forward model (Equation 10.10):

$$V[\mathbf{x}(t_f)] - V[\mathbf{x}(t_0)] \int_{t_0}^{t_f} \left[\int u_r(\sigma) d\sigma \right]^T \cdot \frac{\sum_{i=1}^l c^i \cdot [\mu_{x^i}(\mathbf{x}) \wedge \mu_{D_1^i}(u_1) \wedge \dots \wedge \mu_{D_p^i}(u_p)]}{\sum_{i=1}^l [\mu_{x^i}(\mathbf{x}) \wedge \mu_{D_1^i}(u_1) \wedge \dots \wedge \mu_{D_p^i}(u_p)]} d\sigma, \quad \forall \mathbf{x}, \forall u_r \quad (10.40)$$

The stability of the MLFC-MISO fuzzy control closed-loop in discrete time domain can be proved by the same principle.

10.3 SIMULATION EXAMPLES

In this section, two sets of simulation examples are provided to illustrate the effectiveness of the proposed MLFC-MISO controller. The closed-loop performance is

compared with the conventional control techniques. Satisfactory results are obtained for the time-varying systems in presence of model uncertainties and disturbances.

10.3.1 MAGNETIC BEARING SYSTEM

In this example, the proposed MLFC-MISO controller is compared with a nonlinear controller for a magnetic bearing system, which is used in various industrial applications. The magnetic bearing system is a typical MISO nonlinear system, which uses magnetic forces to suspend a rotor shaft in the middle of air.

A schematic diagram of a single axis of the magnetic bearing system is shown in Figure 10.5. Each electromagnet comprises N turns of conducting coil around the highly permeable magnetic core, and the mass of the rotor is m . Deviation of the rotor from the centered position is denoted as x , which is referred as position variation. The nominal distance of the rotor away from the magnets is h . The coil currents are denoted as i_1 and i_2 , and the input voltages at the coil terminals are e_1 and e_2 .

The magnetic bearing system is modeled as a fourth-order two-input nonlinear system, which can be described in the following state-space form:

$$\begin{aligned}\dot{y}_1 &= y_2 \\ \dot{y}_2 &= -\frac{k}{2m} \left[\frac{i_1^2}{(h+y_1)^2} + \frac{i_2^2}{(h-y_1)^2} \right] \\ \dot{i}_1 &= -\frac{R}{k}(h+y_1)i_1 + (h+y_1)^{-1}y_2i_1 + \frac{1}{k}(h+y_1)e_1 \\ \dot{i}_2 &= -\frac{R}{k}(h-y_1)i_2 - (h-y_1)^{-1}y_2i_2 + \frac{1}{k}(h-y_1)e_2\end{aligned}\quad (10.41)$$

where the four state variables are $\mathbf{x} = [x_1, x_2, x_3, x_4] = [y_1, y_2, i_1, i_2]$ and y_1 is the position variation. The inputs to the system are the two winding voltages e_1 and e_2 , that is, $\mathbf{u} = [u_1, u_2] = [e_1, e_2]$. The process output y is a function of the state variables:

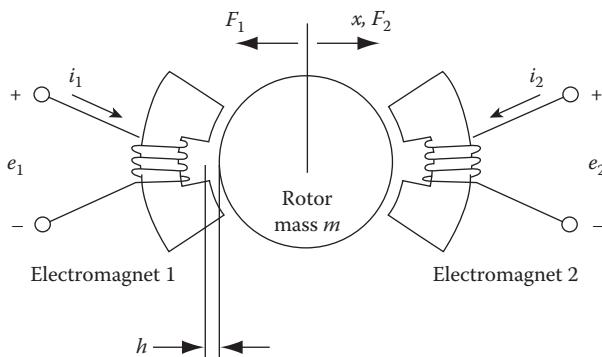


FIGURE 10.5 Schematic diagram for the magnetic bearing system.

$$y = [1 \ 0 \ 0 \ 0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = [1 \ 0 \ 0 \ 0] \begin{bmatrix} y_1 \\ y_2 \\ i_1 \\ i_2 \end{bmatrix} = y_1 \quad (10.42)$$

In this magnetic bearing system, the magnetic flux linkage $\lambda(i, y)$ in each magnet is expressed as

$$\begin{aligned} \lambda_1(i_1, y) &= k \left(\frac{i_1}{h+y} \right), \quad k = \text{constant} \\ \lambda_2(i_2, y) &= k \left(\frac{i_2}{h-y} \right), \quad k = \text{constant} \end{aligned} \quad (10.43)$$

Each of the magnetic bearing has the following nonlinear differential equation to describe the electromagnetic dynamics:

$$\begin{aligned} e_1 &= Ri_1 + \frac{d}{dt} \lambda_1(i_1, y) \\ e_2 &= Ri_2 + \frac{d}{dt} \lambda_2(i_2, y) \end{aligned} \quad (10.44)$$

where R is winding resistance. For input e_1 ,

$$\begin{aligned} &\left[\int e_1(\sigma) d\sigma \right] \cdot y(\tau) \\ &= \left\{ \int \left[Ri_1(\sigma) + \frac{d}{d\sigma} \lambda_1(i_1(\sigma), y(\sigma)) \right] d\sigma \right\} \cdot y(\tau) \\ &= Ry(\tau) \cdot \left[\int i_1(\sigma) d\sigma \right] + y(\tau) \cdot \lambda_1(i_1(\tau), y(\tau)) \\ &= Ry(\tau) \cdot \left[\int i_1(\sigma) d\sigma \right] + \frac{ky(\tau) \cdot i_1(\tau)}{h+y(\tau)} \end{aligned} \quad (10.45)$$

Furthermore,

$$\begin{aligned} &\int_0^t \left[\int e_1(\sigma) d\sigma \right]^T y(\tau) d\tau \\ &= \int_0^t \left\{ Ry(\tau) \cdot \left[\int i_1(\sigma) d\sigma \right] + \frac{ky(\tau) \cdot i_1(\tau)}{h+y(\tau)} \right\} d\tau \\ &= \int_0^t Ry(\tau) \cdot \left[\int i_1(\sigma) d\sigma \right] d\tau + \int_0^t \frac{ky(\tau) \cdot i_1(\tau)}{h+y(\tau)} d\tau \end{aligned} \quad (10.46)$$

Define the storage function as

$$\begin{aligned}
 V[\mathbf{x}(t)] &= \int_0^t Rx_1(\tau) \cdot \left[\int x_3(\sigma) d\sigma \right] d\tau + \int_0^t \frac{kx_1(\tau) \cdot x_3(\tau)}{h + x_1(\tau)} d\tau \\
 &= \int_0^t Ry(\tau) \cdot \left[\int i_1(\sigma) d\sigma \right] d\tau + \int_0^t \frac{ky(\tau) \cdot i_1(\tau)}{h + y(\tau)} d\tau
 \end{aligned} \tag{10.47}$$

with $V(\mathbf{x}(0)) = 0$,

$$\int_0^t \left[\int e_1(\sigma) d\sigma \right]^T y(\tau) d\tau \geq V[\mathbf{x}(t)] - V[\mathbf{x}(0)] \tag{10.48}$$

Similarly, for input e_2 ,

$$\int_0^t \left[\int e_2(\sigma) d\sigma \right]^T y(\tau) d\tau \geq V[\mathbf{x}(t)] - V[\mathbf{x}(0)] \tag{10.49}$$

Thus, even though the magnetic bearing system is an unstable system, its augmentation is input–output passive. Next, three simulation results will illustrate how this unstable multivariable system is stabilized by the multilevel hierarchical fuzzy controller in the presence of unknown disturbance or model variation. The physical dimensions and model parameters are given in Table 10.1.

In order to model this MISO nonlinear system, 1000 input–output training data pairs were generated from the mathematical equations. The input vector to the fuzzy basis function network (FBFN) at the time instant k is defined as $[x_1(k), x_2(k), x_3(k), x_4(k), u_1(k), u_2(k), \Delta u_1(k), \Delta u_2(k)]$. The output of the FBFN is the change of process output $\Delta y(k+1)$. Using the adaptive least-square training algorithm, the FBFN was obtained, from which the inverse fuzzy model of the system was constructed systematically. The input vector to the inverse fuzzy model is $[x_1(k), x_2(k), x_3(k), x_4(k), u_1(k), u_2(k), \tilde{\Delta y}(k+1)]$ and the outputs are $\Delta u_1(k+1)$ and $\Delta u_2(k+1)$, respectively.

TABLE 10.1
Model Parameters for the Magnetic Bearing System

m	Rotor mass	125	g
h	Norminal air gap	1.6	mm
R	Winding resistance	4.75	Ω
k	Magnetic flux parameter	2.788×10^{-6}	$N \text{ m}^2/\text{A}^2$
	Nominal winding inductance	7.5	mH

Suppose the desired process output is y_d . The ideal case is that the process output should be changed from the current value $y(k)$ to the desired value y_d at the next time instant so that $\Delta\tilde{y}(k+1) = y_d - y(k)$. When the input vector is fed into the inverse fuzzy model, the changes of the control actions $[\Delta u_1, \Delta u_2]$ can be obtained using the fuzzy inferencing calculation.

Denote the inputs to the MLFC-MISO controller as the error of the position deviation $\tilde{e}(k)$, the change of the error $\tilde{c}(k)$, and the integral of the error $\tilde{b}(k)$ expressed as

$$\begin{aligned}\tilde{e}(k) &= e(k) \cdot GE = [y_d(k) - y(k)] \cdot GE \\ \tilde{c}(k) &= c(k) \cdot GC = [e(k) - e(k-1)] \cdot GC \\ \tilde{b}(k) &= b(k) \cdot GI = \sum_{i=1}^k e(i) \cdot GI\end{aligned}\tag{10.50}$$

The outputs of the fuzzy controller are the two winding voltages e_1 and e_2 . The fuzzy MFs and the rule tables are constructed according to the inverse fuzzy model of the process. After the fuzzy inferencing calculation, the control actions are obtained as

$$u_1(k) = u_1(k-1) + |\lambda_1| \cdot \Delta u_1(k)\tag{10.51}$$

$$u_2(k) = u_2(k-1) + |\lambda_2| \cdot \Delta u_2(k)\tag{10.52}$$

In order to benchmark the control performance, a nonlinear controller is designed using a combination of input-state linearization and recursive algorithm (Hung et al., 2003). A multiple-loop feedback controller is constructed and the block diagram is shown in Figure 10.6. The dark lines represent vector signals, for example, two winding currents, or two winding voltages. The outermost loop is the linear feedback control that determines the pseudoinput u . The middle loop is the nonlinear feedback linearization of mechanical dynamics, which transforms the pseudoinput u to the reference currents $i_{1\text{ref}}$ and $i_{2\text{ref}}$. The innermost loop is the high-gain linear feedback control of electrical dynamics, which ensures that the actual currents i_1 and i_2 track the reference currents. The simulations are run under three sets of working conditions. The sampling time is 0.0001 s with both controllers. In all the three cases,

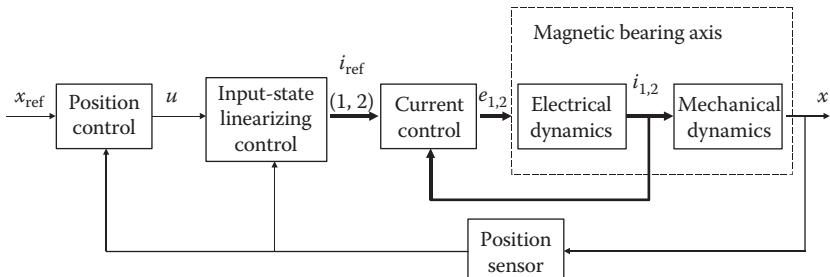


FIGURE 10.6 Block diagram of the nonlinear control for one axis.

better performances are achieved with the hierarchical fuzzy controller in stabilizing the system with shorter transient times and no output oscillation.

Example 10.1 Normal Condition

In this case, the magnetic bearing system is operated under the normal condition without any disturbance or parameter variation. The desired output is defined as a step function and equals 25% of the nominal air gap, which is shown as the solid line in [Figure 10.7a](#). When comparing the results of the two control approaches, both control systems can reach the stable line without any overshoot. The MLFC-MISO controller shows its superiority since it takes less than 0.004 s to stabilize the system to the reference position, while it takes about 0.007 s with the nonlinear controller. The reason that it takes longer time for the nonlinear controller to stabilize the system is because an assumption is made in the controller design such that only one winding current is nonzero at any time instant, and a judicious choice of the scaling gains needs to be made beforehand in the linear feedback control loops. [Figure 10.7b](#) shows the control actions with the MLFC-MISO controller, and [Figure 10.7c](#) shows the reference currents with the nonlinear controller. The control actions in the nonlinear control case are not shown in [Figure 10.7](#), since based on their assumption that only one winding current is nonzero at any instant of time, the calculated winding voltages e_1 and e_2 change signs for each sampling time, which is unable to clearly show in the figure and will be difficult to implement in real-world applications as well.

Example 10.2 With External Force Disturbance

In the actual operation, there always exists some kind of disturbance introduced into the system that affects the system performance. This simulation is designed to illustrate the ability of the MLFC-MISO controller to compensate for the unknown disturbance to the system. The desired output trajectory is the same as that in Example 10.1. Suppose $\delta(t)$ is a disturbance force between the two magnets, which is given as

$$\delta(t) = 10 * \text{rand} \quad (10.53)$$

Thus, the state equations will be modified as

$$\begin{aligned}\dot{y}_1 &= y_2 \\ \dot{y}_2 &= -\frac{k}{2m} \left[\frac{i_1^2}{(h+y_1)^2} + \frac{i_2^2}{(h-y_1)^2} \right] + \frac{\delta(t)}{m} \\ \dot{i}_1 &= -\frac{R}{k}(h+y_1)i_1 + (h+y_1)^{-1}y_2i_1 + \frac{1}{k}(h+y_1)e_1 \\ \dot{i}_2 &= -\frac{R}{k}(h-y_1)i_2 - (h-y_1)^{-1}y_2i_2 + \frac{1}{k}(h-y_1)e_2\end{aligned} \quad (10.54)$$

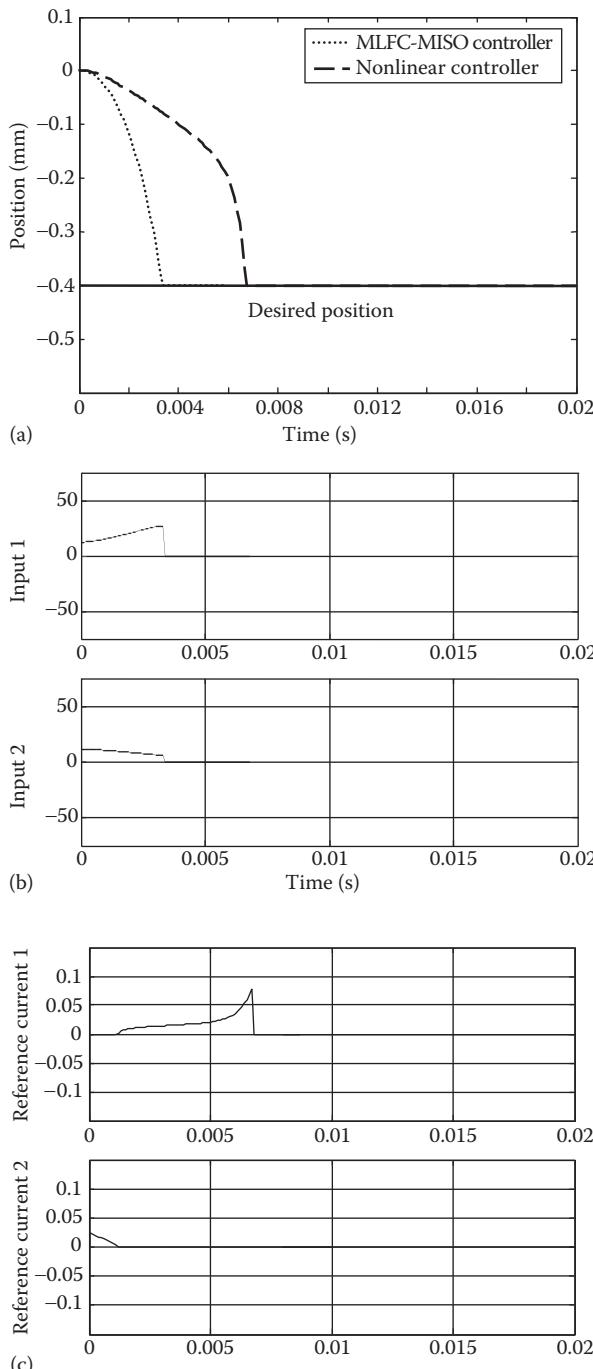
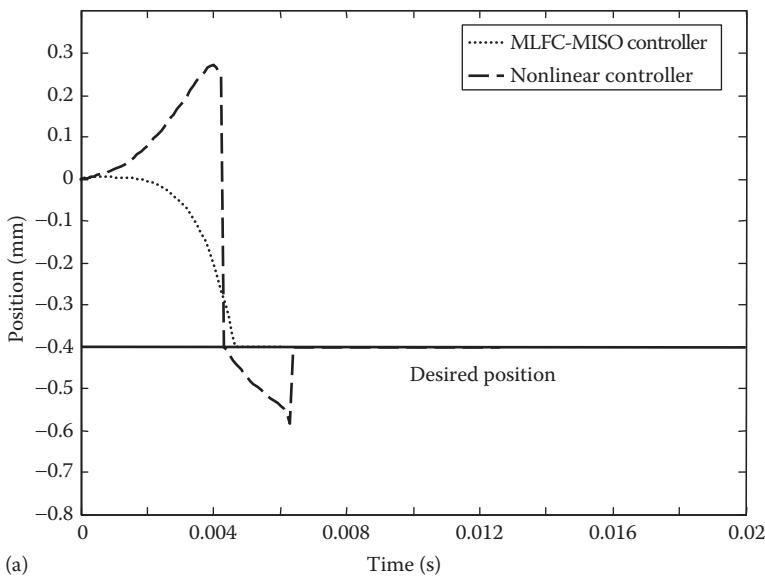
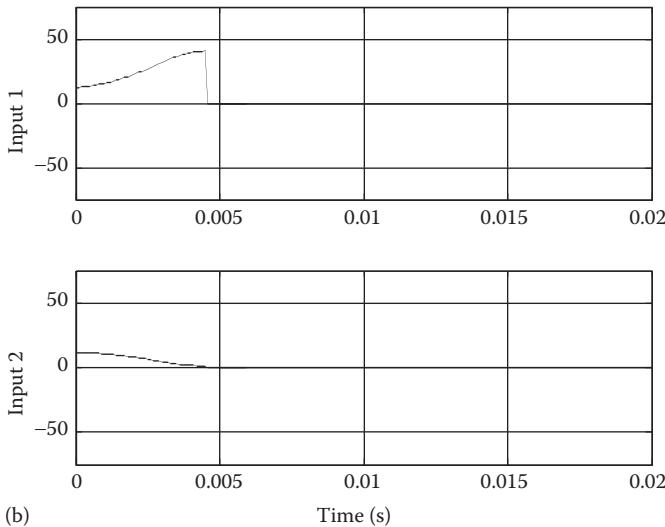


FIGURE 10.7 Simulation comparison under normal condition in Example 10.1: (a) position signals, (b) control actions with MLFC-MISO controller, and (c) reference currents with nonlinear controller.

Figure 10.8a illustrates the magnetic ball position obtained from this simulation, which shows that the MLFC-MISO is quite successful in generating a good system response since the rotor shaft is approaching the desired position without any overshoot. In contrast, there exists a large oscillation in the system output with the nonlinear controller. The magnetic ball approaches the desired position and then overshoots to the inverse direction before stabilizing. This is because in their controller design, an assumption is made that only one winding current is nonzero



(a)



(b)

FIGURE 10.8 Simulation comparison with external force disturbance: (a) position signals, (b) control actions with MLFC-MISO controller, and

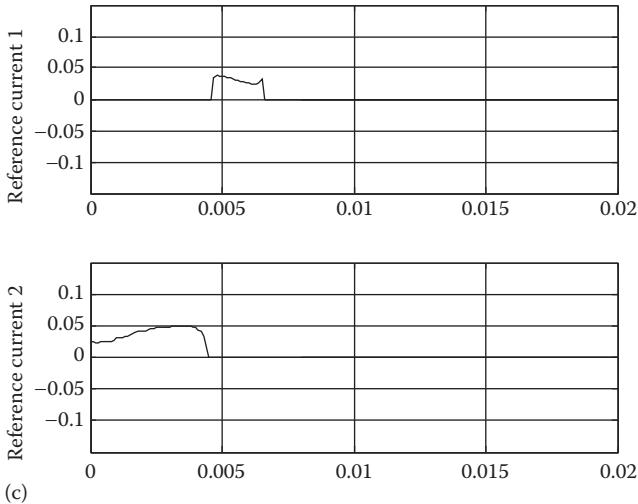


FIGURE 10.8 (continued) (c) reference currents with nonlinear controller.

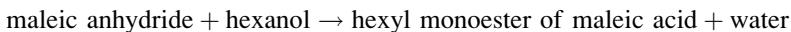
at any time instant, which tends to make the open-loop unstable system oscillate. The transient time in the case with nonlinear controller is longer than the case with MLFC-MISO controller as well.

Example 10.3 With Model Parameter Variation

The purpose of this simulation example is to demonstrate the adaptive capability of the MLFC-MISO with model parameter variation in the control process. Consider the mass of the rotor shaft is changed from 125 to 1000 g. From [Figure 10.9a](#), it is clear that the performance of the MLFC-MISO controller is much better than that of the nonlinear controller. Since the nonlinear controller is designed based on the model parameters under the normal condition, it cannot adapt to the model parameter variation and takes a much longer time to stabilize the system. And also as stated in the previous case, the system response has oscillation before finally stabilizing to the desired position.

10.3.2 FED-BATCH REACTOR

The controlled system considered here is the synthesis of the hexyl monoester of maleic acid, according to the following reaction:



Because of the large heat of reaction ($\Delta H = -33.5 \text{ MJ/kmol}$) in the fed-batch operation, maleic anhydride is first melted and hexanol is added at a regulated rate so that the heat generated is matched by cooling capacity. The state equations proposed in Westerterp et al. (1984) and adopted in Chang and Hsieh (1995) and Santos et al. (2000) are

$$\begin{aligned}
\dot{C}_a &= -\frac{C_a U_1}{V_r} - k_0 \exp(-12,628/T_r) C_a C_b \\
\dot{C}_b &= \frac{(C_{bl} - C_b) U_1}{V_r} - k_0 \exp(-12,628/T_r) C_a C_b \\
\dot{T}_r &= -\frac{\Delta H}{\rho C_p} k_0 \exp(-12,628/T_r) C_a C_b - (T_r - 327) \frac{U_1}{V_r} - U_2(T_r - 327) \\
\dot{V}_r &= U_1
\end{aligned} \tag{10.55}$$

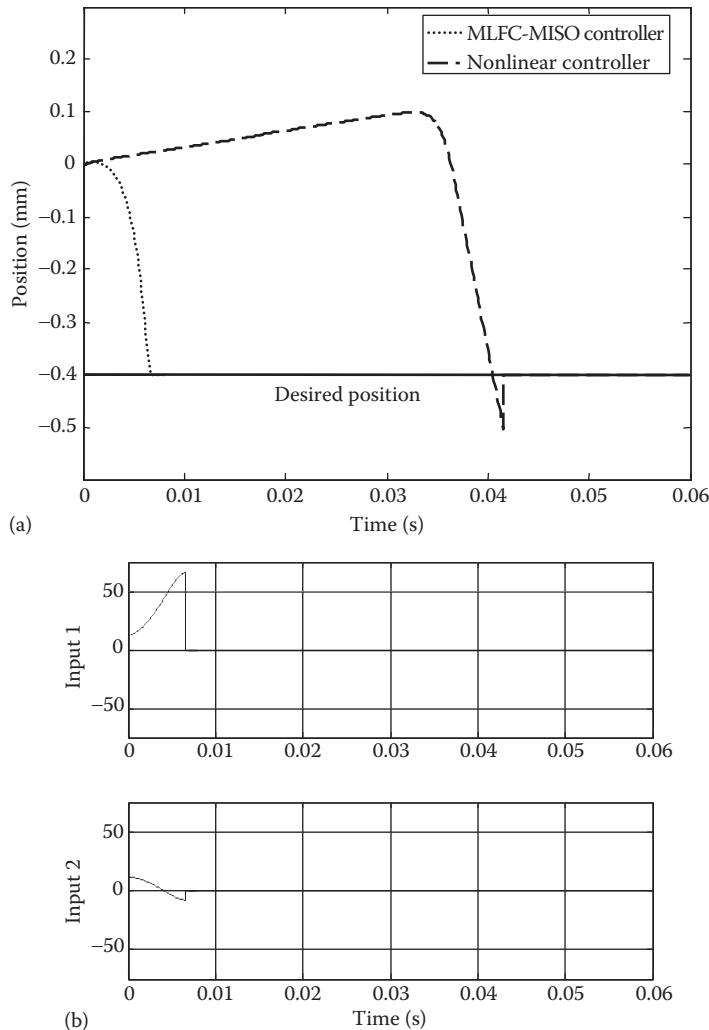


FIGURE 10.9 Simulation comparison with model parameter variation: (a) position signals, (b) control actions with MLFC-MISO controller, and

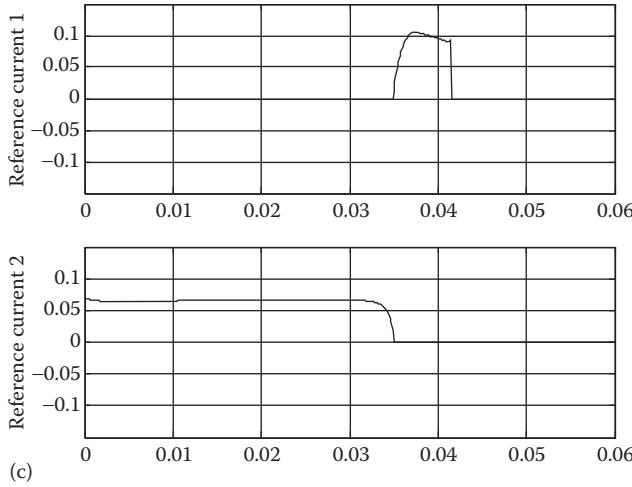


FIGURE 10.9 (continued) (c) reference currents with nonlinear controller.

where the state variables are the concentration of maleic anhydride C_a , the concentration of hexanol C_b , reactor's temperature T_r , and reactor's volume V_r : $\mathbf{x} = [C_a, C_b, T_r, V_r]$. Two process inputs are the flow rate U_1 of hexanol and the coolant flow rate U_2 : $\mathbf{u} = [U_1, U_2]$. The initial conditions are defined as $U_1 = 0.008 \text{ m}^3/\text{s}$ and $U_2 = 0.000253 \text{ s}^{-1}$. The process output is a final conversion of hexanol, which is expressed as

$$y = 1 - \frac{V_r(t_f)C_a(t_f)}{V_r(t_0)C_a(t_0)} \quad (10.56)$$

where t_0 is the start time of the batch. The desired conversion percentage is 99%.

Some constraints are considered in the process as

$$\begin{aligned} U_1(t) &\in [0.0, 0.01] \\ U_2(t) &\in [0.0, 0.000253] \\ T_r &\leq 373 \text{ K} \\ V_r &\leq 6.7 \text{ m}^3 \end{aligned} \quad (10.57)$$

Certain parameters and initial conditions of the fed-batch reactor are listed in [Table 10.2](#).

It is widely known that model predictive control (MPC) is an effective way of controlling a nonlinear process with certain constraints. In Santos et al. (2000), the authors compare the control performance of a nonlinear MPC with a multiple linear MPC. The multiple linear MPC shows better control performance in the sense that it takes less time to finish the material conversion. Hence, in this example the multiple linear MPC is implemented to compare the control performance with the proposed MLFC-MISO controller. First, the nonlinear MISO process model is analyzed and decomposed into a composite model consisting of multiple operating regimes with local linear models in each regime. At each sampling time, the state variables are

TABLE 10.2
**Model Parameters and Initial
 Conditions for a Fed-Batch Reactor**

Parameter	Value
k_0	1.37×10^{12}
$-\Delta H/\rho C_p$	$16.92 \text{ m}^3 \cdot \text{K}/\text{kmol}$
C_{a0}	$10.0 \text{ kmol}/\text{m}^3$
C_{b0}	$0.0 \text{ kmol}/\text{m}^3$
C_{bl}	$9.7 \text{ kmol}/\text{m}^3$
T_0	328 K
V_{r0}	2.2 m^3
V_{rf}	6.7 m^3
$U_{1,\min}$	$0 \text{ m}^3/\text{s}$
$U_{1,\max}$	$0.01 \text{ m}^3/\text{s}$
$U_{2,\min}$	0 s^{-1}
$U_{2,\max}$	0.000253 s^{-1}

measured and the quasi-Newton method is used to find the future control actions by minimizing the error of the process output. The optimum inputs are applied to the process at the next sampling time, and the same procedure is repeated next. In this simulation example, the sampling time is 100 s with both controllers.

Example 10.4 Normal Condition

In this case, the fed-batch reactor is operated in normal condition. The desired conversion percentage is 99%, which is shown as solid line in [Figure 10.10a](#). The actual conversion percentage with the MLFC-MISO controller and the MPC are shown as dotted line and dashed line, respectively. By comparison, both control systems can generate the required control actions for this complex multivariable system to achieve the desired conversion percentage. The MLFC-MISO takes 500 s to finish the conversion, while the MPC takes longer time as 2000 s because the controller design is based on multiple local linear models, which have certain bias as the original non-linear process model. [Figure 10.10b](#) shows the control actions with the MLFC-MISO controller, and [Figure 10.10c](#) shows the control actions with the MPC.

Example 10.5 With Time-Varying Parameter

This simulation is to show the effectiveness of the MLFC-MISO controller in compensating for the time-varying parameter. The desired conversion percentage is the same as that in Example 10.4. Under the normal condition, hexanol has a feed flow C_{bl} of $9.7 \text{ kmol}/\text{m}^3$. In this case, the feed flow is changed linearly from 9.7 to $2 \text{ kmol}/\text{m}^3$ in the first 1500 s as shown in [Figure 10.11](#). From the results shown in [Figure 10.12](#), it is clear that the MLFC-MISO controller takes less time to

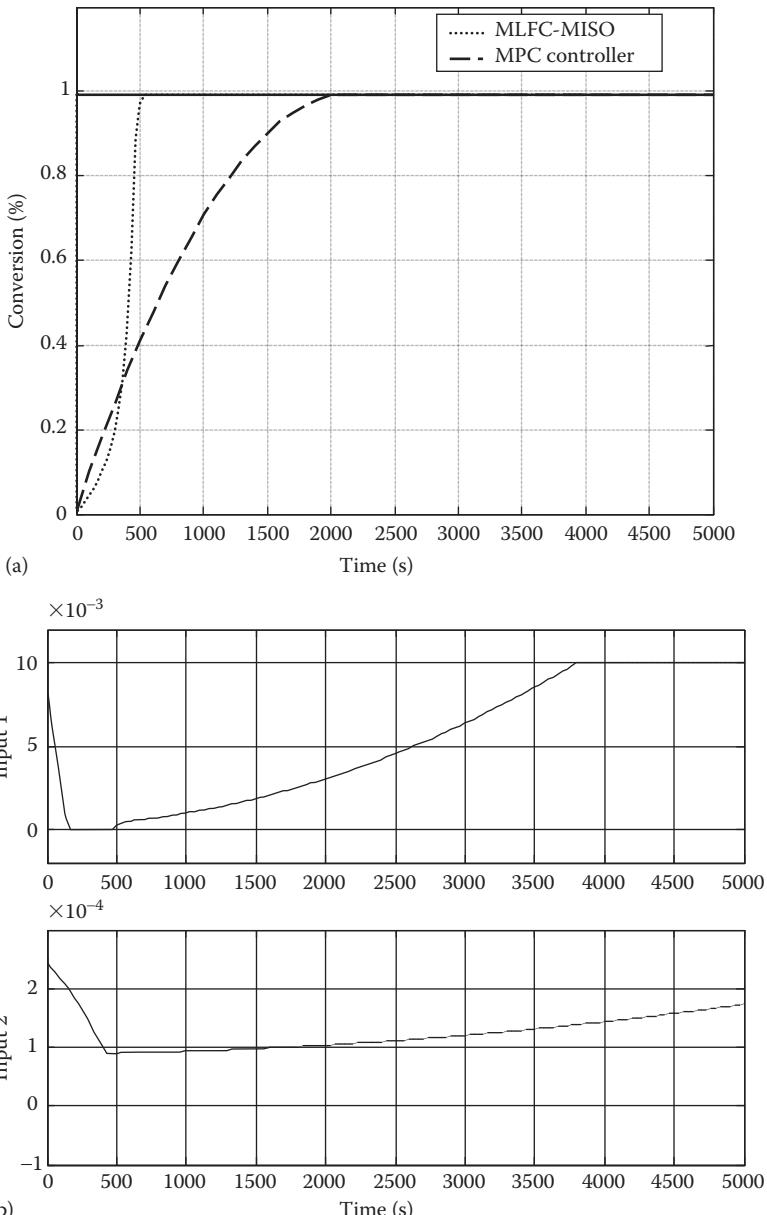


FIGURE 10.10 Simulation comparison under normal condition in Example 10.4:
 (a) conversion rates, (b) control actions with MLFC-MISO controller, and

(continued)

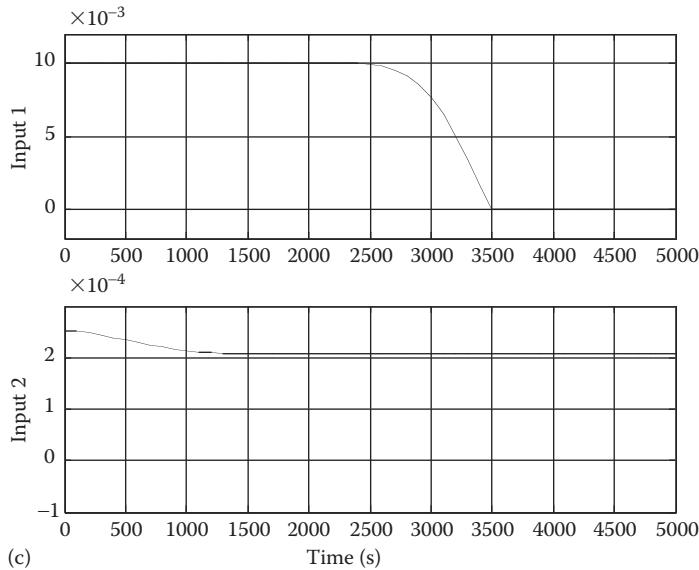


FIGURE 10.10 (continued) (c) control actions with MPC controller.

reach the target value than the MPC does. This is due to the intrinsic adaptation mechanism of the MLFC-MISO controller, which has the ability to adjust the control parameters online following the change of the time-varying parameter. In each sampling time, the control rule table is updated and the better control actions are calculated. Consequently, it takes less time to finish the material conversion.

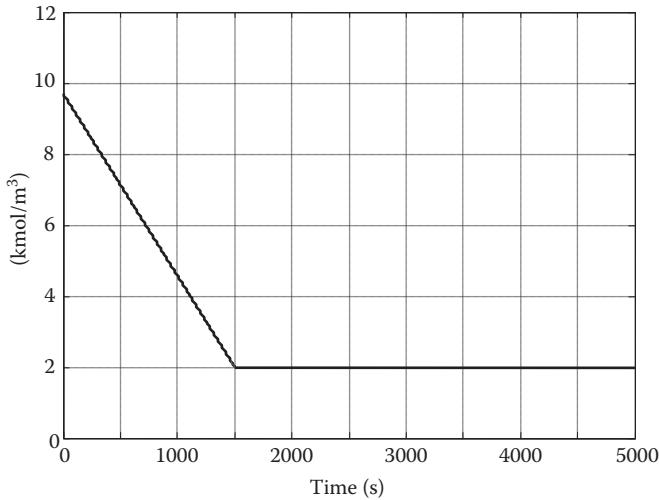


FIGURE 10.11 Time-varying parameter.

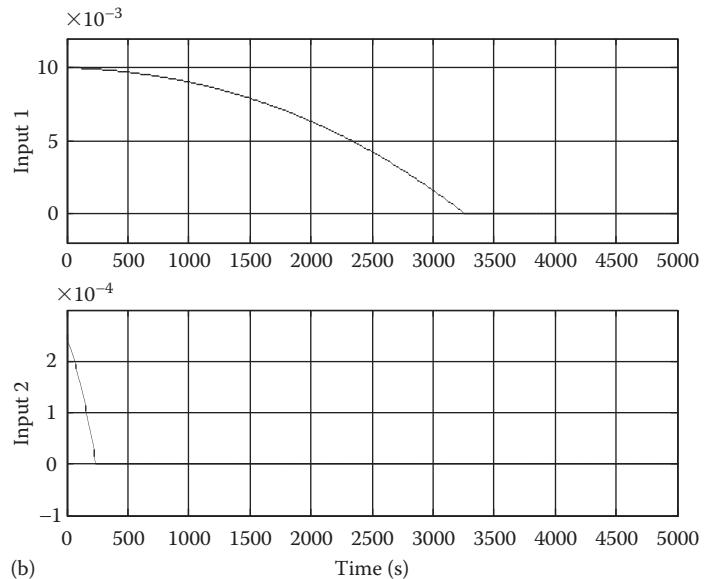
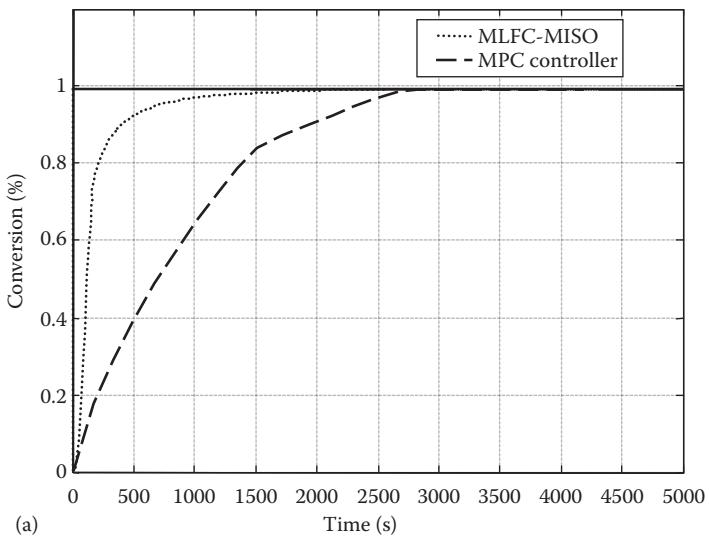


FIGURE 10.12 Simulation comparison with time-varying parameter: (a) conversion rates, (b) control actions with MLFC-MISO controller, and

(continued)

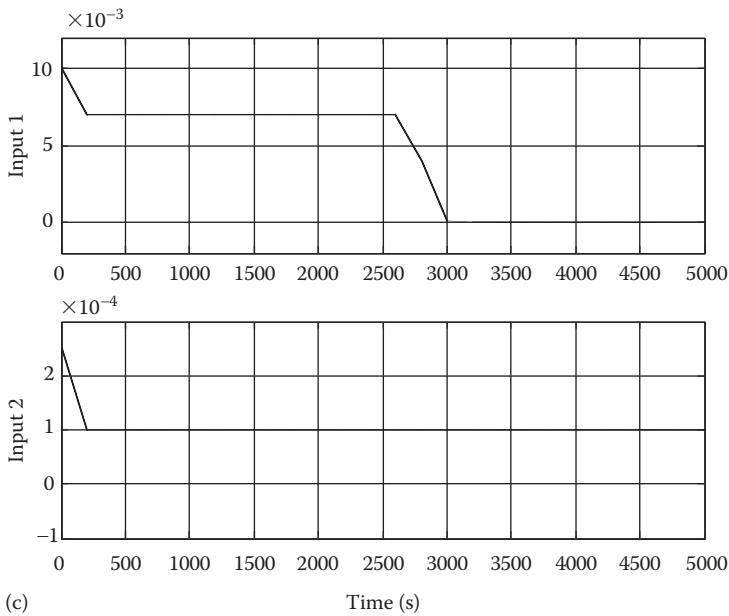


FIGURE 10.12 (continued) (c) control actions with MPC controller.

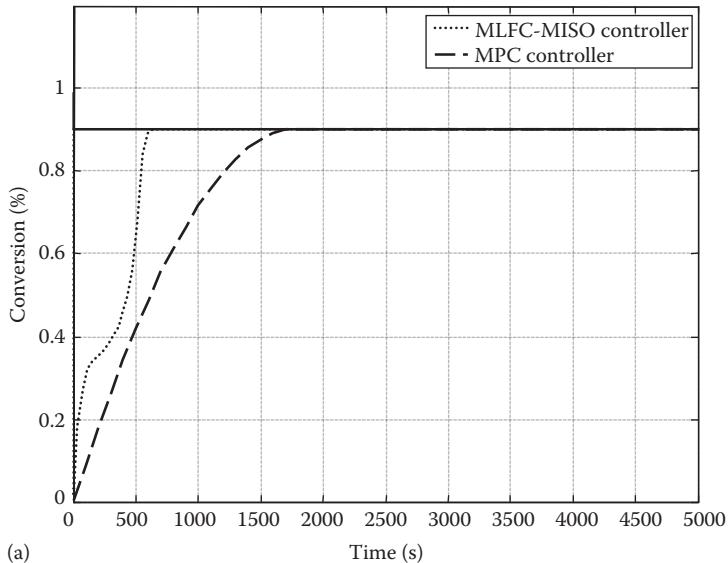
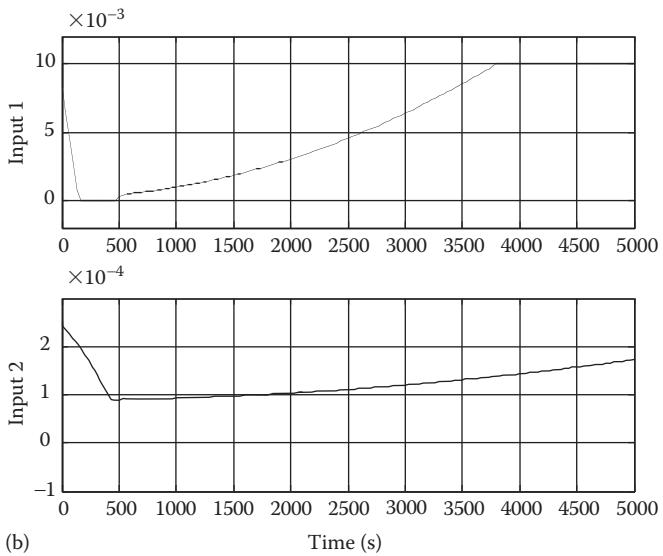
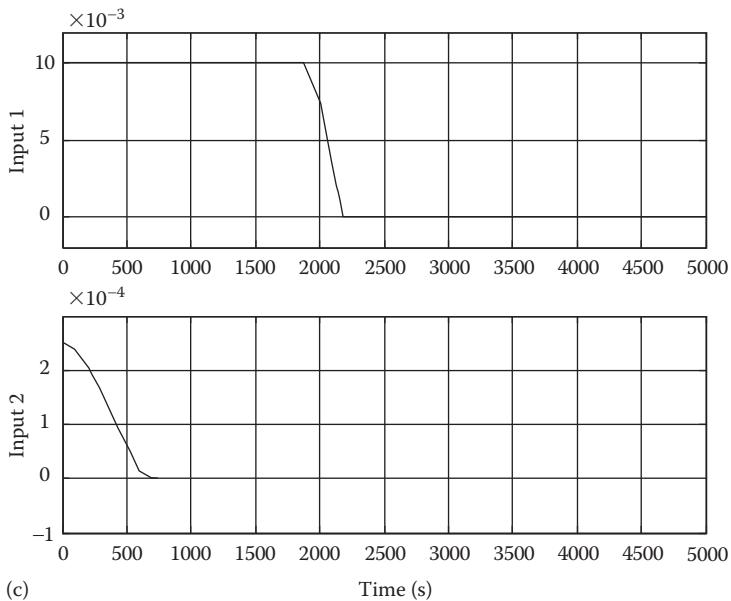


FIGURE 10.13 Simulation comparison with parameter variation under different control objective: (a) conversion rates,



(b)



(c)

FIGURE 10.13 (continued) (b) control actions with MLFC-MISO controller, and (c) control actions with MPC controller.

Example 10.6 With Model Parameter Variation and Different Control Objective

It is assumed the maximum capacity of reactor is changed from $V_{rf} = 6.7$ to 3 m^3 in this case and the control objective is to obtain the conversion percentage of 90%. The control results are provided in [Figure 10.13](#). The MLFC-MISO again shows a superior result than the MPC as in the previous cases.

10.4 CONCLUSION

This chapter presented a systematic procedure for the design of an MLFC for general nonlinear MISO dynamic systems. The proposed fuzzy controller is designed based on the fuzzy inverse model, which can be constructed in such a way to eliminate the multimodality effect in the complex multivariable system. The hierarchical structure for each control variable is designed to update the control parameters online and the resulting closed-loop system for the MISO plant is proved to be input–output passive stable. Two sets of simulation results demonstrate that the proposed fuzzy control technique can be a promising way of controlling time-varying MISO nonlinear systems with inherent system uncertainties and time-varying parameters.

REFERENCES

- Aracil, J. and Gordillo, F., *Stability Issues in Fuzzy Control*, Physica-Verlag, Heidelberg, Germany, 2000.
- Calcev, G., Some remarks in the stability of Mamdani fuzzy control systems, *IEEE Transactions on Fuzzy Systems*, 6(3): 436–442, 1998.
- Chang, J.-S. and Hsieh, W.-Y., Optimization and control of semibatch reactors, *Industrial and Engineering Chemistry Research*, 34: 545–556, 1995.
- Chen, G., Conventional and fuzzy PID controllers: An overview. *International Journal of Intelligent Control Systems*, 2(2): 235–246, 1996.
- Farinwata, S.S., Filev, D., and Langari, R., *Fuzzy Control Synthesis and Analysis*, John Wiley & Sons, Ltd., Chichester, United Kingdom, 2000.
- Hung, J.Y., Albritton, N.G., and Xia, F., Nonlinear control of a magnetic bearing system, *Mechatronics*, 13(6): 621–637, 2003.
- Mann, G.K.I., Hu, B.-G., and Gosine, R.G., Analysis of direct action fuzzy PID controller structures, *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 29(3): 371–388, 1999.
- Mizumoto, M., Realization of PID controls by fuzzy control methods, *IEEE International Conference on Fuzzy Systems*, San Diego, CA, pp. 709–715, 1992.
- Santos, A.I.G., Johansen, T.A., and Cosme, J.M.Z., Nonlinear multiple model predictive control in a fed-batch reactor, *IFAC Symposium on Artificial Intelligence in Real-Time Control*, Budapest, Hungary, 2000.
- Westerterp, K.R., Van Swaaij, W.P.M., and Beenackers, A.A.C.M., *Chemical Reactor Design and Operation*, Wiley, New York, pp. 278–281, 1984.
- Xu, C. and Shin, Y.C., Design of a multi-level fuzzy controller for nonlinear systems and stability analysis, *IEEE Transactions on Fuzzy Systems*, 13(6): 761–778, 2005.
- Xu, C. and Shin, Y.C., A fuzzy inverse model construction method for a general MISO system with a monotonic input–output relationship, *Proceedings of North American Fuzzy Information Processing Society*, June 24–27, San Diego, CA, pp. 1–6, 2007, doi: 10.1109/NAFIPS.2007.383801.
- Ying, H., Practical design of nonlinear fuzzy controllers with stability analysis for regulating processes with unknown mathematical models. *Automatica*, 30(7): 1185–1195, 1994.

11 Knowledge-Based Multivariable Fuzzy Control

Multivariable control is an important issue in industry, where multiple variables often need to be controlled at the same time and interaction effects may exist among the cross-coupled system input/output variables. Most of the existing conventional control techniques rely on the availability of an accurate quantitative model of the system to be controlled, which is not always obtainable for complex real-world applications. Therefore, effective intelligent control methods for such complicated multi-input multi-output (MIMO) systems have been active research topics for decades in both academia and industry.

11.1 COMPLEXITY REDUCTION METHODS

For a multivariable fuzzy system, due to the multidimensionality of the fuzzy relation R , the number of rules will increase exponentially with the number of system input variables. Suppose there are n premise variables in the fuzzy system and each of them has m linguistic values. A total of m^n fuzzy rules will be necessary to cover the overall fuzzy input domain. The resultant compositional fuzzy inferencing calculation is usually too complex to perform for real-time operation. Therefore, it will be useful if we can simplify the fuzzy system structure in order to reduce the memory requirement and online computational load. In this section, different methods are presented and summarized from literature by merging fuzzy terms, pruning fuzzy rules, reducing inputs dimension via mathematical fusion or symbolic fusion, and modular construction of the rule base in order to reduce the multivariable fuzzy system complexity.

11.1.1 RULE BASE SIMPLIFICATION

Suppose a multivariable fuzzy rule base has been established by extracting expert knowledge into linguistic rules or constructing it from numerical input–output data. Following the rule base completeness criterion, the resultant fuzzy rule base needs to cover the entire fuzzy input domain and therefore may contain certain rules that are physically unrealizable, which can be removed from the fuzzy rule base to reduce the complexity and the computation time. On the other hand, the fuzzy rule base constructed from numerical input–output data may contain some degree of redundancy in the form of highly overlapped fuzzy sets. A similar linguistic concept

may be described by more than one fuzzy set, which can be further combined into one set. The rule base simplification criterion can be specified by manual inspection or data-driven approach as follows:

1. If two fuzzy sets for a particular variable are close to each other beyond the threshold limit, they can be fused into a single fuzzy set (Song et al., 1993). The threshold $\lambda \in (0, 1)$ represents the degree above which merging will take place for the specific premise variable. The value of threshold λ is designated by the user. The lower the value of λ , the more fuzzy sets (linguistic terms) can be combined. The closeness of the two fuzzy sets is determined as the cardinality of their intersection divided by the cardinality of their union as

$$c(A, B) = \frac{|\min(\mu_A(x), \mu_B(x))|}{|\max(\mu_A(x), \mu_B(x))|} \quad (11.1)$$

Figure 11.1 illustrates an example of a fuzzy rule base simplification to reduce the number of fuzzy rules, where the first step represents merging fuzzy sets N and Z for variable x into one fuzzy set NZ . The symbol star “*” in the middle table after fusion represents the recomputed membership function (MF) for the consequent variable from the fuzzy sets merging operation. It can be obtained as the weighted average of the individual consequent MF before fusion operation.

2. If a fuzzy set is too narrow respect to its adjacent fuzzy sets, it can be recognized and pruned from the established fuzzy rule base. As shown in

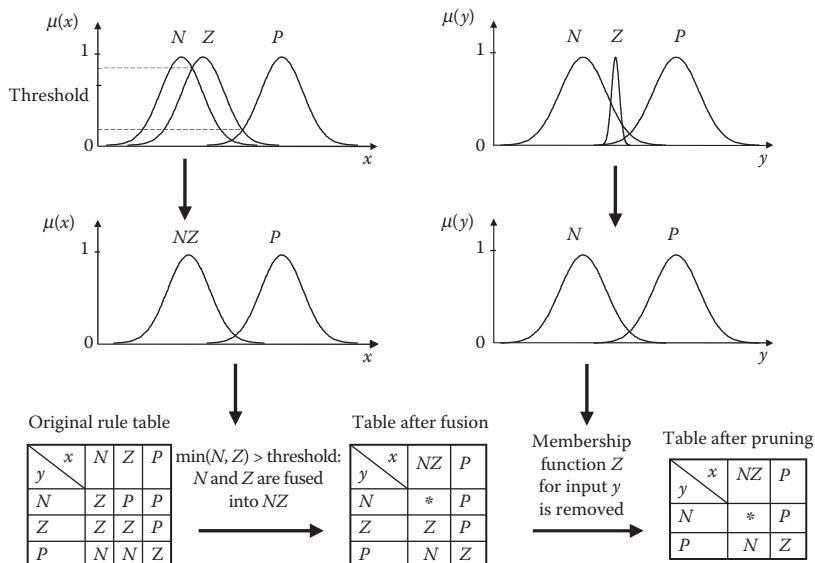


FIGURE 11.1 Fuzzy rule base simplification process. (From Verbruggen, H.B., Zimmermann, H.-J., and Babuška, R., *Fuzzy Algorithms for Control*, Kluwer Academic Publishers, Boston, MA, 1999. With permission.)

the second step in [Figure 11.1](#), the fuzzy set Z for the variable y is removed from the fuzzy rule base, so the final fuzzy rule base structure is simplified and the inferencing calculation load is reduced for real-time applications as well.

11.1.2 DIMENSIONALITY REDUCTION

Unlike the previous method, which mainly focuses on reducing the number of rules by merging the linguistic terms for a certain antecedent variable in a preexisting fuzzy rule base, this method assumes that no rule base has been constructed for the problem under study. By reducing the number of antecedent variables via mathematical fusion or by constructing a multidimensional fuzzy set via symbolic fusion, the multivariable fuzzy rule base synthesis can be effectively reduced.

The first method, a mathematical fusion approach, combines the premise variables linearly by weighted summation before entering into the fuzzy system. As shown in Figure 11.2, three cases are illustrated in terms of two input variables, three input variables, and four input variables, respectively. The weighting factors a , b , c , and d are positive parameters determined by expert experience or physical consideration of the system. However, in general, the premise variables cannot always be trivially fused and combined. Usually, the fusion of the error signal

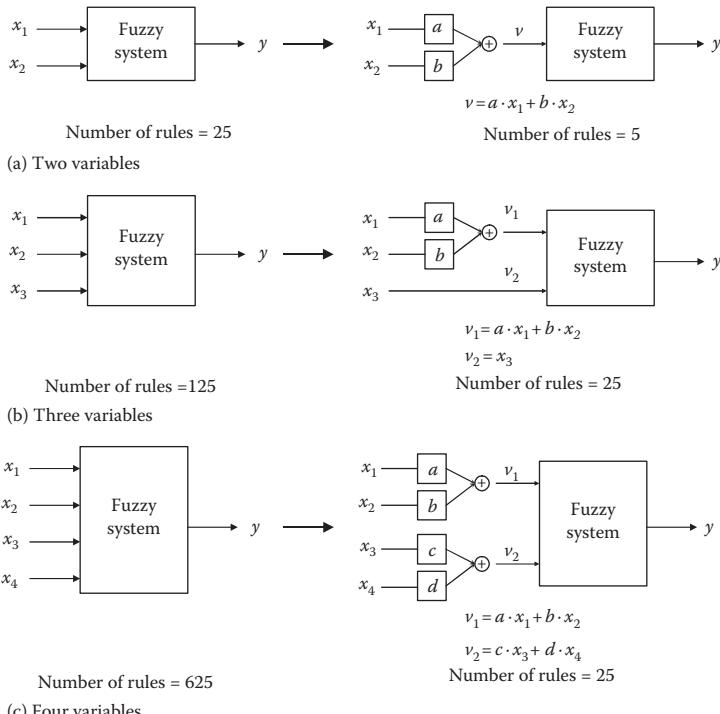


FIGURE 11.2 Mathematical fusion approach for two input variables, three input variables, and four input variables. (From Verbruggen, H.B., Zimmermann, H.-J., and Babuška, R., *Fuzzy Algorithms for Control*, Kluwer Academic Publishers, Boston, MA, 1999. With permission.)

and the change of error gives satisfactory results in many practical applications (Verbruggen, 1999).

The second method of reducing fuzzy rule base dimension is to define a multidimensional fuzzy set, where the premise variables are combined symbolically (Verbruggen, 1999). For example, a multidimensional fuzzy set \mathbf{X} can be defined for a series of input variables $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ by applying the min operator in the Cartesian product space $\mathbf{X} = B_1 \times \dots \times B_n$ with the corresponding membership degree as

$$\mu_{\mathbf{X}}(\mathbf{x}) = \mu_{B_1 \times B_2 \times \dots \times B_n}(x_1, x_2, \dots, x_n) = \min(\mu_{B_1}(x_1), \mu_{B_2}(x_2), \dots, \mu_{B_n}(x_n)) \quad (11.2)$$

where B_1, B_2, \dots, B_n are antecedent MFs for input variables x_1, x_2, \dots, x_n , respectively. And then the fuzzy rule base complexity is reduced exponentially.

11.1.3 STRUCTURED SYSTEMS

With the help of human operator's knowledge, a multivariable fuzzy rule base can be decomposed into a simpler structure to reduce the system complexity, such as a decentralized or hierarchical structure.

Assume the multivariable cross-coupling effect of a considered system is small. Based on the active decomposition of control law, the MIMO fuzzy relation can be decentralized into multiple single-input single-output (SISO) fuzzy relations, which can be designed independently for each individual input–output pair. In this way not only the rule base complexity can be reduced, but the computational load decreases drastically as well (Gegov and Frank, 1995a,b; Gegov, 1996).

On the other hand, with a hierarchically structured fuzzy rule base system, the number of rules is expressed as a linear function of the number of system inputs, instead of an exponential relation (Raju and Zhou, 1991). Considering the case with no multivariable interaction effect, this approach combines the inputs according to the order of their influences on the system, which is often determined based on the knowledge of the system designer. Therefore, a complex multivariable controller is decomposed by starting pairing the most important input variables and composing the intermediate output, which is the premise of the next level fuzzy inferencing. The same pattern reiterates until the final output is obtained as shown in [Figure 11.3](#), where two inputs enter into each fuzzy inferencing level. Suppose there are n premise variables in the overall fuzzy system and each of them has m linguistic values. The resultant hierarchically structured fuzzy system will have a total number of $(n - 1) \cdot m^2$ fuzzy rules. The special advantage of this hierarchical fuzzy system is that an arbitrary input variable can be easily added in or subtracted from the system without affecting other rules in the whole system.

11.2 METHODS TO OPTIMIZE MULTIVARIABLE FUZZY INFERENCE CALCULATION

After identifying a multivariable system and establishing its fuzzy rule base, for a set of system inputs, complex multivariable fuzzy inferencing calculation will involve a derivation process in order to obtain multiple system outputs. Various fuzzy optimization methods have been proposed in literature to expedite the calculation process for multivariable systems.

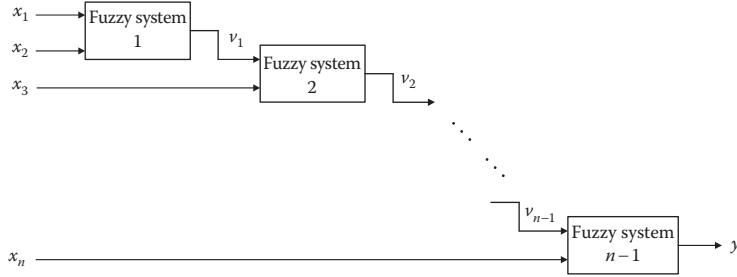


FIGURE 11.3 Hierarchically structured fuzzy rule base system. (From Verbruggen, H.B., Zimmermann, H.-J., and Babuška, R., *Fuzzy Algorithms for Control*, Kluwer Academic Publishers, Boston, MA, 1999. With permission.)

11.2.1 INTERSECTION COEFFICIENTS

DEFINITION 11.1 (Cheng et al., 1982a,b)

If two fuzzy subsets A_i and A_j are convex, the intersection coefficient a_{ij} between these two fuzzy subsets is the maximum value over the span of the universe of discourse X , which is defined as

$$a_{ij} = \max_{x \in X} [\mu_{A_i}(x) \wedge \mu_{A_j}(x)] \quad (11.3)$$

where

$$A_i \subset X$$

$$A_j \subset X$$

a_{ij} is a real number as $a_{ij} \in R$

$\mu_{A_i}(x)$ and $\mu_{A_j}(x)$ are the MFs of the two fuzzy subsets, respectively

It is obvious that if there is no intersection between A_i and A_j , that is, $A_i \cap A_j = \emptyset$, then $a_{ij} = 0$.

Following the same principle, the intersection coefficient b_{kl} between two convex fuzzy subsets B_k and B_l in the universe of discourse Y is defined as

$$b_{kl} = \max_{y \in Y} [\mu_{B_k}(y) \wedge \mu_{B_l}(y)] \quad (11.4)$$

where

$$B_k \subset Y$$

$$B_l \subset Y$$

$$b_{kl} \in R$$

Next, the intersection coefficient β_{ijkl} between two convex fuzzy subsets D_{ik} and D_{jl} in the Cartesian product space $X \times Y$ is defined as

$$\beta_{ijkl} = \max_{(x,y) \in X \times Y} [\mu_{D_{ik}}(x, y) \wedge \mu_{D_{jl}}(x, y)] \quad (11.5)$$

where

$$\begin{aligned} D_{ik} &= A_i \times B_k \subset X \times Y \\ D_{jl} &= A_j \times B_l \subset X \times Y \\ \beta_{ijkl} &\in R \end{aligned}$$

THEOREM 11.1 (Cheng et al., 1982a,b)

The intersection coefficient β_{ijkl} in the Cartesian product space $X \times Y$ is equal to the smaller value of the intersection coefficients in the universes of discourse X and Y :

$$\begin{aligned} \beta_{ijkl} &= \max_{(x,y) \in X \times Y} [\mu_{D_{ik}}(x, y) \wedge \mu_{D_{jl}}(x, y)] \\ &= \max_{(x,y) \in X \times Y} \left\{ [\mu_{A_i}(x) \wedge \mu_{B_k}(y)] \wedge [\mu_{A_j}(x) \wedge \mu_{B_l}(y)] \right\} \\ &= \max_{(x,y) \in X \times Y} \left\{ [\mu_{A_i}(x) \wedge \mu_{A_j}(x)] \wedge [\mu_{B_k}(y) \wedge \mu_{B_l}(y)] \right\} \\ &= \max_{x \in X} [\mu_{A_i}(x) \wedge \mu_{A_j}(x)] \wedge \max_{y \in Y} [\mu_{B_k}(y) \wedge \mu_{B_l}(y)] \\ &= a_{ij} \wedge b_{kl} \end{aligned} \quad (11.6)$$

It is then evident that

$$\beta_{ijkl} = \begin{cases} 0 & \text{if } A_i \cap A_j = \emptyset \text{ or } B_k \cap B_l = \emptyset \\ 1 & \text{if } i = j \text{ and } k = l \\ (0, 1) & \text{otherwise} \end{cases} \quad (11.7)$$

■

Example 11.1 Expression of a Fuzzy PD Controller for a General SISO Plant Based on the Intersection Coefficient (Cheng et al., 1982b)

Consider a fuzzy PD controller for a general SISO system as shown in [Figure 11.4](#).

The fuzzy control rule can be expressed as

$$R_{ik}: \text{ IF } e \text{ is } A_i \text{ and } \Delta e \text{ is } B_k, \text{ THEN } u \text{ is } C_{ik} \quad (11.8)$$

where

A_i and B_k represent the linguistic values for the error signal e and the change of error

Δe , within the universes of discourse E and ΔE , respectively

C_{ik} is the linguistic value for the control action u

Each control rule R_{ik} can be defined as a triple-dimensional fuzzy relation and described as

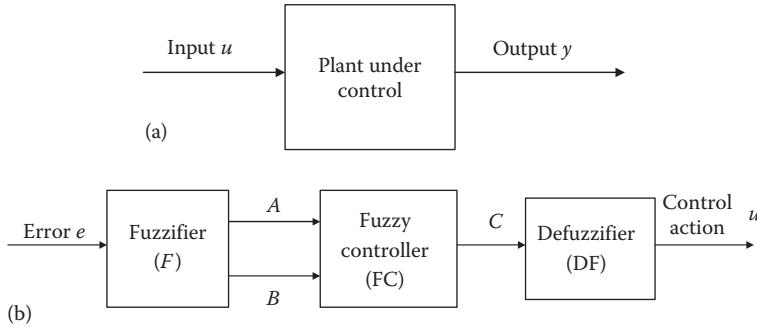


FIGURE 11.4 (a) SISO plant. (b) Fuzzy PD controller for a SISO plant. (From Cheng, W.-M., Ren, S.-J., Wu, C.-F., and Tsuei, T.-H., *Fuzzy Information and Decision Processes*, North-Holland Publishing Company, Amsterdam, the Netherlands, 1982b. With permission.)

$$R_{ik} = (A_i \times B_k) \times C_{ik} \quad (11.9)$$

where the first symbol “ \times ” represents the Cartesian product operation, that is, min intersection, and the second symbol “ \times ” denotes the fuzzy implication, that is, Mamdani min operation. With this declaration, the control rule R_{ik} can be denoted as $\mu_{R_{ik}}(e, \Delta e, u) = [\mu_{A_i}(e) \wedge \mu_{B_k}(\Delta e)] \wedge \mu_{C_{ik}}(u)$ and the entire control rule base can be combined as the union of each individual rule as

$$R = \bigcup_{i,k} R_{ik} \quad (11.10)$$

where $i = 1, \dots, r_1$ and $k = 1, \dots, r_2$ represent the dimensions of the control rule base for the error e and the change of error Δe , respectively. If the current fuzzy input variables are A_j and B_l , then the fuzzy control output C_{jl} can be calculated as

$$\begin{aligned} \mu_{C_{jl}}(u) &= \left[\mu_{A_j}(e) \wedge \mu_{B_l}(\Delta e) \right] \circ \mu_R(e, \Delta e, u) \\ &= \mu_{D_{jl}}(e, \Delta e) \circ \left[\bigcup_{i,k} \mu_{R_{ik}}(e, \Delta e, u) \right] \\ &= \bigcup_{i,k} \left[\mu_{D_{jl}}(e, \Delta e) \circ \mu_{R_{ik}}(e, \Delta e, u) \right] \\ &= \bigcup_{i,k} \left(\mu_{D_{jl}}(e, \Delta e) \circ \{ [\mu_{A_i}(e) \wedge \mu_{B_k}(\Delta e)] \wedge \mu_{C_{ik}}(u) \} \right) \\ &= \bigcup_{i,k} \max_{(e, \Delta e) \in E \times \Delta E} \left\{ \mu_{D_{jl}}(e, \Delta e) \wedge [\mu_{D_{ik}}(e, \Delta e) \wedge \mu_{C_{ik}}(u)] \right\} \\ &= \bigcup_{i,k} \left\{ \max_{(e, \Delta e) \in E \times \Delta E} [\mu_{D_{jl}}(e, \Delta e) \wedge \mu_{D_{ik}}(e, \Delta e)] \wedge \mu_{C_{ik}}(u) \right\} \\ &= \bigcup_{i,k} \left[\beta_{ijkl} \wedge \mu_{C_{ik}}(u) \right] \end{aligned} \quad (11.11)$$

Example 11.2 Expression of a Multivariable Fuzzy PD Control System for a General Two-Input Two-Output Plant Based on the Intersection Coefficients (Cheng et al., 1982b)

Next, the fuzzy control expression is extended to multivariable cases, such as a two-input two-output case as shown in Figure 11.5.

In Figure 11.5b, the symbols A^1 and B^1 represent the fuzzified error e_1 and the change of error Δe_1 , respectively, A^2 and B^2 are the fuzzified e_2 and Δe_2 , C^{11} is the fuzzified control action u_1 calculated from e_1 , C^{12} is the fuzzified control action u_2 calculated from e_1 , C^{21} is the fuzzified control action u_1 calculated from e_2 , and C^{22} is the fuzzified control action u_2 calculated from e_2 . The fuzzy control rule for the control action u_1 calculated from e_1 can be represented as

$$R_{11ik}: \text{IF } e_1 \text{ is } A_{1i} \text{ AND } \Delta e_1 \text{ is } B_{1k}, \text{ THEN } u_1 \text{ is } C_{11ik} \quad (11.12)$$

where

A_{1i} and B_{1k} represent the linguistic values for the error signal e_1 and the change of error Δe_1 , within the universes of discourse E_1 and ΔE_1 , respectively
 C_{11ik} is the linguistic value for the control action u_1 from e_1

If the current fuzzy input variables are A_{1j} and B_{1l} for e_1 and Δe_1 , respectively, from the previous derivation for the SISO case, the fuzzy control output C_{11jl} can be obtained from the extension of the single variable case as

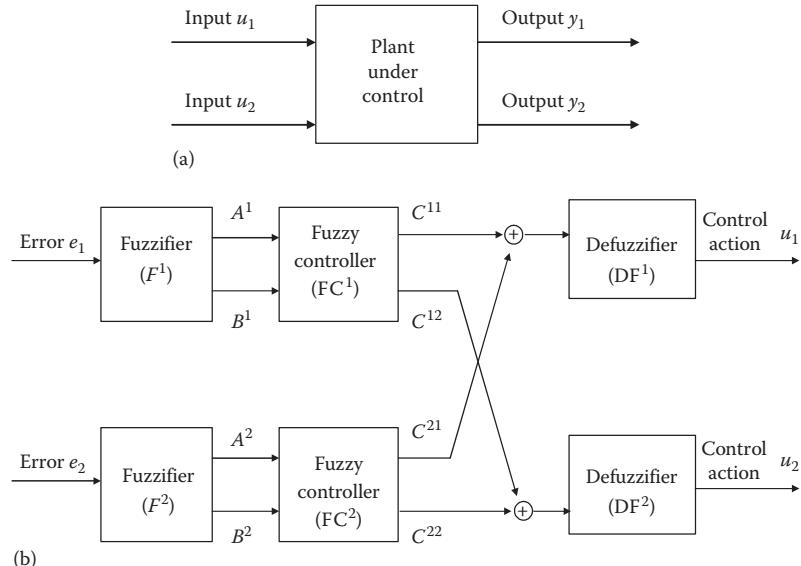


FIGURE 11.5 (a) Two-input two-output plant. (b) Multivariable fuzzy PD control system for a two-input two-output plant. (From Cheng, W.-M., Ren, S.-J., Wu, C.-F., and Tsuei, T.-H., *Fuzzy Information and Decision Processes*, North-Holland Publishing Company, Amsterdam, the Netherlands, 1982b. With permission.)

$$\begin{aligned}
\mu_{C_{11jl}}(u_1) &= [\mu_{A_{1j}}(e_1) \wedge \mu_{B_{1l}}(\Delta e_1)] \circ \mu_{R_{11}}(e_1, \Delta e_1, u_1) \\
&= \mu_{D_{11jl}}(e_1, \Delta e_1) \circ \left[\bigcup_{i,k} \mu_{R_{11ik}}(e_1, \Delta e_1, u_1) \right] \\
&= \bigcup_{i,k} \left[\mu_{D_{11jl}}(e_1, \Delta e_1) \circ \mu_{R_{11ik}}(e_1, \Delta e_1, u_1) \right] \\
&= \bigcup_{i,k} (\mu_{D_{11jl}}(e_1, \Delta e_1) \circ \{\mu_{A_{1i}}(e_1) \wedge \mu_{B_{1k}}(\Delta e_1)\} \wedge \mu_{C_{11ik}}(u_1)) \\
&= \bigcup_{i,k} \max_{(e_1, \Delta e_1) \in E_1 \times \Delta E_1} \{\mu_{D_{11jl}}(e_1, \Delta e_1) \wedge [\mu_{D_{11ik}}(e_1, \Delta e_1) \wedge \mu_{C_{11ik}}(u_1)]\} \\
&= \bigcup_{i,k} \left\{ \max_{(e_1, \Delta e_1) \in E_1 \times \Delta E_1} [\mu_{D_{11jl}}(e_1, \Delta e_1) \wedge \mu_{D_{11ik}}(e_1, \Delta e_1)] \wedge \mu_{C_{11ik}}(u_1) \right\} \\
&= \bigcup_{i,k} [\beta_{ijkl}^1 \wedge \mu_{C_{11ik}}(u_1)]
\end{aligned} \tag{11.13}$$

where β_{ijkl}^1 represents the intersection coefficient in the Cartesian product space $E_1 \times \Delta E_1$ as

$$\begin{aligned}
\beta_{ijkl}^1 &= \max_{(e_1, \Delta e_1) \in E_1 \times \Delta E_1} \{[\mu_{A_{1j}}(e_1) \wedge \mu_{B_{1l}}(\Delta e_1)] \wedge [\mu_{A_{1i}}(e_1) \wedge \mu_{B_{1k}}(\Delta e_1)]\} \\
&= \max_{(e_1, \Delta e_1) \in E_1 \times \Delta E_1} [\mu_{D_{11jl}}(e_1, \Delta e_1) \wedge \mu_{D_{11ik}}(e_1, \Delta e_1)]
\end{aligned} \tag{11.14}$$

Similarly, the fuzzy control rules for C^{12} , C^{21} , and C^{22} can be represented as

$$R_{12ik}: \text{IF } e_1 \text{ is } A_{1i} \text{ AND } \Delta e_1 \text{ is } B_{1k}, \text{ THEN } u_2 \text{ is } C_{12ik} \tag{11.15a}$$

$$R_{21ik}: \text{IF } e_2 \text{ is } A_{2i} \text{ AND } \Delta e_2 \text{ is } B_{2k}, \text{ THEN } u_1 \text{ is } C_{21ik} \tag{11.15b}$$

$$R_{22ik}: \text{IF } e_2 \text{ is } A_{2i} \text{ AND } \Delta e_2 \text{ is } B_{2k}, \text{ THEN } u_2 \text{ is } C_{22ik} \tag{11.15c}$$

If the current fuzzy input variables are A_{2j} and B_{2l} for e_2 and Δe_2 , respectively, then the intersection coefficient in the Cartesian product space $E_2 \times \Delta E_2$ can be denoted as

$$\begin{aligned}
\beta_{ijkl}^2 &= \max_{(e_2, \Delta e_2) \in E_2 \times \Delta E_2} \left\{ [\mu_{A_{2j}}(e_2) \wedge \mu_{B_{2l}}(\Delta e_2)] \wedge [\mu_{A_{2i}}(e_2) \wedge \mu_{B_{2k}}(\Delta e_2)] \right\} \\
&= \max_{(e_2, \Delta e_2) \in E_2 \times \Delta E_2} [\mu_{D_{22jl}}(e_2, \Delta e_2) \wedge \mu_{D_{22ik}}(e_2, \Delta e_2)]
\end{aligned} \tag{11.16}$$

And then the fuzzy control outputs C_{12jl} , C_{21jl} , and C_{22jl} can be calculated as

$$\begin{aligned}
\mu_{C_{12jl}}(u_2) &= [\mu_{A_{1j}}(e_1) \wedge \mu_{B_{1l}}(\Delta e_1)] \circ \mu_{R_{12}}(e_1, \Delta e_1, u_2) \\
&= \bigcup_{i,k} \left[\mu_{D_{11jl}}(e_1, \Delta e_1) \circ \mu_{R_{12ik}}(e_1, \Delta e_1, u_2) \right] \\
&= \bigcup_{i,k} \max_{(e_1, \Delta e_1) \in E_1 \times \Delta E_1} \left\{ \mu_{D_{11jl}}(e_1, \Delta e_1) \wedge [\mu_{D_{11ik}}(e_1, \Delta e_1) \wedge \mu_{C_{12ik}}(u_2)] \right\} \\
&= \bigcup_{i,k} \left\{ \max_{(e_1, \Delta e_1) \in E_1 \times \Delta E_1} [\mu_{D_{11jl}}(e_1, \Delta e_1) \wedge \mu_{D_{11ik}}(e_1, \Delta e_1)] \wedge \mu_{C_{12ik}}(u_2) \right\} \\
&= \bigcup_{i,k} [\beta_{ijkl}^2 \wedge \mu_{C_{12ik}}(u_2)]
\end{aligned} \tag{11.17}$$

$$\begin{aligned}
\mu_{C_{21jl}}(u_1) &= \left[\mu_{A_{2j}}(e_2) \wedge \mu_{B_{2l}}(\Delta e_2) \right] \circ \mu_{R_{21}}(e_2, \Delta e_2, u_1) \\
&= \bigcup_{i,k} \left[\mu_{D_{22jl}}(e_2, \Delta e_2) \circ \mu_{R_{21ik}}(e_2, \Delta e_2, u_1) \right] \\
&= \bigcup_{i,k} \max_{(e_2, \Delta e_2) \in E_2 \times \Delta E_2} \left\{ \mu_{D_{22jl}}(e_2, \Delta e_2) \wedge [\mu_{D_{22ik}}(e_2, \Delta e_2) \wedge \mu_{C_{21ik}}(u_1)] \right\} \\
&= \bigcup_{i,k} \left\{ \max_{(e_2, \Delta e_2) \in E_2 \times \Delta E_2} \left[\mu_{D_{22jl}}(e_2, \Delta e_2) \wedge \mu_{D_{22ik}}(e_2, \Delta e_2) \right] \wedge \mu_{C_{21ik}}(u_1) \right\} \\
&= \bigcup_{i,k} \left[\beta_{ijkl}^2 \wedge \mu_{C_{21ik}}(u_1) \right]
\end{aligned} \tag{11.18}$$

$$\begin{aligned}
\mu_{C_{22jl}}(u_2) &= \left[\mu_{A_{2j}}(e_2) \wedge \mu_{B_{2l}}(\Delta e_2) \right] \circ \mu_{R_{22}}(e_2, \Delta e_2, u_2) \\
&= \bigcup_{i,k} \left[\mu_{D_{22jl}}(e_2, \Delta e_2) \circ \mu_{R_{22ik}}(e_2, \Delta e_2, u_2) \right] \\
&= \bigcup_{i,k} \max_{(e_2, \Delta e_2) \in E_2 \times \Delta E_2} \left\{ \mu_{D_{22jl}}(e_2, \Delta e_2) \wedge [\mu_{D_{22ik}}(e_2, \Delta e_2) \wedge \mu_{C_{22ik}}(u_2)] \right\} \\
&= \bigcup_{i,k} \left\{ \max_{(e_2, \Delta e_2) \in E_2 \times \Delta E_2} \left[\mu_{D_{22jl}}(e_2, \Delta e_2) \wedge \mu_{D_{22ik}}(e_2, \Delta e_2) \right] \wedge \mu_{C_{22ik}}(u_2) \right\} \\
&= \bigcup_{i,nk} \left[\beta_{ijkl}^2 \wedge \mu_{C_{22ik}}(u_2) \right]
\end{aligned} \tag{11.19}$$

The resultant two fuzzy control outputs for u_1 and u_2 are obtained as

$$\begin{aligned}
\mu_{C_{jl}}(u_1) &= \mu_{C_{11jl}}(u_1) \vee \mu_{C_{21jl}}(u_1) = \left\{ \bigcup_{i,k} \left[\beta_{ijkl}^1 \wedge \mu_{C_{11ik}}(u_1) \right] \right\} \vee \left\{ \bigcup_{i,k} \left[\beta_{ijkl}^2 \wedge \mu_{C_{21ik}}(u_1) \right] \right\} \\
\mu_{C_{jl}}(u_2) &= \mu_{C_{12jl}}(u_2) \vee \mu_{C_{22jl}}(u_2) = \left\{ \bigcup_{i,k} \left[\beta_{ijkl}^1 \wedge \mu_{C_{12ik}}(u_2) \right] \right\} \vee \left\{ \bigcup_{i,k} \left[\beta_{ijkl}^2 \wedge \mu_{C_{22ik}}(u_2) \right] \right\}
\end{aligned} \tag{11.20}$$

11.2.2 DECOMPOSITION OF A MULTIDIMENSIONAL FUZZY RULE BASE

Consider a general multidimensional fuzzy control rule base with n inputs and p outputs as

$$R^i: \text{IF } x_1 \text{ is } A_{1i} \text{ AND } \dots \text{ AND } x_n \text{ is } A_{ni}, \text{ THEN } u_1 \text{ is } B_{1i} \text{ AND } \dots \text{ AND } u_p \text{ is } B_{pi} \tag{11.21}$$

where

R^i ($i = 1, \dots, r$) represents the i th fuzzy rule

x_1, \dots, x_n are n controller input variables, which are assumed to be independent of each other

u_1, \dots, u_p are p controller output variables

A_{1i}, \dots, A_{ni} and B_{1i}, \dots, B_{pi} are linguistic values for each input and output variable in the universes of discourse X_1, \dots, X_n and U_1, \dots, U_p , respectively

Suppose the current fuzzy input variables are A'_1, \dots, A'_n . With min intersection Cartesian product operation and Mamdani's min fuzzy implication, the output fuzzy set B' in the Cartesian product space $U_1 \times \dots \times U_p$ can be determined from fuzzy inferencing calculation as

$$\begin{aligned}
\mu_{B'}(u_1, \dots, u_p) &= [\mu_{A'_1}(x_1) \wedge \dots \wedge \mu_{A'_n}(x_n)] \circ \mu_R(x_1, \dots, x_n, \mu_1, \dots, u_p) \\
&= \bigcup_{x_1, \dots, x_n} \left\{ [\mu_{A'_1}(x_1) \wedge \dots \wedge \mu_{A'_n}(x_n)] \wedge \mu_R(x_1, \dots, x_n, u_1, \dots, u_p) \right\} \\
&= \bigcup_{x_1} \left(\mu_{A'_1}(x_1) \wedge \left\{ \dots \wedge \bigcup_{x_n} [\mu_{A'_n}(x_n) \wedge \mu_R(x_1, \dots, x_n, u_1, \dots, u_p)] \right\} \right) \\
&= \bigcup_{x_1} \left(\mu_{A'_1}(x_1) \wedge \left\{ \dots \wedge \bigcup_{x_n} [\mu_{A'_n}(x_n) \wedge \bigcup_{i=1}^r [\mu_{B'_{1i}}(x_1, \dots, x_n, u_1, \dots, u_p)]] \right\} \right) \\
&= \bigcup_{x_1} \left[\mu_{A'_1}(x_1) \wedge \left(\dots \wedge \bigcup_{x_n} \left\{ \mu_{A'_n}(x_n) \wedge \bigcup_{i=1}^r [\mu_{A_{1i}}(x_1) \wedge \dots \wedge \mu_{A_{ni}}(x_n) \wedge \mu_{B_{1i}}(u_1) \wedge \dots \wedge \mu_{B_{pi}}(u_p)] \right\} \right) \right] \\
&= \bigcup_{i=1}^r \bigcup_{x_1} \left[\mu_{A'_1}(x_1) \wedge \left(\dots \wedge \bigcup_{x_n} \left\{ \mu_{A'_n}(x_n) \wedge [\mu_{A_{1i}}(x_1) \wedge \dots \wedge \mu_{A_{ni}}(x_n) \wedge \mu_{B_{1i}}(u_1) \wedge \dots \wedge \mu_{B_{pi}}(u_p)] \right\} \right) \right] \\
&= \bigcup_{i=1}^r \left\{ \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{1i}}(x_1)] \wedge \dots \wedge \bigcup_{x_n} [\mu_{A'_n}(x_n) \wedge \mu_{A_{ni}}(x_n)] \wedge [\mu_{B_{1i}}(u_1) \wedge \dots \wedge \mu_{B_{pi}}(u_p)] \right\} \\
&= \bigcup_{i=1}^r \left(\begin{array}{c} \left\{ \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{1i}}(x_1)] \wedge \mu_{B_{1i}}(u_1) \right\} \wedge \dots \wedge \left\{ \bigcup_{x_n} [\mu_{A'_n}(x_n) \wedge \mu_{A_{ni}}(x_n)] \wedge \mu_{B_{1i}}(u_1) \right\} \\ \vdots \\ \left\{ \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{1i}}(x_1)] \wedge \mu_{B_{pi}}(u_p) \right\} \wedge \dots \wedge \left\{ \bigcup_{x_n} [\mu_{A'_n}(x_n) \wedge \mu_{A_{ni}}(x_n)] \wedge \mu_{B_{pi}}(u_p) \right\} \end{array} \right) \\
&= \bigcup_{i=1}^r \left\{ \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{1i}}(x_1)] \wedge \mu_{B_{1i}}(u_1) \right\} \wedge \dots \wedge \bigcup_{i=1}^r \left\{ \bigcup_{x_n} [\mu_{A'_n}(x_n) \wedge \mu_{A_{ni}}(x_n)] \wedge \mu_{B_{1i}}(u_1) \right\} \\
&\quad \wedge \\
&\quad \vdots \\
&\quad \wedge \\
&= \bigcup_{i=1}^r \left\{ \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{1i}}(x_1)] \wedge \mu_{B_{pi}}(u_p) \right\} \wedge \dots \wedge \bigcup_{i=1}^r \left\{ \bigcup_{x_n} [\mu_{A'_n}(x_n) \wedge \mu_{A_{ni}}(x_n)] \wedge \mu_{B_{pi}}(u_p) \right\}
\end{aligned} \tag{11.22}$$

Denote

$$\gamma_{ki} = \bigcup_{x_k} [\mu_{A'_k}(x_k) \wedge \mu_{A_{ki}}(x_k)] \tag{11.23}$$

then each individual output fuzzy set B'_j for the j th output can be represented as

$$\mu_{B'_j}(u_j) = \bigcap_{k=1}^n \left\{ \bigcup_{i=1}^r [\gamma_{ki} \wedge \mu_{B_{ji}}(u_j)] \right\} \tag{11.24}$$

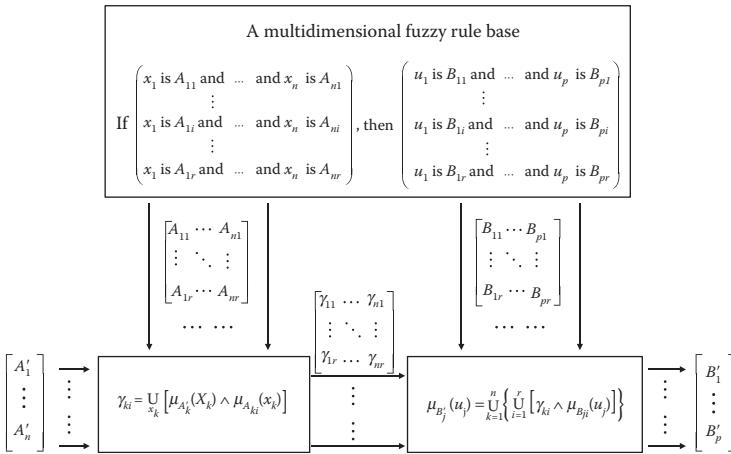


FIGURE 11.6 Decomposition of a multidimensional fuzzy rule base. (From Walichiewicz, L., *Cybernetics and Systems Research 2: Proceedings of the 7th European Meeting on Cybernetics and Systems Research*, North-Holland, NY, 1984. With permission.)

The structure of decomposition for a multidimensional fuzzy rule base is shown in Figure 11.6.

Example 11.3 Decomposition of a Two-Input Two-Output Fuzzy Control Rule Base (Walichiewicz, 1984)

Consider a two-input two-output closed-loop fuzzy control system as shown in Figure 11.7.

The fuzzy control rules are of the form

$$R': \text{IF } x_1 \text{ is } A_{1i} \text{ AND } x_2 \text{ is } A_{2i}, \text{ THEN } u_1 \text{ is } B_{1i} \text{ AND } u_2 \text{ is } B_{2i} \quad (11.25)$$

which are derived from human experts and are assumed to contain three rules as follows:

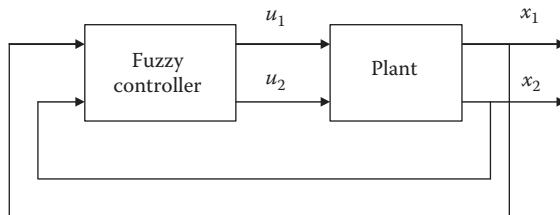


FIGURE 11.7 Two-input two-output closed-loop fuzzy control system. (From Walichiewicz, L., *Cybernetics and Systems Research 2: Proceedings of the 7th European Meeting on Cybernetics and Systems Research*, 1984. With permission.)

- R^1 : IF x_1 is small AND x_2 is small, THEN u_1 is small AND u_2 is big
 R^2 : IF x_1 is medium AND x_2 is medium, THEN u_1 is medium AND u_2 is medium
 R^3 : IF x_1 is big AND x_2 is big, THEN u_1 is big AND u_2 is small

(11.26)

Each universe of discourse of X_1 , X_2 , U_1 , and U_2 is represented by 11 discrete elements as:

$$X_1 = \{x_1^1, x_1^2, \dots, x_1^{11}\}, \quad X_2 = \{x_2^1, x_2^2, \dots, x_2^{11}\} \\ U_1 = \{u_1^1, u_1^2, \dots, u_1^{11}\}, \quad U_2 = \{u_2^1, u_2^2, \dots, u_2^{11}\}$$
(11.27)

The three fuzzy sets are defined in Table 11.1.
where

$$Y_k = \{y_k^1, y_k^2, \dots, y_k^{11}\} = \begin{cases} X_k = \{x_k^1, x_k^2, \dots, x_k^{11}\} & \text{for } k = 1, 2 \\ U_{k-2} = \{u_{k-2}^1, u_{k-2}^2, \dots, u_{k-2}^{11}\} & \text{for } k = 3, 4 \end{cases}$$
(11.28)

The actual fuzzy inputs A'_1 and A'_2 are given as follows:

$$\begin{aligned} A'_1 &= 0/x_1^1 + 0.2/x_1^2 + 1/x_1^3 + 0.2/x_1^4 + 0/x_1^5 + 0/x_1^6 + 0/x_1^7 \\ &\quad + 0/x_1^8 + 0/x_1^9 + 0/x_1^{10} + 0/x_1^{11} \\ &= [0 \quad 0.2 \quad 1 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \\ A'_2 &= 1/x_2^1 + 0.1/x_2^2 + 0/x_2^3 + 0/x_2^4 + 0/x_2^5 + 0/x_2^6 + 0/x_2^7 \\ &\quad + 0/x_2^8 + 0/x_2^9 + 0/x_2^{10} + 0/x_2^{11} \\ &= [1 \quad 0.1 \quad 0 \quad 0] \end{aligned}$$
(11.29)

Since 11 elements are defined in each universe of discourse of X_1 , X_2 , U_1 , and U_2 , the overall fuzzy control system will contain $11^4 = 14,641$ rules. For each pair of input variables A'_1 and A'_2 , 14,641 iterations of fuzzy inferencing calculation are required to obtain the resultant fuzzy output B'_1 and B'_2 . The computation load is quite heavy for real-time implementation. In comparison, by employing the decomposition method for multidimensional fuzzy control rules, the calculation will be drastically simplified, where each coefficient γ_{ki} is calculated as

TABLE 11.1
Definition of the Three Fuzzy Sets

	y_k^1	y_k^2	y_k^3	y_k^4	y_k^5	y_k^6	y_k^7	y_k^8	y_k^9	y_k^{10}	y_k^{11}
Small	1.0	0.75	0.5	0.25	0	0	0	0	0	0	0
Medium	0	0	0	0.3	0.5	1	0.5	0.3	0	0	0
Big	0	0	0	0	0	0	0	0.25	0.5	0.75	1

Source: Walichiewicz, L., *Cybernetics and Systems Research 2: Proceedings of the 7th European Meeting on Cybernetics and Systems Research*, North-Holland, NY, 1984. With permission.

$$\begin{aligned}
\gamma_{11} &= \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{11}}(x_1)] \\
&= \bigcup_{x_1} \left\{ [0 \quad 0.2 \quad 1 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \right\} \\
&= \bigcup_{x_1} \left\{ [0 \quad 0.2 \quad 0.5 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \right\} \\
&= 0.5
\end{aligned} \tag{11.30}$$

$$\begin{aligned}
\gamma_{12} &= \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{12}}(x_1)] \\
&= \bigcup_{x_1} \left\{ [0 \quad 0.2 \quad 1 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \right\} \\
&= \bigcup_{x_1} \left\{ [0 \quad 0 \quad 0 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \right\} \\
&= 0.2
\end{aligned} \tag{11.31}$$

$$\begin{aligned}
\gamma_{13} &= \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{13}}(x_1)] \\
&= \bigcup_{x_1} \left\{ [0 \quad 0.2 \quad 1 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \right\} \\
&= \bigcup_{x_1} \left\{ [0 \quad 0 \quad 0] \right\} \\
&= 0
\end{aligned} \tag{11.32}$$

$$\begin{aligned}
\gamma_{21} &= \bigcup_{x_2} [\mu_{A'_2}(x_2) \wedge \mu_{A_{21}}(x_2)] \\
&= \bigcup_{x_2} \left\{ [1 \quad 0.1 \quad 0 \quad 0] \right\} \\
&= \bigcup_{x_2} \left\{ [1 \quad 0.1 \quad 0 \quad 0] \right\} \\
&= 1
\end{aligned} \tag{11.33}$$

$$\begin{aligned}
\gamma_{22} &= \bigcup_{x_2} [\mu_{A'_2}(x_2) \wedge \mu_{A_{22}}(x_2)] \\
&= \bigcup_{x_2} \left\{ [1 \quad 0.1 \quad 0 \quad 0] \right\} \\
&= \bigcup_{x_2} \left\{ [0 \quad 0 \quad 0] \right\} \\
&= 0
\end{aligned} \tag{11.34}$$

$$\begin{aligned}
\gamma_{23} &= \bigcup_{x_2} [\mu_{A'_2}(x_2) \wedge \mu_{A_{23}}(x_2)] \\
&= \bigcup_{x_2} \left\{ [1 \quad 0.1 \quad 0 \quad 0] \right. \\
&\quad \left. \wedge [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.25 \quad 0.5 \quad 0.75 \quad 1] \right\} \\
&= \bigcup_{x_2} \left\{ [0 \quad 0 \quad 0] \right\} \\
&= 0
\end{aligned} \tag{11.35}$$

The two fuzzy outputs B'_1 and B'_2 are then calculated as

$$\begin{aligned}
\mu_{B'_1}(u_1) &= \bigcap_{k=1}^n \left\{ \bigcup_{i=1}^r [\gamma_{ki} \wedge \mu_{B_{1i}}(u_1)] \right\} \\
&= \left\{ [\gamma_{11} \wedge \mu_{B_{11}}(u_1)] \vee [\gamma_{12} \wedge \mu_{B_{12}}(u_1)] \vee [\gamma_{13} \wedge \mu_{B_{13}}(u_1)] \right\} \\
&\quad \wedge \left\{ [\gamma_{21} \wedge \mu_{B_{21}}(u_1)] \vee [\gamma_{22} \wedge \mu_{B_{22}}(u_1)] \vee [\gamma_{23} \wedge \mu_{B_{23}}(u_1)] \right\} \\
&= \left\{ [0.5 \quad 0.5 \quad 0.5 \quad 0.25 \quad 0 \quad 0] \right. \\
&\quad \left. \vee [0 \quad 0 \quad 0 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0] \right\} \\
&= \left\{ [0.5 \quad 0.5 \quad 0.5 \quad 0.25 \quad 0 \quad 0] \right. \\
&\quad \left. \wedge [0 \quad 0 \quad 0] \right\} \\
&= \left[0.5 \quad 0.5 \quad 0.5 \quad 0.25 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0 \right] \\
&\quad \wedge \left[1 \quad 0.75 \quad 0.5 \quad 0.25 \quad 0 \right] \\
&= \left[0.5 \quad 0.5 \quad 0.5 \quad 0.25 \quad 0 \right]
\end{aligned} \tag{11.36}$$

$$\begin{aligned}
\mu_{B'_2}(u_2) &= \bigcap_{k=1}^n \left\{ \bigcup_{i=1}^r [\gamma_{ki} \wedge \mu_{B_{2i}}(u_2)] \right\} \\
&= \left\{ [\gamma_{11} \wedge \mu_{B_{21}}(u_2)] \vee [\gamma_{12} \wedge \mu_{B_{22}}(u_2)] \vee [\gamma_{13} \wedge \mu_{B_{23}}(u_2)] \right\} \\
&\quad \wedge \left\{ [\gamma_{21} \wedge \mu_{B_{21}}(u_2)] \vee [\gamma_{22} \wedge \mu_{B_{22}}(u_2)] \vee [\gamma_{23} \wedge \mu_{B_{23}}(u_2)] \right\} \\
&= \left\{ [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.25 \quad 0.5 \quad 0.5 \quad 0.5] \right. \\
&\quad \left. \vee [0 \quad 0 \quad 0 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0 \quad 0 \quad 0] \right\} \\
&= \left\{ [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.25 \quad 0.5 \quad 0.75 \quad 1] \right. \\
&\quad \left. \wedge [0 \quad 0 \quad 0] \right\} \\
&= [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.25 \quad 0.5 \quad 0.5 \quad 0.5]
\end{aligned} \tag{11.37}$$

11.2.3 SIMPLIFICATION OF A MULTIDIMENSIONAL FUZZY RULE BASE

Consider a general multivariable fuzzy system with three inputs and two outputs as shown in Figure 11.8.

The corresponding fuzzy rules are in the form of

$$R^i: \text{IF } x_1 \text{ is } A_{1i} \text{ AND } x_2 \text{ is } A_{2i} \text{ AND } x_3 \text{ is } A_{3i}, \text{ THEN } u_1 \text{ is } B_{1i} \text{ AND } u_2 \text{ is } B_{2i} \quad (11.38)$$

where

R^i ($i = 1, \dots, r$) represents the i th fuzzy rule

x_k ($k = 1, 2, 3$) is the input linguistic variable

u_j ($j = 1, 2$) is the output linguistic variable

A_{ki} is the i th fuzzy value of the k th input variable defined in the universe of discourse X_k

B_{ji} is the i th fuzzy value of the j th output variable in the universe of discourse U_j

Each fuzzy rule R^i can be represented as a five-dimensional fuzzy relation as

$$R^i = (A_{1i} \times A_{2i} \times A_{3i}) \times (B_{1i} \times B_{2i}) \quad (11.39)$$

The overall multivariable fuzzy system is expressed as a combination of each individual rule as

$$\mu_R(x_1, x_2, x_3, u_1, u_2) = \bigcup_{i=1}^r [\mu_{A_{1i}}(x_1) \wedge \mu_{A_{2i}}(x_2) \wedge \mu_{A_{3i}}(x_3) \wedge \mu_{B_{1i}}(u_1) \wedge \mu_{B_{2i}}(u_2)] \quad (11.40)$$

Suppose the number of the fuzzy values assigned in the universes of discourse X_k and U_j are q_k and p_j , respectively. Then the resultant multivariable fuzzy system will contain a total of $q_1 \cdot q_2 \cdot q_3 \cdot p_1 \cdot p_2$ fuzzy rules. Given the fuzzy input variables A'_1 , A'_2 , and A'_3 , with min Cartesian product operation and Mamdani's min fuzzy implication, the resultant output fuzzy set B' in the Cartesian product space $U_1 \times U_2$ can be obtained from the following fuzzy inferencing calculation as

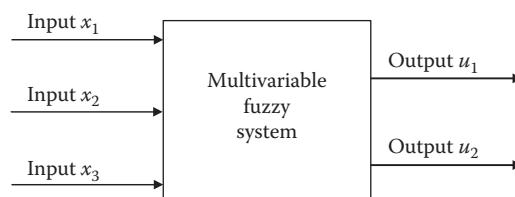


FIGURE 11.8 Three-input two-output fuzzy system.

$$\begin{aligned}
\mu_{B'}(u_1, u_2) &= [\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_2) \wedge \mu_{A'_3}(x_3)] \circ \mu_R(x_1, x_2, x_3, u_1, u_2) \\
&= \bigcup_{x_1, x_2, x_3} \{[\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_2) \wedge \mu_{A'_3}(x_3)] \wedge \mu_R(x_1, x_2, x_3, u_1, u_2)\} \\
&= \bigcup_{x_1} \left(\mu_{A'_1}(x_1) \wedge \bigcup_{x_2} \left\{ \mu_{A'_2}(x_2) \wedge \bigcup_{x_3} \left[\mu_{A'_3}(x_3) \wedge \bigcup_{i=1}^r \mu_{R^i}(x_1, x_2, x_3, u_1, u_2) \right] \right\} \right) \\
&= \bigcup_{x_1} \left[\mu_{A'_1}(x_1) \wedge \bigcup_{x_2} \left(\mu_{A'_2}(x_2) \wedge \bigcup_{x_3} \left\{ \mu_{A'_3}(x_3) \wedge \bigcup_{i=1}^r \right. \right. \right. \\
&\quad \times [\mu_{A_{1i}}(x_1) \wedge \mu_{A_{2i}}(x_2) \wedge \mu_{A_{3i}}(x_3) \wedge \mu_{B_{1i}}(u_1) \wedge \mu_{B_{2i}}(u_2)] \left. \right\} \left. \right) \left. \right] \\
&= \bigcup_{i=1}^r \bigcup_{x_1} \left[\mu_{A'_1}(x_1) \wedge \bigcup_{x_2} \left(\mu_{A'_2}(x_2) \wedge \bigcup_{x_3} \left\{ \mu_{A'_3}(x_3) \right. \right. \right. \\
&\quad \wedge [\mu_{A_{1i}}(x_1) \wedge \mu_{A_{2i}}(x_2) \wedge \mu_{A_{3i}}(x_3) \wedge \mu_{B_{1i}}(u_1) \wedge \mu_{B_{2i}}(u_2)] \left. \right\} \left. \right) \left. \right]
\end{aligned}$$

With the possible interaction effect existing among the three input variables,

$$\begin{aligned}
\mu_{B'}(u_1, u_2) &\subseteq \bigcup_{i=1}^r \left\{ \begin{array}{l} \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{1i}}(x_1)] \wedge \bigcup_{x_2} [\mu_{A'_2}(x_2) \wedge \mu_{A_{2i}}(x_2)] \wedge \bigcup_{x_3} [\mu_{A'_3}(x_3) \wedge \mu_{A_{3i}}(x_3)] \\ \wedge [\mu_{B_{1i}}(u_1) \wedge \mu_{B_{2i}}(u_2)] \end{array} \right\} \\
&= \bigcup_{i=1}^r \left(\begin{array}{l} \left\{ \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{1i}}(x_1)] \wedge \mu_{B_{1i}}(u_1) \right\} \wedge \left\{ \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{1i}}(x_1)] \wedge \mu_{B_{2i}}(u_2) \right\} \\ \wedge \left\{ \bigcup_{x_2} [\mu_{A'_2}(x_2) \wedge \mu_{A_{2i}}(x_2)] \wedge \mu_{B_{1i}}(u_1) \right\} \wedge \left\{ \bigcup_{x_2} [\mu_{A'_2}(x_2) \wedge \mu_{A_{2i}}(x_2)] \wedge \mu_{B_{2i}}(u_2) \right\} \\ \wedge \left\{ \bigcup_{x_3} [\mu_{A'_3}(x_3) \wedge \mu_{A_{3i}}(x_3)] \wedge \mu_{B_{1i}}(u_1) \right\} \wedge \left\{ \bigcup_{x_3} [\mu_{A'_3}(x_3) \wedge \mu_{A_{3i}}(x_3)] \wedge \mu_{B_{2i}}(u_2) \right\} \end{array} \right) \\
&= \bigcup_{i=1}^r \left\{ \begin{array}{l} \left\{ \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{1i}}(x_1)] \wedge \mu_{B_{1i}}(u_1) \right\} \wedge \bigcup_{i=1}^r \left\{ \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{A_{1i}}(x_1)] \wedge \mu_{B_{2i}}(u_2) \right\} \\ \wedge \bigcup_{i=1}^r \left\{ \bigcup_{x_2} [\mu_{A'_2}(x_2) \wedge \mu_{A_{2i}}(x_2)] \wedge \mu_{B_{1i}}(u_1) \right\} \wedge \bigcup_{i=1}^r \left\{ \bigcup_{x_2} [\mu_{A'_2}(x_2) \wedge \mu_{A_{2i}}(x_2)] \wedge \mu_{B_{2i}}(u_2) \right\} \\ \wedge \bigcup_{i=1}^r \left\{ \bigcup_{x_3} [\mu_{A'_3}(x_3) \wedge \mu_{A_{3i}}(x_3)] \wedge \mu_{B_{1i}}(u_1) \right\} \wedge \bigcup_{i=1}^r \left\{ \bigcup_{x_3} [\mu_{A'_3}(x_3) \wedge \mu_{A_{3i}}(x_3)] \wedge \mu_{B_{2i}}(u_2) \right\} \end{array} \right\} \\
&= \bigcup_{x_1} \left\{ \mu_{A'_1}(x_1) \wedge \bigcup_{i=1}^r [\mu_{A_{1i}}(x_1) \wedge \mu_{B_{1i}}(u_1)] \right\} \wedge \bigcup_{x_1} \left\{ \mu_{A'_1}(x_1) \wedge \bigcup_{i=1}^r [\mu_{A_{1i}}(x_1) \wedge \mu_{B_{2i}}(u_2)] \right\} \\
&\quad \wedge \bigcup_{x_2} \left\{ \mu_{A'_2}(x_2) \wedge \bigcup_{i=1}^r [\mu_{A_{2i}}(x_2) \wedge \mu_{B_{1i}}(u_1)] \right\} \wedge \bigcup_{x_2} \left\{ \mu_{A'_2}(x_2) \wedge \bigcup_{i=1}^r [\mu_{A_{2i}}(x_2) \wedge \mu_{B_{2i}}(u_2)] \right\} \\
&\quad \wedge \bigcup_{x_3} \left\{ \mu_{A'_3}(x_3) \wedge \bigcup_{i=1}^r [\mu_{A_{3i}}(x_3) \wedge \mu_{B_{1i}}(u_1)] \right\} \wedge \bigcup_{x_3} \left\{ \mu_{A'_3}(x_3) \wedge \bigcup_{i=1}^r [\mu_{A_{3i}}(x_3) \wedge \mu_{B_{2i}}(u_2)] \right\} \\
&= \bigcup_{x_1} \left\{ \mu_{A'_1}(x_1) \wedge \bigcup_{i=1}^r [\mu_{A_{1i}}(x_1) \wedge \mu_{B_{1i}}(u_1)] \right\} \wedge \bigcup_{x_1} \left\{ \mu_{A'_1}(x_1) \wedge \bigcup_{i=1}^r [\mu_{A_{1i}}(x_1) \wedge \mu_{B_{2i}}(u_2)] \right\} \\
&\quad \wedge \bigcup_{x_2} \left\{ \mu_{A'_2}(x_2) \wedge \bigcup_{i=1}^r [\mu_{A_{2i}}(x_2) \wedge \mu_{B_{1i}}(u_1)] \right\} \wedge \bigcup_{x_2} \left\{ \mu_{A'_2}(x_2) \wedge \bigcup_{i=1}^r [\mu_{A_{2i}}(x_2) \wedge \mu_{B_{2i}}(u_2)] \right\} \\
&\quad \wedge \bigcup_{x_3} \left\{ \mu_{A'_3}(x_3) \wedge \bigcup_{i=1}^r [\mu_{A_{3i}}(x_3) \wedge \mu_{B_{1i}}(u_1)] \right\} \wedge \bigcup_{x_3} \left\{ \mu_{A'_3}(x_3) \wedge \bigcup_{i=1}^r [\mu_{A_{3i}}(x_3) \wedge \mu_{B_{2i}}(u_2)] \right\}
\end{aligned} \tag{11.41}$$

Define a two-dimensional fuzzy relation R_{kj} as

$$R_{kj} = A_{ki} \times B_{ji} \quad k = 1, 2, 3 \quad j = 1, 2$$

$$\text{where } \mu_{R_{kj}}(x_k, u_j) = \bigcup_{i=1}^r [\mu_{A_{ki}}(x_k) \wedge \mu_{B_{ji}}(u_j)] \quad (11.42)$$

Then,

$$\begin{aligned} \mu_{B'}(u_1, u_2) \subseteq & \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{R_{11}}(x_1, u_1)] \wedge \bigcup_{x_1} [\mu_{A'_1}(x_1) \wedge \mu_{R_{12}}(x_1, u_2)] \\ & \wedge \bigcup_{x_2} [\mu_{A'_2}(x_2) \wedge \mu_{R_{21}}(x_2, u_1)] \wedge \bigcup_{x_2} [\mu_{A'_2}(x_2) \wedge \mu_{R_{22}}(x_2, u_2)] \\ & \wedge \bigcup_{x_3} [\mu_{A'_3}(x_3) \wedge \mu_{R_{31}}(x_3, u_1)] \wedge \bigcup_{x_3} [\mu_{A'_3}(x_3) \wedge \mu_{R_{32}}(x_3, u_2)] \end{aligned} \quad (11.43)$$

By neglecting the cross-coupling effect among the multiple antecedent variables, the multidimensional fuzzy rule base will be simplified as

$$B' = A'_1 \circ R_{11} \wedge A'_1 \circ R_{12} \wedge A'_2 \circ R_{21} \wedge A'_2 \circ R_{22} \wedge A'_3 \circ R_{31} \wedge A'_3 \circ R_{32} \quad (11.44)$$

And then each individual output can be calculated by the projection of the compound fuzzy set B' on the respective universe of discourse as

$$\begin{aligned} \mu_{B'_1}(u_1) &= \max_{u_2} \mu_{B'}(u_1, u_2) \\ \mu_{B'_2}(u_2) &= \max_{u_1} \mu_{B'}(u_1, u_2) \end{aligned} \quad (11.45)$$

Then we have

$$\begin{aligned} B'_1 &= A'_1 \circ R_{11} \wedge A'_2 \circ R_{21} \wedge A'_3 \circ R_{31} \\ B'_2 &= A'_1 \circ R_{12} \wedge A'_2 \circ R_{22} \wedge A'_3 \circ R_{32} \end{aligned} \quad (11.46)$$

From the derivation, the complex five-dimensional fuzzy relation R^i is simplified as the combination of multiple two-dimensional fuzzy relations R_{kj} , which contains $q_k \cdot p_j$ fuzzy rules. The computational load is reduced consequently for the multivariable fuzzy rule base.

Defining $*$ as the (\circ, \wedge) -operator, this multivariable fuzzy system can be expressed in a compact form as

$$B'_c = A'_c * R_c \quad (11.47)$$

where

$$B'_c = \begin{bmatrix} B'_1 \\ B'_2 \end{bmatrix}$$

$$A'_c = [A'_1 \ A'_2 \ A'_3]$$

$$R_c = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \\ R_{31} & R_{32} \end{bmatrix}$$

To better understand the nature of the linguistic description (Equation 11.38) and its mathematical counterpart (Equation 11.46), a symbolic block diagram is introduced for this multidimensional fuzzy rule base as shown in Figure 11.9, which consists of input signals, branch points, functional blocks, inner signals, intersection blocks, and output signals. The functional blocks are composed of the max-min composition \circ and the fuzzy relation R_{kj} , which are further linked up by the intersection \wedge operators. The arrows indicate the direction of signal flows from input variables to output variables. The inner signals and the output signals are of the form

$$\begin{aligned} B'_{kj} &= A'_k \circ R_{kj} \quad k = 1, 2, 3 \quad j = 1, 2 \\ B'_j &= \bigcap_{k=1}^3 B'_{kj} \quad j = 1, 2 \end{aligned} \quad (11.48)$$

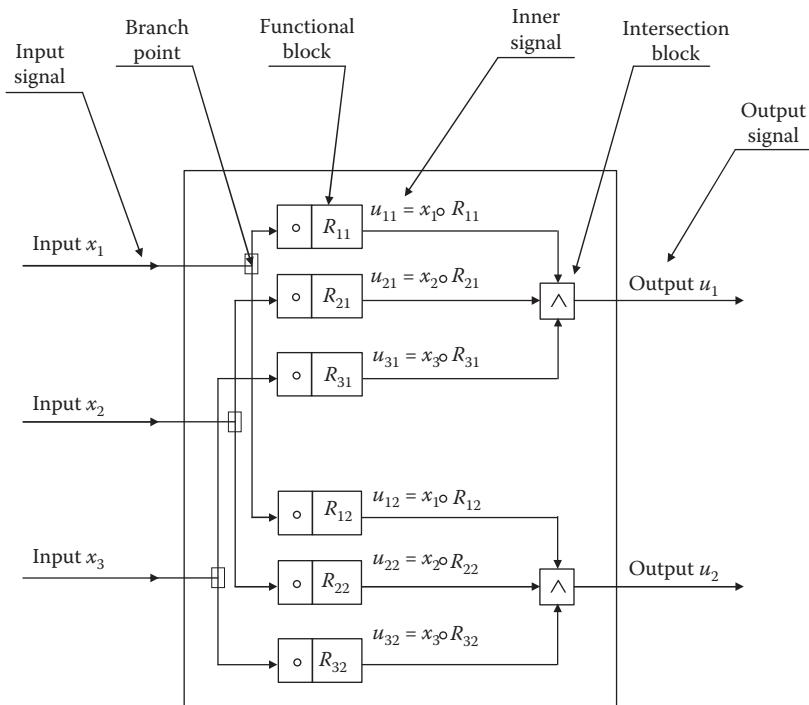


FIGURE 11.9 Simplification of a multidimensional fuzzy rule base. (From Gupta, M.M., Kiszka, J.B., and Trojan, G.M., *IEEE Trans. Syst. Man Cybern.*, SMC-16, 638, 1986. With permission.)

Therefore, the simplification of a general multivariable fuzzy system with n inputs and p outputs is described as

$$\begin{aligned} B'_{kj} &= A'_k \circ R_{kj} \quad k = 1, \dots, n \quad j = 1, \dots, p \\ B'_j &= \bigcap_{k=1}^n B'_{kj} \quad j = 1, \dots, p \end{aligned} \quad (11.49)$$

Example 11.4 Simplification of a Four-Input Two-Output Fuzzy Control Rule Base (Gupta, 1986)

Consider an interconnected tank system with controlled volumetric flow supplies as shown in Figure 11.10, where the tank levels and the liquid flow rates are viewed as the system input variables, and the rates of the tank level changes are the system outputs. The schematic structure for this four-input two-output system is shown in Figure 11.11.

The fuzzy system model is established from human operator's observation and the corresponding fuzzy rules are constructed as

- R^1 : IF $x_1 = 0$ AND $x_2 = 0$ AND $u_1 = 0$ AND $u_2 = 0$,
THEN $\Delta x_1 = 0$ AND $\Delta x_2 = 0$
- R^2 : IF $x_1 = 0$ AND $x_2 = 0$ AND $u_1 = \text{small}$ AND $u_2 = \text{big}$,
THEN $\Delta x_1 = \text{below positive}$ AND $\Delta x_2 = \text{positive}$
- R^3 : IF $x_1 = \text{small}$ AND $x_2 = \text{small}$ AND $u_1 = \text{big}$ AND $u_2 = \text{medium}$,
THEN $\Delta x_1 = \text{positive}$ AND $\Delta x_2 = \text{below positive}$
- R^4 : IF $x_1 = \text{between small and medium}$ AND $x_2 = \text{small}$ AND
 $u_1 = \text{between small and medium}$ AND $u_2 = \text{medium}$,
THEN $\Delta x_1 = 0$ AND $\Delta x_2 = 0$

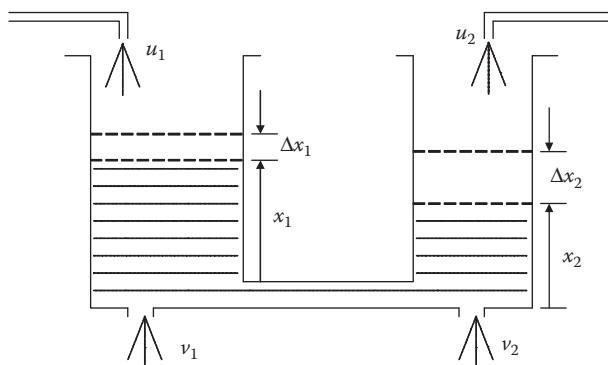


FIGURE 11.10 Interconnected tank system. (From Gupta, M.M., Kisza, J.B., and Trojan, G.M., *IEEE Trans. Syst. Man Cybern.*, SMC-16, 638, 1986. With permission.)

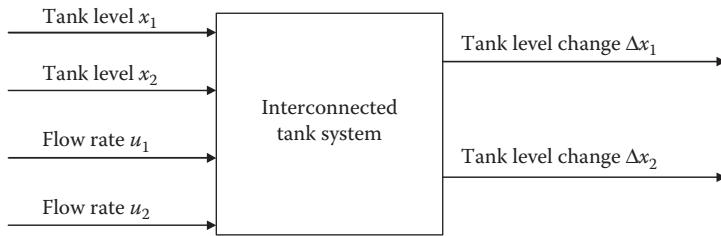


FIGURE 11.11 Four-input two-output tank system. (From Gupta, M.M., Kiszka, J.B., and Trojan, G.M., *IEEE Trans. Syst. Man Cybern.*, SMC-16, 638, 1986. With permission.)

R^5 : IF $x_1 = \text{medium}$ AND $x_2 = \text{medium}$ AND
 $u_1 = \text{between zero and small}$ AND $u_2 = \text{medium}$,
THEN $\Delta x_1 = \text{negative}$ AND $\Delta x_2 = 0$

R^6 : IF $x_1 = \text{medium}$ AND $x_2 = \text{medium}$ AND $u_1 = 0$ AND $u_2 = \text{big}$,
THEN $\Delta x_1 = \text{negative}$ AND $\Delta x_2 = 0$

R^7 : IF $x_1 = \text{between small and medium}$ AND $x_2 = \text{medium}$ AND
 $u_1 = \text{medium}$ AND $u_2 = \text{big}$,
THEN $\Delta x_1 = \text{below positive}$ AND $\Delta x_2 = 0$

R^8 : IF $x_1 = \text{medium}$ AND $x_2 = \text{between medium and big}$ AND
 $u_1 = \text{big}$ AND $u_2 = \text{medium}$,
THEN $\Delta x_1 = 0$ AND $\Delta x_2 = 0$

R^9 : IF $x_1 = \text{big}$ AND $x_2 = \text{big}$ AND
 $u_1 = \text{between zero and small}$ AND $u_2 = \text{between 0 and small}$,
THEN $\Delta x_1 = \text{negative}$ AND $\Delta x_2 = \text{negative}$

R^{10} : IF $x_1 = \text{medium}$ AND $x_2 = \text{medium}$ AND $u_1 = \text{small}$ AND $u_2 = 0$,
THEN $\Delta x_1 = 0$ AND $\Delta x_2 = \text{above negative}$

R^{11} : IF $x_1 = \text{between small and medium}$ AND $x_2 = \text{small}$ AND
 $u_1 = \text{between zero and small}$ AND $u_2 = 0$,
THEN $\Delta x_1 = \text{negative}$ AND $\Delta x_2 = \text{above negative}$

R^{12} : IF $x_1 = \text{small}$ AND $x_2 = \text{small}$ AND $u_1 = 0$ AND $u_2 = 0$,
THEN $\Delta x_1 = \text{negative}$ AND $\Delta x_2 = \text{negative}$

R^{13} : IF $x_1 = \text{above zero}$ AND $x_2 = \text{above zero}$ AND $u_1 = \text{big}$ AND $u_2 = \text{big}$,
THEN $\Delta x_1 = \text{positive}$ AND $\Delta x_2 = \text{positive}$

R^{14} : IF $x_1 = \text{between small and medium}$ AND $x_2 = \text{between small and medium}$ AND
 $u_1 = \text{below big}$ AND $u_2 = \text{medium}$,
THEN $\Delta x_1 = \text{below positive}$ AND $\Delta x_2 = \text{below positive}$

TABLE 11.2
Fuzzy Sets for the Tank Levels x_1, x_2

Universe	Z	ZS	S	SM	M	MB	B	AS	BZS	BS	BSM
1	1	0.5	0	0	0	0	0	0	1	0.5	0
2	0.5	1	0.5	0	0	0	0	0	1	1	0
3	0	0.5	1	0.5	0	0	0	0.5	0.5	1	1
4	0	0	0.5	1	0.5	0	0	1	0	0	1
5	0	0	0	0.5	1	0.5	0	1	0	0	0.5
6	0	0	0	0	0.5	1	0.5	0	0	0	0
7	0	0	0	0	0	0.5	1	0	0	0	0

Source: Gupta, M.M., Kiszka, J.B., and Trojan, G.M., *IEEE Trans. Syst. Man Cybern.*, SMC-16, 638, 1986. With permission.

Note: Z, zero; ZS, between zero and small; S, small; SM, between small and medium; M, medium; MB, between medium and big; B, big; AS, above small; BZS, between zero and small; BS, below small; BSM, below small and medium.

The subjective definitions of the fuzzy sets for the tank levels x_1, x_2 ; the flow rates u_1, u_2 ; and the rates of the tank level changes $\Delta x_1, \Delta x_2$ are shown in Tables 11.2 through 11.4, respectively.

Based on the procedure explanation of the simplification for the multidimensional fuzzy rule base, the individual two-dimensional fuzzy relation R_{kj} in this system can be formulated as

TABLE 11.3
Fuzzy Sets for the Flow Rates u_1, u_2

Universe	Z	ZS	S	SM	M	B	AS	BSM	BB
1	1	0.5	0	0	0	0	0	0	0
2	0.5	1	0.5	0	0	0	0	0	0
3	0	0.5	1	0.5	0	0	0.5	1	0
4	0	0	0.5	1	0.5	0	1	1	0
5	0	0	0	0.5	1	0	1	0.5	0.5
6	0	0	0	0	0.5	0.5	0	0	1
7	0	0	0	0	0	1	0	0	1

Source: Gupta, M.M., Kiszka, J.B., and Trojan, G.M., *IEEE Trans. Syst. Man Cybern.*, SMC-16, 638, 1986. With permission.

Note: Z, zero; ZS, between zero and small; S, small; SM, between small and medium; M, medium; B, big; AS, above small; BSM, below small and medium; BB, below big.

TABLE 11.4
Fuzzy Sets for the Rates of the Tank Level
Changes $\Delta x_1, \Delta x_2$

Universe	N	AN	Z	BP	P
-3	1	1	0	0	0
-2	1	1	0.5	0	0
-1	0.5	1	1	0	0
0	0	0	1	0	0
1	0	0	1	1	0.5
2	0	0	0.5	1	1
3	0	0	0	1	1

Source: Gupta, M.M., Kiszka, J.B., and Trojan, G.M., *IEEE Trans. Syst. Man Cybern.*, SMC-16, 638, 1986. With permission.

Note: N, negative; AN, above negative; Z, zero; BP, below positive; P, positive.

$$\begin{aligned}
 \mu_{R_{11}}(x_1, \Delta x_1) &= \bigcup_{i=1}^{14} [\mu_i(x_1) \wedge \mu_i(\Delta x_1)] & \mu_{R_{12}}(x_1, \Delta x_2) &= \bigcup_{i=1}^{14} [\mu_i(x_1) \wedge \mu_i(\Delta x_2)] \\
 \mu_{R_{21}}(x_2, \Delta x_1) &= \bigcup_{i=1}^{14} [\mu_i(x_2) \wedge \mu_i(\Delta x_1)] & \mu_{R_{22}}(x_2, \Delta x_2) &= \bigcup_{i=1}^{14} [\mu_i(x_2) \wedge \mu_i(\Delta x_2)] \\
 \mu_{R_{31}}(u_1, \Delta x_1) &= \bigcup_{i=1}^{14} [\mu_i(u_1) \wedge \mu_i(\Delta x_1)] & \mu_{R_{32}}(u_1, \Delta x_2) &= \bigcup_{i=1}^{14} [\mu_i(u_1) \wedge \mu_i(\Delta x_2)] \\
 \mu_{R_{41}}(u_2, \Delta x_1) &= \bigcup_{i=1}^{14} [\mu_i(u_2) \wedge \mu_i(\Delta x_1)] & \mu_{R_{42}}(u_2, \Delta x_2) &= \bigcup_{i=1}^{14} [\mu_i(u_2) \wedge \mu_i(\Delta x_2)]
 \end{aligned} \tag{11.50}$$

where

$$\begin{aligned}
 \mu_1(x_1) \wedge \mu_1(\Delta x_1) &= \left[\begin{array}{ccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \wedge \left[\begin{array}{ccccccc} 1 & 1 & 0.5 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0.5 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0.5 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0.5 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0.5 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0.5 & 0 & 0 & 0 & 0 \end{array} \right] \\
 &= \left[\begin{array}{ccccccc} 1 & 1 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right]
 \end{aligned}$$

$$\mu_2(x_1) \wedge \mu_2(\Delta x_1) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \wedge \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

⋮

$$\mu_{14}(x_1) \wedge \mu_{14}(\Delta x_1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\wedge \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

And then we can obtain

$$\begin{aligned}
 R_{11} &= \begin{bmatrix} 0 & 0.5 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 1 & 1 \\ 1 & 1 & 0.5 & 0.5 & 0.5 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 1 & 1 & 0.5 & 0 & 0 & 0 & 0 \end{bmatrix} \quad R_{12} = \begin{bmatrix} 0 & 0.5 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 1 & 1 \\ 1 & 1 & 0.5 & 0.5 & 0.5 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 1 & 1 & 0.5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 R_{21} &= \begin{bmatrix} 0 & 0.5 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 1 & 1 & 1 & 0.5 & 0.5 \\ 1 & 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \end{bmatrix} \quad R_{22} = \begin{bmatrix} 0 & 0.5 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0.5 & 0.5 \\ 0.5 & 0.5 & 1 & 1 & 1 & 0.5 & 0 \\ 1 & 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \\
 R_{31} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0.5 & 0 \\ 1 & 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0.5 & 1 & 1 & 1 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 & 0.5 & 1 & 1 & 1 \\ 0 & 0.5 & 0.5 & 0.5 & 1 & 1 & 1 \\ 0 & 0.5 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad R_{32} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0.5 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0.5 & 0.5 \\ 1 & 1 & 1 & 1 & 0.5 & 0.5 & 1 & 1 \\ 0.5 & 0.5 & 1 & 1 & 1 & 1 & 0.5 & 0.5 \\ 0 & 0.5 & 1 & 1 & 1 & 1 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 & 0.5 & 1 & 1 & 1 & 1 \\ 0 & 0.5 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\
 R_{41} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0.5 & 0 \\ 1 & 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 1 & 1 & 0.5 & 0 & 1 & 1 & 1 \end{bmatrix} \quad R_{42} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0.5 & 0 \\ 1 & 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0.5 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0.5 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}
 \end{aligned}$$

Suppose currently the four input variables are given as $X'_1 = \text{big}$, $X'_2 = \text{small}$, $U'_1 = \text{between zero and small}$, and $U'_2 = 0$. Then the corresponding two output fuzzy sets can be expressed in a compact form as

$$\begin{bmatrix} \Delta X'_1 \\ \Delta X'_2 \end{bmatrix} = [X'_1 \ X'_2 \ U'_1 \ U'_2] * \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \\ R_{31} & R_{32} \\ R_{41} & R_{42} \end{bmatrix} \quad (11.51)$$

where each component is determined as

$$\begin{aligned}
 \Delta X'_1 &= X'_1 \circ R_{11} \wedge X'_2 \circ R_{21} \wedge U'_1 \circ R_{31} \wedge U'_2 \circ R_{41} \\
 &= [1 \ 1 \ 0.5 \ 0.5 \ 0.5 \ 0.5 \ 0] \wedge [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1] \\
 &\quad \wedge [1 \ 1 \ 0.5 \ 0.5 \ 0.5 \ 0.5 \ 0.5] \wedge [1 \ 1 \ 1 \ 1 \ 1 \ 0.5 \ 0] \\
 &= [1 \ 1 \ 0.5 \ 0.5 \ 0.5 \ 0.5 \ 0] \\
 &\approx \text{negative} \quad (11.52)
 \end{aligned}$$

$$\begin{aligned}
\Delta X'_2 &= X'_1 \circ R_{12} \wedge X'_2 \circ R_{22} \wedge U'_1 \circ R_{32} \wedge U'_2 \circ R_{42} \\
&= [1 \ 1 \ 0.5 \ 0.5 \ 0.5 \ 0.5 \ 0] \wedge [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1] \\
&\quad \wedge [1 \ 1 \ 1 \ 1 \ 0.5 \ 0.5] \wedge [1 \ 1 \ 1 \ 1 \ 1 \ 0.5 \ 0] \\
&= [1 \ 1 \ 0.5 \ 0.5 \ 0.5 \ 0.5 \ 0] \\
&\approx \text{negative}
\end{aligned} \tag{11.53}$$

11.3 MULTIVARIABLE FUZZY CONTROLLER TO DEAL WITH THE CROSS-COUPING EFFECT

Many complex industrial systems are multivariable nonlinear and time-varying, where accurate mathematical models are often difficult to obtain. Intelligent fuzzy control systems are more and more frequently designed for such applications due to its particular advantages over conventional control techniques in dealing with complex systems without precise analytical models. Automotive and chemical industries are two of such typical application areas. When developing a multivariable fuzzy controller for these systems, it is sometimes difficult to infer the proper commands for each control variable considering the high dimension of the established control rule base, which brings up computational complexity in actual applications. Various multivariable fuzzy control structures have been proposed to improve the real-time calculation efficiency.

11.3.1 MIXED FUZZY CONTROLLER

A general MIMO system often has complicated dynamic coupling behavior. It is usually difficult to establish an accurate dynamic model and decouple it along each degree of freedom for controller design. Hence, a mixed fuzzy controller was proposed by Lian and Huang (2001) based on the physical analysis of the controlled process, which is composed of a set of regular fuzzy controllers and decoupling fuzzy controllers. Each regular fuzzy controller is designed for each degree of freedom of the multivariable system. And then, appropriate decoupling fuzzy controllers are designed to compensate for the coupling effects of system dynamics among each degree of freedom. The overall fuzzy control system is illustrated in [Figure 11.12](#).

Step 1: Regular Fuzzy Control Design

The inputs to each regular fuzzy controller are system output error and the change of error along each degree of freedom as

$$e_i(k) = y_{di}(k) - y_i(k) \tag{11.54a}$$

$$\Delta e_i(k) = e_i(k) - e_i(k - 1) \tag{11.54b}$$

where

$e_i(k)$ represents the error signal at the sampling time k along the i th degree of freedom

$\Delta e_i(k)$ is the change of error along the i th degree of freedom

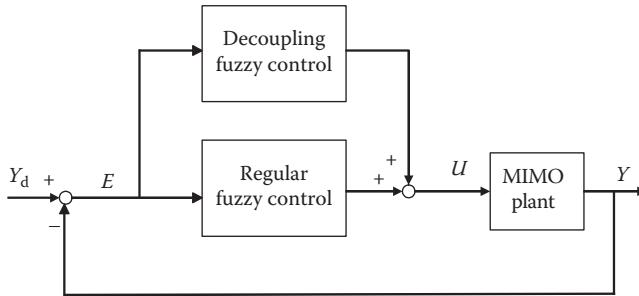


FIGURE 11.12 Mixed fuzzy control system. (From Lian, R.-J. and Huang, S.-J., *Fuzzy Sets Syst.*, 120, 73, 2001. With permission.)

$y_{di}(k)$ is the desired output of the i th degree of freedom
 $y_i(k)$ is the actual output of the i th degree of freedom

The fuzzy rule base of the regular fuzzy controller is expressed in the following form as

$$R^l: \text{IF } e_i(k) = E_i^l \text{ AND } \Delta e_i(k) = CE_i^l, \text{ THEN } \Delta u_i(k) = CU_i^l \quad (11.55)$$

where

R^l ($l = 1, 2, \dots, L$) denotes the l th rule, L is the total number of fuzzy rules in the control rule base

$\Delta u_i(k)$ is the change of control action for the i th degree of freedom

E_i^l , CE_i^l , and CU_i^l are the linguistic variables for $e_i(k)$, $\Delta e_i(k)$, and $\Delta u_i(k)$ in the l th rule, respectively

The final control action $u_i(k)$ is then updated incrementally during each sampling period as

$$u_i(k) = u_i(k - 1) + \Delta u_i(k) \quad (11.56)$$

Step 2: Decoupling Fuzzy Control Design

For a typical MIMO system, the system output is often influenced by more than one system input due to cross-coupled system dynamic characteristics such that the multiple system inputs and multiple system outputs interact with each other. For such a multi-variable system with n inputs and p outputs, the interaction effect may be expressed as

$$y_i = P_i(u_1, \dots, u_n, y_1, \dots, y_p) \quad (11.57a)$$

$$y_j = P_j(u_1, \dots, u_n, y_1, \dots, y_p) \quad (11.57b)$$

where $P_i(\cdot)$ and $P_j(\cdot)$ are nonlinear functions representing the multivariable interactions. Thus, for the i th output y_i , the input u_i can be viewed to have the main effect on y_i and the other inputs have secondary effects. Similarly, for the j th output y_j , u_j has the main effect and the other inputs hold secondary effects.

Therefore, the overall design idea of mixed fuzzy control is formulated such that the main effect of the multivariable interaction is addressed by a regular fuzzy controller and the secondary effect is solved by designing an appropriate decoupling fuzzy controller. The fuzzy rule base of the decoupling fuzzy controller from the j th output y_j to the i th input u_i can be represented as

$$\text{DR}^l: \text{IF } e_j(k) = E_j^l \text{ AND } \Delta e_j(k) = \text{CE}_j^l, \text{ THEN } u_{j \rightarrow i}(k) = U_{j \rightarrow i}^l \quad (11.58)$$

where

DR^l ($l = 1, 2, \dots, L$) denotes the l th decoupling fuzzy control rule

$u_{j \rightarrow i}(k)$ is the control action calculated from the output y_j to the input u_i through the decoupling fuzzy control inferencing

In every sampling time, the control output obtained from the decoupling fuzzy controller is an absolute control action instead of an incremental control action. The main reason is that the coupling effect may change during each sampling interval so it does not have an accumulating feature. E_j^l , CE_j^l , and $U_{j \rightarrow i}^l$ are the linguistic variables for $e_j(k)$, $\Delta e_j(k)$, and $u_{j \rightarrow i}(k)$ in the l th fuzzy rule, respectively. And then the decoupling effect is incorporated into the regular fuzzy controller to compensate for the multivariable interaction and the final control action $U_i(k)$ is expressed as

$$U_i(k) = u_i(k) + \sum_{j \neq i} u_{j \rightarrow i}(k) \quad (11.59)$$

For a general multivariable system with two inputs and two outputs, the corresponding block diagram of the overall mixed fuzzy control system is illustrated in [Figure 11.13](#).

11.3.2 MULTIOBJECTIVE FUZZY CONTROLLER

For a system with multiple control objectives, a multivariable coordinating control structure was proposed by Lim and Bien (1996). The overall fuzzy control system consists of a supervisory controller and many subfuzzy controllers. Each subcontroller is a typical fuzzy controller responsible for a particular control objective and the supervisory controller coordinates the multiple subfuzzy controllers by adjusting the individual weight to each control objective.

Consider a general nonlinear system with p outputs in the state-space form as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u(t)) \quad (11.60a)$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), u(t)) \quad (11.60b)$$

where

$\mathbf{x}(t) = [x_1(t), \dots, x_n(t)]^T \in \Re^n$ is an n -dimensional state vector

$\mathbf{x}(0)$ is the state vector at the initial time

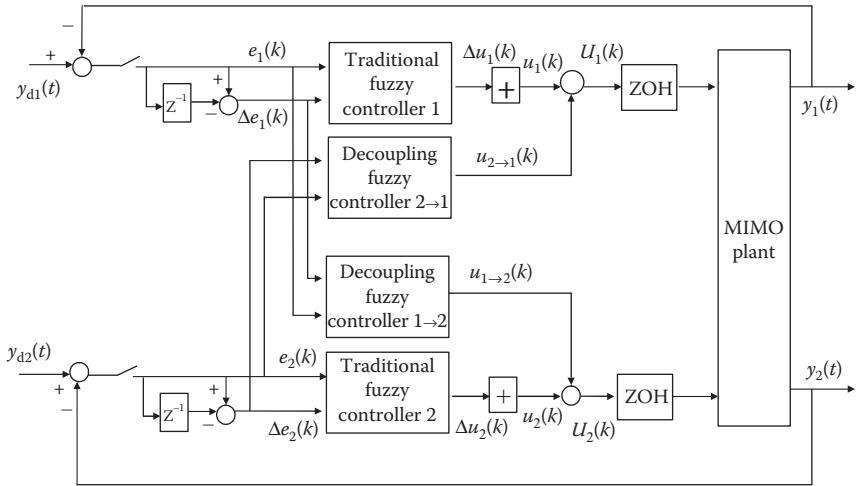


FIGURE 11.13 Mixed fuzzy control system for a two-input two-output plant. (From Lian, R.-J. and Huang, S.-J., *Fuzzy Sets Syst.*, 120, 73, 2001. With permission.)

$\mathbf{y}(t) = [y_1(t), \dots, y_p(t)]^T \in \mathbb{R}^p$ is a p -dimensional system output vector
 $u(t)$ is the scalar input

$f(\cdot): \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$ and $g(\cdot): \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^p$ are smooth functions representing
the system's nonlinear dynamics, respectively

Suppose there are a total of M control objectives for this system to satisfy:

$$Q_i = q_i(\mathbf{x}(t), u(t)) \quad i = 1, \dots, M \quad (11.61)$$

where $q_i(\cdot)$ denotes a certain function of $\mathbf{x}(t)$ and $u(t)$. Then, the control problem may be formulated as a multiobjective optimization problem to find a suitable control action $u(t)$, which maximizes the M control objective functions Q_i ($i = 1, \dots, M$) in Equation 11.61 within the constraints of Equations 11.60. Furthermore, a concept of a satisfaction degree for a particular control objective is proposed and defined as the value of a criterion representing the degree of how much the particular control objective is satisfied. Denote the satisfaction degree of the i th control objective as P_i , where $P_i \in [0, 1]$. The value of zero represents that the control objective is not achieved at all and the value of unit corresponds to the perfectly matched control objective.

The overall multiobjective fuzzy controller can be illustrated in Figure 11.14. During the multiobjective fuzzy control design process, the multiple subfuzzy controllers are first designed independently for the M control objectives such that each of subcontroller maximizes the individual satisfaction degree P_i . Next, the respective weight factor for each subfuzzy controller needs to be determined by the supervisory controller, where the fuzzy rule base is composed of the following linguistic description:

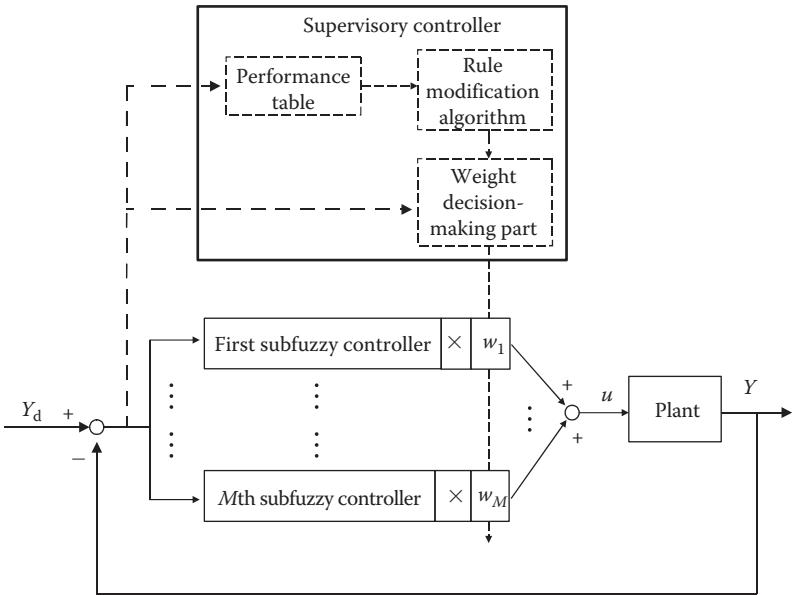


FIGURE 11.14 Multiobjective fuzzy control system. (From Lim, T. and Bien, Z., *Int. J. Appl. Mat. Comput. Sci.*, 6, 565, 1996. With permission.)

$$\begin{aligned}
 R^l: & \text{ IF } y_1(t) = Y_1^l \text{ AND } \dots y_k(t) = Y_k^l \dots \text{ AND } y_p(t) = Y_p^l, \\
 & \text{ THEN } w_1(t) = W_1^l \text{ AND } \dots w_i(t) = W_i^l \dots \text{ AND } w_M(t) = W_M^l
 \end{aligned} \quad (11.62)$$

where

R^l ($l = 1, 2, \dots, L$) denotes the l th rule, L is the number of fuzzy rules

$y_k(t)$ is the k th system output

$w_i(t)$ is the weight for the i th subfuzzy controller

Y_k^l and W_i^l are the linguistic variables for $y_k(t)$ and $w_i(t)$ in the l th rule, respectively

The final control action $u(t)$ of the overall multiobjective fuzzy controller is a weighted sum of the M subcontrol actions $u_i(t)$ as

$$u(t) = \sum_{i=1}^M w_i(t) \cdot u_i(t) \quad (11.63)$$

It is worth to note the difficulty in determining the optimal MFs of the fuzzy rule base in the supervisory controller during the weight decision-making process. Therefore, a self-organizing mechanism is adopted in the supervisory controller during the learning process to modify the MFs and optimize the overall system performance. As shown in Figure 11.14, the performance table and the rule modification algorithm

in the supervisory controller constitute the self-organizing controller (SOC), where the learning procedure is described as below:

1. The performance table evaluates the control performance according to the satisfaction degree of each control objective.
2. The following condition on the satisfaction degree is checked

$$\min_{i=1,\dots,M} P_i \geq \alpha \quad (11.64)$$

where $\alpha \in [0, 1]$ is the least satisfaction limit determined by the controller designer. If the condition in Equation 11.64 is satisfied, that is, if the current multiobjective fuzzy controller achieves all control objectives with respect to each satisfaction degree, the self-organizing learning procedure is stopped. Otherwise, the next step will proceed.

3. This step determines how much the output MF in the supervisory controller needs to be modified. Assume the output MFs W_i^l are isosceles triangles within the universe of discourse $[0, 1]$, whose center is located at λ_{il} . Denote $P_a = \frac{\sum_{i=1}^M P_i}{M}$, which provides a reference value to determine which control objective requires modification. If $P_a \leq P_i$, then λ_{il} needs to move to a larger value. Otherwise, λ_{il} should move to a smaller value. The amount of change in λ_{il} is determined as

$$\delta_i = \eta \cdot \frac{P_a - P_i}{\sum_{i=1}^M |P_a - P_i|} \quad (11.65)$$

where η is the learning rate determined by the controller designer.

4. The rule modification algorithm utilizes δ_i to update the center of the MFs λ_{il} as

$$\lambda_{il,\text{new}} = \lambda_{il} + \delta_i \quad (11.66)$$

In this manner, the satisfaction degree is improved by modifying the output MF in the supervisory controller for each control objective. The learning process of the SOC is updated repetitively until the condition in Equation 11.64 is satisfied for the M control objectives.

11.4 CONCLUSION

In the aforementioned multivariable fuzzy control strategies, the fuzzy control rules are generated either based on the knowledge of human operators or the qualitative analysis of the multivariable systems to be controlled. Due to the possible presence of the nonlinearity and uncertainties, these fuzzy control rules may mistakenly interpret the input–output relationship and thus deteriorate the control performance.

Hence, a model-based multivariable fuzzy controller would be desirable to increase the accuracy of the established fuzzy rules and improve the overall system performance, which will be explained in detail in the next chapter.

REFERENCES

- Cheng, W.-M., Jen, S.-C., Wu, C.-F., and Tsuei, T.-H., The intersection of fuzzy subsets and the robustness of fuzzy control, *Proceedings International Conference on Cybernetics and Society*, Cambridge, MA, pp. 826–829, 1982a.
- Cheng, W.-M., Ren, S.-J., Wu, C.-F., and Tsuei, T.-H., An expression for fuzzy controller, *Fuzzy Information and Decision Processes*, Gupta, M.M. and Sanchez, E. (Eds.), North-Holland Publishing Company, Amsterdam, the Netherlands, pp. 411–413, 1982b.
- Gegov, A.E. and Frank, P.M., Reduction of multidimensional relations in fuzzy control systems, *Systems and Control Letters*, 25: 307–313, 1995a.
- Gegov, A.E. and Frank, P.M., Decentralized fuzzy control of multivariable system by active decomposition of control laws, *International Journal of Control*, 62: 781–798, 1995b.
- Gegov, A.E., *Distributed Fuzzy Control of Multivariable Systems*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1996.
- Gupta, M.M., Kiszka, J.B., and Trojan, G.M., Multivariable structure of fuzzy control systems, *IEEE Transactions on Systems, Man and Cybernetics*, SMC-16(5): 638–656, 1986.
- Lian, R.-J. and Huang, S.-J., A mixed fuzzy controller for MIMO systems, *Fuzzy Sets and Systems*, 120: 73–93, 2001.
- Lim, T. and Bien, Z., FLC design for multi-objective systems, *International Journal of Applied Mathematics and Computer Science*, 6(3): 565–580, 1996.
- Raju, G.V.S., Zhou, J., and Kisner, R.A., Hierarchical fuzzy control. *International Journal of Control*, 54(5): 1201–1216, 1991.
- Song, B.G., Marks II, R.J., Oh, S., Arabshahi, P., Caudell, T.P., and Choi, J.J., Adaptive membership function fusion and annihilation in fuzzy IF-THEN rules, *Proceedings of FUZZ-IEEE/IFES'93*, San Francisco, CA, pp. 961–967, 1993.
- Verbruggen, H.B., Zimmermann, H.-J., and Babuška, R., *Fuzzy Algorithms for Control*, Kluwer Academic Publishers, Boston, MA, 1999.
- Walichiewicz, L., Decomposition of linguistic rules in the design of a multi-dimensional fuzzy control algorithm, *Cybernetics and Systems Research 2: Proceedings of the 7th European Meeting on Cybernetics and Systems Research*, North-Holland, New York, pp. 557–561, 1984.

12 Model-Based Multivariable Fuzzy Control

Many real-world multivariable systems are nonlinear, complex, and time-varying, while multiple system input–output variables interact and sometimes even fight against each other, causing significant challenges for system control and optimization. Furthermore, in many practical applications, an accurate quantitative model of the system is not available or difficult to obtain, which subsequently makes it impossible to analyze the system multivariable interaction characteristics.

In this chapter, nonlinear multi-input multi-output (MIMO) systems are modeled by fuzzy basis function networks (FBFNs) based on a set of linguistic if-then rules, and then its multivariable interaction property is analyzed locally around the operating point based on the system relative gain array (RGA), which is systematically formulated using the system steady-state gain values. Two simulation examples of chemical distillation columns are presented to illustrate the system interaction degree obtained by this method, which is sufficiently accurate compared with the result calculated from the explicit mathematical models of the system.

Next, a systematic design procedure of a hierarchical multivariable fuzzy controller is presented based on the obtained system RGA and the proper control actions are calculated to compensate for the interaction effects of the MIMO system. This multivariable fuzzy controller is constructed with two orthogonal fuzzy control engines. The horizontal fuzzy control engine for each system input–output pair has a hierarchical structure to update the control parameters online and compensate for unknown system variations. The perpendicular fuzzy control engine is designed based on the system RGA to eliminate the multivariable interaction effect. The resultant closed-loop fuzzy control system is shown to be input–output passive stable. Two sets of simulation examples are shown to demonstrate that this fuzzy control strategy can be a promising way in controlling multivariable time-varying nonlinear systems with unknown system uncertainties and time-varying parameters.

12.1 FUZZY MODEL OF MULTIVARIABLE SYSTEMS

Consider a MIMO dynamic system with p inputs and q outputs, which can be written as a combination of multi-input single-output systems as

$$\begin{aligned}
y_j(k) = f_j[u_1(k-1), u_1(k-2), \dots, u_1(k-m_1+1), u_1(k-m_1), \\
\dots \\
u_p(k-1), u_p(k-2), \dots, u_p(k-m_p+1), u_p(k-m_p), \\
y_j(k-1), y_j(k-2), \dots, y_j(k-n_j+1), y_j(k-n_j)]
\end{aligned} \tag{12.1}$$

where

u_1, \dots, u_p are the p inputs to the system

y_j is the j th output

the indices m_1, \dots, m_p and n_j represent the system orders for the p inputs and
the j th output $f_j(\cdot)$ is the smooth function representing the system's nonlinear
dynamics

A system model can be constructed directly from the input/output measurements and
can be represented as the following fuzzy system:

$$\begin{aligned}
R^i: \text{IF } u_1(k-1) = A_{11}^i \text{ AND } u_1(k-2) = A_{12}^i \text{ AND } \dots \text{ AND } u_1(k-m_1) = A_{1m_1}^i \text{ AND } \\
\dots \\
u_p(k-1) = A_{p1}^i \text{ AND } u_p(k-2) = A_{p2}^i \text{ AND } \dots \text{ AND } u_p(k-m_p) = A_{pm_p}^i \text{ AND } \\
y_j(k-1) = B_{j1}^i \text{ AND } y_j(k-2) = B_{j2}^i \text{ AND } \dots \text{ AND } y_j(k-n_j) = B_{jn_j}^i \\
\text{THEN } y_j(k) = b_j^i
\end{aligned} \tag{12.2}$$

where

R^i ($i = 1, 2, \dots, l$) denotes the i th rule of the fuzzy model, l is the total number
of fuzzy rules

$A_{11}^i, \dots, A_{1m_1}^i, A_{p1}^i, \dots, A_{pm_p}^i, B_{j1}^i, \dots, B_{jn_j}^i$ are Gaussian membership functions
(MFs)

b_j^i represents a singleton function

The fuzzy system can be modeled by an FBFN since the FBFN provides a natural way of integrating all heterogeneous forms of information available from the system, such as analytical equations, heuristic rules, and experimental data. In the FBFN, a fuzzy system is represented as a series expansion of fuzzy basis functions (FBFs), which are algebraic superpositions of MFs. Each FBF corresponds to one fuzzy logic rule. Also, it has been shown that FBFN has the ability to uniformly approximate any continuous nonlinear function to a prescribed accuracy with a finite number of basis functions (Wang and Mendel, 1992). Assume a singleton fuzzifier, product inference, a centroid defuzzifier, and Gaussian MFs are used. For a given crisp input vector,

$$\begin{aligned}
\mathbf{x} &= [x_1, \dots, x_{m_1+\dots+m_p+n_j}]^T \\
&= [u_1(k-1), \dots, u_1(k-m_1), \dots, u_p(k-1), \dots, u_p(k-m_p), y_j(k-1), \dots, y_j(k-n_j)]^T
\end{aligned} \tag{12.3}$$

the system output $y_j(k)$ can be obtained as

$$\begin{aligned} y_j(k) &= f_j(\mathbf{x}) \\ &= \frac{\sum_{i=1}^l \left\{ b_j^i \cdot \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\}}{\sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right]} \end{aligned} \quad (12.4)$$

where $f_j: \mathbf{x} \subset \Re^{m_1+\dots+m_p+n_j} \rightarrow y_j \subset \Re \cdot \mu_i(x_t)$ is the Gaussian MF defined by

$$\mu_i(x_t) = \exp \left[-\frac{1}{2} \left(\frac{x_t - m_t^i}{\sigma_t^i} \right)^2 \right] \quad (12.5)$$

where m_t^i and σ_t^i are real-valued parameters as the center and width of each Gaussian MF. The system output in Equation 12.4 can also be written in its expansion form as

$$\begin{aligned} y_j(k) &= f_j(\mathbf{x}) \\ &= f_j(u_1(k-1), \dots, u_1(k-m_1), \dots, u_p(k-1), \dots, u_p(k-m_p), y_j(k-1), \dots, y_j(k-n_j)) \\ &= \sum_{i=1}^l \left(b_j^i \cdot \left\{ \prod_{t_1=1}^{m_1} \mu_{A_{1t_1}^i} [u_1(k-t_1)] \right\} \cdot \dots \cdot \left\{ \prod_{t_p=1}^{m_p} \mu_{A_{pt_p}^i} [u_p(k-t_p)] \right\} \cdot \left\{ \prod_{t_y=1}^{n_j} \mu_{B_{jt_y}^i} [y_j(k-t_y)] \right\} \right) \\ &\quad \sum_{i=1}^l \left(\left\{ \prod_{t_1=1}^{m_1} \mu_{A_{1t_1}^i} [u_1(k-t_1)] \right\} \cdot \dots \cdot \left\{ \prod_{t_p=1}^{m_p} \mu_{A_{pt_p}^i} [u_p(k-t_p)] \right\} \cdot \left\{ \prod_{t_y=1}^{n_j} \mu_{B_{jt_y}^i} [y_j(k-t_y)] \right\} \right) \end{aligned} \quad (12.6)$$

where

$$\begin{aligned} \mu_{A_{1t_1}^i} [u_1(k-t_1)] &= \exp \left\{ -\frac{1}{2} \left[\frac{u_1(k-t_1) - m_{1t_1}^i}{\sigma_{1t_1}^i} \right]^2 \right\}, \\ &\dots \\ \mu_{A_{pt_p}^i} [u_p(k-t_p)] &= \exp \left\{ -\frac{1}{2} \left[\frac{u_p(k-t_p) - m_{pt_p}^i}{\sigma_{pt_p}^i} \right]^2 \right\}, \\ \mu_{B_{jt_y}^i} [y_j(k-t_y)] &= \exp \left\{ -\frac{1}{2} \left[\frac{y_j(k-t_y) - m_{jt_y}^i}{\sigma_{jt_y}^i} \right]^2 \right\} \end{aligned} \quad (12.7)$$

Suppose a series of experiments are performed and the corresponding input-output data are measured and recorded. The objective of the system modeling is to determine the center m_t^i and the width σ_t^i of each Gaussian MF in the FBFN. The adaptive least-squares (ALS) algorithm proposed by Lee and Shin (2003), based on the least-squares method and genetic algorithm (GA), is used for the autonomous construction

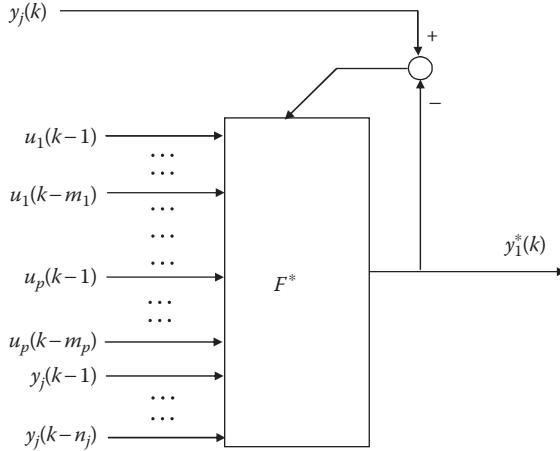


FIGURE 12.1 Nonlinear dynamic system modeling using FBFN.

of FBFN. It tries to sequentially locate one node $\{m_t^i, \sigma_t^i\}$ at a time that will maximize the error reduction measure using the GA until fuzzy rules are determined. The detailed construction procedure was explained by Lee and Shin (2003). The diagram of the system identification is shown in Figure 12.1, where $u_1(k-1), \dots, u_1(k-m_1), \dots, u_p(k-1), \dots, u_p(k-m_p), y_j(k-1), \dots, y_j(k-n_j)$ are the experimental input variables, F^* is the resultant FBFN model. $y_j(k)$ is the experimental output, and $y_j^*(k)$ is the model output from the FBFN.

12.2 MULTIVARIABLE INTERACTION ANALYSIS

The relative gain array (RGA) analysis method, known as one of the most systematic measures in quantifying the input–output interactions among control loops for multivariable linear systems (McAvoy, 1983), was originally proposed by Bristol (1966). The purpose of the relative gain is to provide the change in the gain of one loop when the other loops are closed in an interacting system. The individual relative gain element indicates the influence of a particular input to a specific output, which only requires the steady-state gain information on the system model and provides the degree of the multivariable interactions in a straightforward way. It can also be used as a measure of system closed-loop stability property and predict the sensitivity of the multivariable system to modeling errors (Grossdidier et al., 1985; Skogestad and Morari, 1987). Thus, this method has been used widely in multivariable control analysis. When the system is nonlinear, it is also possible to compute the RGA locally, after linearizing the model around the operating point (Reeves and Arkun, 1988; Chang and Yu, 1990).

However, accurate mathematical system models are not easy to derive in many industrial problems, such as complex manufacturing processes and chemical

processes, where a large number of variables are involved in the processes and interact with each other. The interactions are often nonlinear and not easy to quantify with crisp numeric precision, especially with inherent process uncertainties and disturbances. When optimization or control of such systems is desired, the interaction effects between the multiple inputs and multiple outputs need to be understood. The multivariable interaction analysis is however difficult to execute for these systems without mathematical models. In this section, a novel multivariable interaction analysis method is described for a general nonlinear MIMO system based on a system fuzzy model, which can be constructed from experimental data, heuristic rules, or a combination of both. First, the nonlinear multivariable system is modeled as an FBFN, which has been proved to have the ability to model any continuous nonlinear function to an arbitrary accuracy. And then, the steady-state gain array is calculated based on the FBFN around a specific operating point. Finally, the RGA is analyzed based on the obtained steady-state gains and the value of each array element indicates the interaction degree of each input–output pair for this multivariable system. The derivation procedure is explained in a systematic way and two simulation examples are presented to demonstrate the effectiveness of this interaction analysis method for a general MIMO nonlinear system without any explicit mathematical model.

12.2.1 RELATIVE GAIN ARRAY

For a MIMO system with p inputs and q outputs, when the system is operated in an open-loop condition as shown in Figure 12.2a, suppose a change in the input Δu_r is introduced and the other inputs are kept as constant. As a result, the system outputs will be affected and the corresponding changes are denoted as $[\Delta y_1, \dots, \Delta y_q]^\top$. In this condition, the steady-state gain between the r th input and the j th output is $k_{jr} = \Delta y_j / \Delta u_r$ (when $\Delta u_k = 0, \forall k \neq r$). In the other case, as shown in Figure 12.2b, suppose the $u_r - y_j$ pair is open, while all the other loops are closed with a perfect controller C to keep all the other outputs to be unchanged. In this closed-loop operation, the same change in Δu_r is introduced, and the steady-state gain can be calculated as $l_{jr} = \Delta y_j / \Delta u_r$ (when $\Delta y_k = 0, \forall k \neq j$). The relative gain for the pair of the r th input and the j th output is then defined as the ratio of the open-loop gain to the close-loop gain as

$$\begin{aligned}\lambda_{jr} &= \frac{k_{jr}}{l_{jr}} = \frac{(\Delta y_j / \Delta u_r) \text{ all loops are open}}{(\Delta y_j / \Delta u_r) \text{ all loops are closed except for the } u_r - y_j \text{ pair}} \\ &= \frac{(\Delta y_j / \Delta u_r)_{\Delta u_k=0, \text{ for all } k \neq r}}{(\Delta y_j / \Delta u_r)_{\Delta y_k=0, \text{ for all } k \neq j}}\end{aligned}\quad (12.8)$$

The element λ_{jr} is a relative measure of interaction between the system input u_r and the system output y_j , which indicates the ratio of the effect of a particular input u_r has on a particular output y_j with all loops open to the effect when all the other loops are closed. In Equation 12.8, the numerator denotes the case when all loops are open and the output y_j is only affected by the input u_r , and the denominator represents

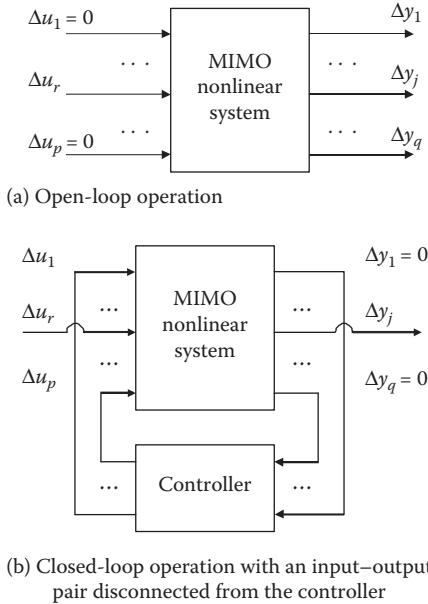


FIGURE 12.2 RGA definition.

the combined effects from the other inputs to the specific output y_j when all the other loops are closed. These effects are coming from the feedback control loops through the perfect controller C . Thus, the factor $1/\lambda_{jr}$ determines how much the loop gain of the $u_r - y_j$ loop changes because of the closure of the other loops. It should be noted here that the relative gain λ_{jr} indicates a relative input–output interaction degree, and this analysis method is different from the conventional control theory, where the system behavior can be characterized by the system gain and frequency information in the frequency response.

12.2.1.1 Relative Gain Array for Square Systems

By considering the ratio λ_{jr} , Bristol (1966) obtained a dimensionless number, which is independent of the scaling of input/output variables. Consider the following transfer function for a multivariable system, obtained by linearizing the system equation around a nominal operating condition:

$$\Delta\mathbf{y}(s) = \mathbf{G}(s)\Delta\mathbf{u}(s) \quad (12.9)$$

where $\Delta\mathbf{u}(s)$ and $\Delta\mathbf{y}(s)$ are the deviations of system inputs and system outputs with respect to their nominal values. A square system is considered with both $\Delta\mathbf{u}(s)$ and $\Delta\mathbf{y}(s)$ being vectors of dimension m . $\mathbf{G}(s)$ is an $m \times m$ system gain matrix whose element is indicated as $[\mathbf{G}(s)]_{jr}$ between the r th input and the j th output.

For a nonsingular square system, the open-loop gain matrix $\mathbf{G}(0)$ is the system transfer function matrix $\mathbf{G}(s)$ in the steady state ($s=0$) with $\mathbf{G}(0) \cdot \Delta \mathbf{u} = \Delta \mathbf{y}$. In open-loop operating condition as shown in [Figure 12.2a](#), the other inputs are kept unchanged except that Δu_r is introduced to the system. Thus,

$$\mathbf{G}(0) \cdot \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \Delta u_r \\ \vdots \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \\ \vdots \\ \Delta y_j \\ \vdots \\ \Delta y_{m-1} \\ \Delta y_m \end{bmatrix}$$

For the input–output pair $u_r - y_j$, $[\mathbf{G}(0)]_{jr} \cdot \Delta u_r = \Delta y_j$ and the steady-state gain is $k_{jr} = \Delta y_j / \Delta u_r = [\mathbf{G}(0)]_{jr}$.

On the other hand, the closed-loop gain is defined as the gain between y_j and u_r when all other outputs are under perfect control, which always holds the system outputs at their set point values. In the steady state ($s=0$) of the system shown in [Figure 12.2b](#), the perfect controller C is equal to $[\mathbf{G}(0)]^{-1}$ and the equation holds as $[\mathbf{G}(0)]^{-1} \cdot \Delta \mathbf{y} = \Delta \mathbf{u}$ (Chang and Yu, 1990). When all the other system outputs are kept at their set point values, we obtain

$$[\mathbf{G}(0)]^{-1} \cdot \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \Delta y_j \\ \vdots \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \vdots \\ \Delta u_r \\ \vdots \\ \Delta u_{m-1} \\ \Delta u_m \end{bmatrix}$$

For the input–output pair $u_r - y_j$, $\{[\mathbf{G}(0)]^{-1}\}_{rj} \cdot \Delta y_j = \Delta u_r$ and the steady-state gain is obtained as $l_{jr} = \Delta y_j / \Delta u_r = 1 / \{[\mathbf{G}(0)]^{-1}\}_{rj}$.

The RGA for this multivariable system is then defined in a matrix form as

$$\Lambda = [\lambda_{jr}] \quad r = 1, 2, \dots, m \quad j = 1, 2, \dots, m \quad (12.10)$$

where

- r is the index of system input
- j is the index of system output

The array element values can be obtained through the element-by-element multiplication as

$$\Lambda = \mathbf{G}(0) \times \{[\mathbf{G}(0)]^{-1}\}^T \quad (12.11)$$

where “ \times ” denotes Schur product (or Hadamard product) (Bristol, 1966; McAvoy, 1983). The elements of the obtained RGA have the following properties (McAvoy, 1983; Deshpande, 1989):

- The sum of the elements of each row and each column of the RGA is always unity:

$$\sum_{r=1}^m \lambda_{jr} = \sum_{j=1}^m \lambda_{jr} = 1 \quad (12.12)$$

- The element λ_{jr} is dimensionless, thus it is invariant under input and output scaling.
- The value of λ_{jr} is a measure of the steady-state interaction between the system input u_r and the system output y_j .
- Since the relative gain element tells how much the gain of one loop changes when the other loops are closed, it provides an indication of the multi-variable interaction for a particular input–output loop pair.

12.2.1.2 Relative Gain Array for Nonsquare Systems

In the real world applications, systems having an unequal number of inputs and outputs are not uncommon. For control purpose, they are usually treated as square systems by first adding or deleting an appropriate number of inputs or outputs from the system, and then using a centralized controller to control this squared system. However, it is often not desirable to add or delete the requisite number of inputs or outputs to obtain the squared system. When the number of inputs is more than the number of outputs, adding unnecessary outputs can be costly, and deleting inputs leaves fewer variables manipulatable to achieve the desired control goal. On the other hand, adding new inputs increases the cost, and reducing the number of outputs decreases the amount of feedback information available to the system (Reeves and Arkun, 1988). Hence, it is preferable that superior control performance can be achieved in their original nonsquare form rather than squaring them. Consequently, the RGA analysis method should be extended to nonsquare multivariable systems as well.

Same as the square RGA, the nonsquare RGA can be used to assess the performance of nonsquare multivariable systems based on the steady-state information. Consider a nonsquare system with p inputs and q outputs as

$$\Delta\mathbf{y}(s) = \mathbf{G}(s)\Delta\mathbf{u}(s) \quad (12.13)$$

The dimensions of $\Delta\mathbf{u}(s)$ and $\Delta\mathbf{y}(s)$ are p and q , and $\mathbf{G}(s)$ is a $q \times p$ transfer function matrix. From the results shown in Chang and Yu (1990) and Reeves and Arkun (1988), the nonsquare relative gain element λ_{jr}^N is defined as the ratio of open-loop gain to closed-loop gain in the least-square sense. For a nonsquare system, the interpretation of the closed-loop gain in the denominator of Equation 12.8 is different from the square system case because of the presence of the inverse term introduced in the derivation of the RGA. A more general definition that is capable of handling the cases with nonsquare multivariable systems is required.

PROPERTY 12.1 (Graybill, 1969)

Let $\mathbf{G} \in C^{q \times p}$

1. If $\mathbf{G} \in C_q^{q \times p}$, then $\mathbf{G}^+ \in \mathbf{G}^H (\mathbf{G}\mathbf{G}^H)^{-1}$ and $\mathbf{G}\mathbf{G}^+ = \mathbf{I}$.
2. If $\mathbf{G} \in C_p^{q \times p}$, then $\mathbf{G}^+ \in (\mathbf{G}^H \mathbf{G})^{-1} \mathbf{G}^H$ and $\mathbf{G}^+ \mathbf{G} = \mathbf{I}$.

This theorem establishes the expressions for the pseudoinverse of a matrix with full rank. $C_r^{q \times p}$ denotes a complex $q \times p$ matrix with rank r , and the superscript H indicates the Hermitian transpose of a matrix. In this way, $\mathbf{G} \in C_q^{q \times p}$ means that the system has more inputs than outputs, and $\mathbf{G} \in C_p^{q \times p}$ is the opposite case.

It is noted that Equation 12.11 is still valid when the number of system inputs is not equal to the number of system outputs. This problem is essentially related to the computation of the matrix inverse in Equation 12.11. When \mathbf{G} is not square, \mathbf{G}^+ is a pseudoinverse rather than an inverse matrix. Thus, from the steady-state transfer function matrix $\mathbf{G}(0)$ and its pseudoinverse $\mathbf{G}(0)^+$, the nonsquare RGA Λ^N can be evaluated as

$$\Lambda^N = \mathbf{G}(0) \times [\mathbf{G}(0)^+]^T \quad (12.14)$$

Note here in this case, since \mathbf{G} is not square, it is no longer true that any row or column sums to 1. Instead, if $p > q$, we have $\mathbf{G}\mathbf{G}^+ = \mathbf{I}$, hence the elements in each row of the RGA sum to 1, $\sum_{r=1}^p \lambda_{jr} = 1$. If $p < q$, then $\mathbf{G}^+ \mathbf{G} = \mathbf{I}$ and thus the elements in each column sum to 1 as $\sum_{j=1}^q \lambda_{jr} = 1$. ■

12.2.2 INTERACTION ANALYSIS IN MULTIVARIABLE FUZZY MODELS

Real-world multivariable systems are often nonlinear, time-varying with unexpected uncertainties and disturbances (Xu and Shin, 2007). An accurate mathematical system model is often difficult to obtain. For example, grinding processes are complex in nature with multiple cutting points, which are defined by a large number of grits possessing irregular shapes, sizes, and spacing. Various uncertainties and disturbances inherently exist in the process, which make it difficult to formulate accurate mathematical models to encompass all the aspects of grinding processes. A large number of variables are involved in the process with multiple inputs and multiple outputs mutually affecting each other at the same time, where many of these interactions are nonlinear, time-varying, and difficult to quantify with crisp numeric precision. In distillation columns in chemical processes, when pulse inputs are given in the reflux flow, the resultant liquid flows, vapor rates, and compositions will overshoot with significant oscillation instead of returning smoothly to stable status, which makes it difficult to setup the system transfer function matrix by observing system steady-state input–output data (Doukas and Luyben, 1978). Thus, in real-industrial applications without a system explicit analytical model, challenges often exist in analysis, optimization, and control of multivariable systems with inherent interaction properties.

In this section, the multivariable interaction analysis is performed based on a system FBFN model, which can be constructed from experimental data, human knowledge, or a combination of both. For a MIMO fuzzy model in the form of FBFN, the input–output interaction effect cannot be expressed with a single fuzzy rule. Thus, RGA will be calculated for the combination of all fuzzy rules to derive the interaction effects among these variables. The standard method of calculating RGA depends on the availability of a linear system model, which is not directly applicable to the MIMO fuzzy model. Because of the inherent nonlinear nature existing in the FBFN model, the behavior of the nonlinear fuzzy model depends on the operating point. Thus, the FBFN model needs to be first linearized around the point of interest, and the RGA will be calculated locally based on local linearization.

Consider a general MIMO nonlinear system, which is modeled in the form of FBFN as in Equation 12.2. Around a certain operating point \mathbf{x}_0 , where

$$\mathbf{x}_0 = [u_{10}(k-1), \dots, u_{10}(k-m_1), \dots, u_{p0}(k-1), u_{p0}(k-m_p), y_{j0}(k-1), \dots, y_{j0}(k-n_j)]^T \quad (12.15)$$

the nonlinear system in Equation 12.6 can be locally linearized as follows:

$$\begin{aligned} y_j(k) &= f_j(\mathbf{x}) \\ &= f_j(u_1(k-1), \dots, u_1(k-m_1), \dots, u_p(k-1), \dots, u_p(k-m_p), y_j(k-1), \dots, y_j(k-n_j)) \\ &= f_{j0}(u_{10}(k-1), \dots, u_{10}(k-m_1), \dots, u_{p0}(k-1), \dots, u_{p0}(k-m_p), y_{j0}(k-1), \dots, y_{j0}(k-n_j)) \\ &\quad + \frac{\partial f_j}{\partial u_1(k-1)} \Delta u_1(k-1) + \dots + \frac{\partial f_j}{\partial u_1(k-m_1)} \Delta u_1(k-m_1) \\ &\quad + \dots + \frac{\partial f_j}{\partial u_p(k-1)} \Delta u_p(k-1) + \dots + \frac{\partial f_j}{\partial u_p(k-m_p)} \Delta u_p(k-m_p) \\ &\quad + \frac{\partial f_j}{\partial y_j(k-1)} \Delta y_j(k-1) + \dots + \frac{\partial f_j}{\partial y_j(k-n_j)} \Delta y_j(k-n_j) \end{aligned} \quad (12.16)$$

where $\frac{\partial f_j}{\partial u_i(k-1)}$ is a simplified expression, which denotes the function value evaluated at point \mathbf{x}_0 as

$$\frac{\partial f_j}{\partial u_1(k-1)} = \left. \frac{\partial f_j}{\partial u_1(k-1)} \right|_{\mathbf{x}=\mathbf{x}_0} \quad (12.17)$$

Similarly, $\frac{\partial f_j}{\partial u_1(k-m_1)}, \dots, \frac{\partial f_j}{\partial u_p(k-1)}, \dots, \frac{\partial f_j}{\partial u_p(k-m_p)}, \dots, \frac{\partial f_j}{\partial y_j(k-1)}, \dots, \frac{\partial f_j}{\partial y_j(k-n_j)}$ are also the values evaluated at point \mathbf{x}_0 . Denote

$$\begin{aligned} \Delta y_j(k) &= y_j(k) - y_{j0}(k) \\ &= y_j(k) - f_{j0}(u_{10}(k-1), \dots, u_{10}(k-m_1), \dots, u_{p0}(k-1), \dots, u_{p0}(k-m_p), \\ &\quad y_{j0}(k-1), \dots, y_{j0}(k-n_j)) \end{aligned} \quad (12.18)$$

Then we have

$$\begin{aligned}
\Delta y_j(k) &= \frac{\partial f_j}{\partial u_1(k-1)} \Delta u_1(k-1) + \cdots + \frac{\partial f_j}{\partial u_1(k-m_1)} \Delta u_1(k-m_1) \\
&\quad + \cdots + \frac{\partial f_j}{\partial u_p(k-1)} \Delta u_p(k-1) + \cdots + \frac{\partial f_j}{\partial u_p(k-m_p)} \Delta u_p(k-m_p) \\
&\quad + \frac{\partial f_j}{\partial y_j(k-1)} \Delta y_j(k-1) + \cdots + \frac{\partial f_j}{\partial y_j(k-n_j)} \Delta y_j(k-n_j)
\end{aligned} \tag{12.19}$$

Thus, at steady state, the input-output gains are obtained as

$$\begin{aligned}
g_{j1} &= \frac{\sum_{t_1=1}^{m_1} \frac{\partial f_j}{\partial u_1(k-t_1)}}{1 - \sum_{t_y=1}^{n_j} \frac{\partial f_j}{\partial y_j(k-t_y)}} = \frac{\frac{\partial f_j}{\partial u_1(k-1)} + \cdots + \frac{\partial f_j}{\partial u_1(k-m_1)}}{1 - \left[\frac{\partial f_j}{\partial y_j(k-1)} + \cdots + \frac{\partial f_j}{\partial y_j(k-n_j)} \right]} \\
&\vdots \\
g_{jp} &= \frac{\sum_{t_p=1}^{m_p} \frac{\partial f_j}{\partial u_p(k-t_p)}}{1 - \sum_{t_y=1}^{n_j} \frac{\partial f_j}{\partial y_j(k-t_y)}} = \frac{\frac{\partial f_j}{\partial u_p(k-1)} + \cdots + \frac{\partial f_j}{\partial u_p(k-m_p)}}{1 - \left[\frac{\partial f_j}{\partial y_j(k-1)} + \cdots + \frac{\partial f_j}{\partial y_j(k-n_j)} \right]}
\end{aligned} \tag{12.20}$$

From Equations 12.4 and 12.5, one can obtain

$$\begin{aligned}
\frac{\partial f_j}{\partial u_r(k-t_r)} &= \frac{\left(\sum_{i=1}^l \left\{ b_j^i \cdot \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right] \right\} \right)' \cdot \left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right] \right\}}{\left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right] \right\}^2} \\
&\quad - \frac{\left(\sum_{i=1}^l \left\{ b_j^i \cdot \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right] \right\} \right) \cdot \left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right] \right\}'}{\left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right] \right\}^2} \\
&= - \frac{\sum_{i=1}^l \left\{ b_j^i \cdot \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right] \left[\frac{u_r(k-t_r)-m_{rt_r}^i}{\sigma_{rt_r}^2} \right] \right\}}{\sum_{i=1}^l \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right]} \\
&\quad + \frac{\left(\sum_{i=1}^l \left\{ b_j^i \cdot \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right] \right\} \right) \cdot \left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right] \left[\frac{u_r(k-t_r)-m_{rt_r}^i}{\sigma_{rt_r}^2} \right] \right\}}{\left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\cdots+m_p+n_j} \mu_i(x_t) \right] \right\}^2} \\
&\quad \dots
\end{aligned}$$

$$\begin{aligned}
\frac{\partial f_j}{\partial y_j(k-t_y)} &= \frac{\left(\sum_{i=1}^l \left\{ b_j^i \cdot \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\} \right)' \cdot \left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\}}{\left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\}^2} \\
&\quad - \frac{\left(\sum_{i=1}^l \left\{ b_j^i \cdot \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\} \right) \cdot \left(\sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right)' }{\left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\}^2} \\
&= - \frac{\sum_{i=1}^l \left\{ (b_j^i) \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \left[\sum_{t_y=1}^{n_j} \frac{y_j(k-t_y)-m_{j,t_y}^i}{\sigma_{j,t_y}^{i,2}} \right] \right\}}{\sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right]} \\
&\quad + \frac{\left(\sum_{i=1}^l \left\{ (b_j^i) \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\} \right) \cdot \left(\sum_{i=1}^l \left\{ \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \left[\sum_{t_y=1}^{n_j} \frac{y_j(k-t_y)-m_{j,t_y}^i}{\sigma_{j,t_y}^{i,2}} \right] \right\} \right)}{\left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\}^2}
\end{aligned}$$

Thus,

$$\begin{aligned}
\sum_{r=1}^{m_r} \frac{\partial f_j}{\partial u_r(k-t_r)} &= - \frac{\sum_{i=1}^l \left\{ (b_j^i) \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \left[\sum_{t_r=1}^{m_r} \frac{u_r(k-t_r)-m_{r,t_r}^i}{\sigma_{r,t_r}^{i,2}} \right] \right\}}{\sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right]} \\
&\quad + \frac{\left(\sum_{i=1}^l \left\{ (b_j^i) \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\} \right) \cdot \left(\sum_{i=1}^l \left\{ \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \left[\sum_{t_r=1}^{m_r} \frac{u_r(k-t_r)-m_{r,t_r}^i}{\sigma_{r,t_r}^{i,2}} \right] \right\} \right)}{\left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\}^2} \\
&\quad \dots \\
\sum_{t_y=1}^{n_j} \frac{\partial f_j}{\partial y_j(k-t_y)} &= - \frac{\sum_{i=1}^l \left\{ (b_j^i) \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \left[\sum_{t_y=1}^{n_j} \frac{y_j(k-t_y)-m_{j,t_y}^i}{\sigma_{j,t_y}^{i,2}} \right] \right\}}{\sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right]} \\
&\quad + \frac{\left(\sum_{i=1}^l \left\{ (b_j^i) \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\} \right) \cdot \left(\sum_{i=1}^l \left\{ \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \left[\sum_{t_y=1}^{n_j} \frac{y_j(k-t_y)-m_{j,t_y}^i}{\sigma_{j,t_y}^{i,2}} \right] \right\} \right)}{\left\{ \sum_{i=1}^l \left[\prod_{t=1}^{m_1+\dots+m_p+n_j} \mu_i(x_t) \right] \right\}^2}
\end{aligned}$$

And then the steady-state gain array can be easily obtained as

$$\mathbf{G}(0) = [g_{jr}] = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ g_{q1} & g_{q2} & \cdots & g_{qp} \end{bmatrix} \quad (12.21)$$

The RGA can be calculated as $\Lambda = \mathbf{G}(0) \times [\mathbf{G}(0)^{-1}]^T$ for square systems and $\Lambda^N = \mathbf{G}(0) \times [\mathbf{G}(0)^+]^T$ for nonsquare systems.

The relative gain λ_{jr} determines how much the loop gain of the $u_r - y_j$ loop changes because of the closure of the other loops. In this way, the property of the interaction effect of this multivariable system can be indicated by the value of element λ_{jr} as follows (Bristol, 1966; Kinnaert, 1995; Shinskey, 1996):

1. $\lambda_{jr} < 0$: In this case, the gain between the input u_r and the output y_j changes sign when the other loops are closed. Closing the other loops will induce instability in general.
2. $\lambda_{jr} = 0$: In this condition, the input u_r has no influence on the output y_j around the considered operating point.
3. $0 < \lambda_{jr} < 1$: Such situation indicates that closing the other loops will increase the gain between the input u_r and the output y_j . The system may get oscillatory or even unstable when the other loops are closed.
4. $\lambda_{jr} = 1$: In this situation, the open-loop gain and the closed-loop gain between the input u_r and the output y_j are identical. Closing the other loops will not affect the gain of the loop between the input u_r and the output y_j around the consideration point.
5. $\lambda_{jr} > 1$: Such situation occurs when closing the other loops decreases the gain between the input u_r and the output y_j . The system response tends to be sluggish. The larger λ_{jr} , the worse the interactions.

12.2.3 SIMULATION EXAMPLES

In this section, simulation examples are presented for two chemical distillation columns. An industrial distillation column is often viewed as a series of integrated and cascaded tanks. The integration usually exhibits a complex and nonintuitive behavior, and it is difficult to understand the overall system property based on the knowledge of each individual tank's characteristics. Furthermore, the processes are often highly nonlinear, and the process behavior heavily depends on each input magnitude and certain operating condition. An understanding of the dynamics, operation, and control of distillation columns is becoming an important issue, especially without an accurate system mathematical model (Skodestad, 1997). In these two simulation examples, the multivariable interaction degree is calculated from the trained system FBFN model, and the interaction degree is illustrated to be similar to the result obtained from the explicit system mathematical model. This demonstrates the effectiveness of the FBFN-based method as an alternative way in practical industrial applications in the actual conditions when the system's accurate analytical model is not available.

Example 12.1 Distillation Column

In this example, a high-purity binary distillation column is used as a case study. Consider the following model of a high-purity distillation column for a binary mixture (Morari and Zafiriou, 1989):

$$\begin{bmatrix} y_D \\ y_B \end{bmatrix} = \begin{bmatrix} \frac{87.8}{1+T_1s} & -\frac{87.8}{1+T_1s} + \frac{1.4}{1+T_2s} \\ \frac{108.2}{1+T_1s} & -\frac{108.2}{1+T_1s} - \frac{1.4}{1+T_2s} \end{bmatrix} \begin{bmatrix} L \\ V \end{bmatrix} \quad (12.22)$$

where

$T_1 = 194$ min is the time constant for changes in the external flows

$T_2 = 15$ min is the time constant for changes in internal flows

y_D is the mole fraction of light component in the over-head vapor

y_B is the mole fraction of light component in the bottom product

L is the reflux flow

V is the boiler flow

An FBFN model is constructed using 1000 input-output data pairs sampled with sampling time of 2 min. Consider the model of the distillation column valid at $y_D = 1.4$ and $y_B = -1.4$. For the first system output y_D , three fuzzy rules are obtained using the ALS algorithm based on the least-squares method and GA (Lee and Shin, 2003) as

$$\begin{aligned} R^i: \text{IF } & L(k) = A_{11}^i \text{ AND } L(k-1) = A_{12}^i \text{ AND} \\ & V(k) = A_{21}^i \text{ AND } V(k-1) = A_{22}^i \text{ AND} \\ & y_D(k-1) = B_{y1}^i \text{ AND } y_D(k-2) = B_{y2}^i, \\ \text{THEN } & y_D(k) = b_y^i \end{aligned}$$

The parameters for the input and output MFs are as follows:

Rule	m_{11}	σ_{11}	m_{12}	σ_{12}	m_{21}	σ_{21}	m_{22}	σ_{22}	m_{y1}	σ_{y1}	m_{y2}	σ_{y2}	b_y
1	0.77	5.57	0.71	1.78	1.04	2.72	0.55	2.25	1.76	0.96	1.76	1.62	3.05
2	0.12	1.29	1.26	1.72	0.88	2.26	0.71	1.98	0.72	1.04	0.61	4.19	-2.2
3	0.46	1.30	0.32	1.17	1.41	1.44	0.43	1.57	1.15	2.18	0.59	2.91	2.48

For the second system output y_B , three fuzzy rules are obtained in the form of

$$\begin{aligned} R^i: \text{IF } & L(k) = A_{11}^i \text{ AND } L(k-1) = A_{12}^i \text{ AND} \\ & V(k) = A_{21}^i \text{ AND } V(k-1) = A_{22}^i \text{ AND} \\ & y_B(k-1) = B_{x1}^i \text{ AND } y_B(k-2) = B_{x2}^i, \\ \text{THEN } & y_B(k) = b_x^i \end{aligned}$$

The parameters for the input and output MFs are as follows:

Rule	m_{11}	σ_{11}	m_{12}	σ_{12}	m_{21}	σ_{21}	m_{22}	σ_{22}	m_{x1}	σ_{x1}	m_{x2}	σ_{x2}	b_x
1	0.75	2.83	0.77	3.86	0.31	1.70	0.98	3.37	-2.6	3.49	-2.7	1.28	-7.1
2	0.96	3.74	0.51	1.79	0.71	3.38	0.1	1.94	-1.1	4.86	-2.1	1.27	2.57
3	1.22	0.20	-0.0	0.6	0.22	1.24	1.27	2.72	-1.7	3.95	-1.1	2.00	-0.1

At the current steady state as $y_D = 1.4$ and $y_B = -1.4$, the input–output gain matrix is calculated based on Equations 12.20 and 12.21 as

$$\mathbf{G}(0) = \begin{bmatrix} 87.29 & -85.89 \\ 107.58 & -108.98 \end{bmatrix}$$

And the RGA for this two-input two-output system is calculated as

$$\Lambda = \mathbf{G}(0) \times [\mathbf{G}(0)^{-1}]^T = \begin{bmatrix} 34.87 & -33.87 \\ -33.87 & 34.87 \end{bmatrix}$$

Comparing $\tilde{\Lambda}$ with the system RGA obtained from the system mathematical model in Skodestad (1997) at the same operating condition:

$$\tilde{\Lambda} = \begin{bmatrix} 35.0688 & -34.0688 \\ -34.0688 & 35.0688 \end{bmatrix}$$

These two results are similar, which indicates the proposed RGA analysis method as an effective alternative way to derive the multivariable interaction with a sufficient precision in the condition when the system mathematical model is not available.

Example 12.2 Distillation Column

This example is a side-stream distillation column to separate benzene, toluene, and xylene. The system mathematical equation is described in Doukas and Luyben (1978) and Chang and Yu (1990):

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} -9.811e^{-1.59s} & 0.374e^{-7.75s} & -11.3e^{-3.79s} \\ \frac{1}{11.36s + 1} & \frac{(22.2s + 1)^2}{(21.74s + 1)^2} & \frac{5.24e^{-60s}}{(400s + 1)^2} \\ \frac{5.984e^{-2.24s}}{14.29s + 1} & \frac{-1.986e^{-0.71s}}{(66.67s + 1)^2} & \frac{-0.33e^{-0.68s}}{(2.38s + 1)^2} \\ \frac{2.38e^{-0.42s}}{(1.43s + 1)^2} & \frac{0.0204e^{-0.59s}}{(7.14s + 1)^2} & \frac{4.48e^{-0.52s}}{(11.11s + 1)^2} \\ \frac{-11.67e^{-1.91s}}{12.19s + 1} & \frac{-0.176e^{-0.48s}}{(6.9s + 1)^2} & \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad (12.23)$$

The three input variables are reboiler duty u_1 , reflux ratio u_2 , and side-stream flow rate u_3 . The four process outputs are toluene in bottom y_1 , toluene in bottom y_2 , benzene in side draw y_3 , and benzene in side draw y_4 . One thousand input–output data pairs are generated based on the system model, and a set of FBFN models are constructed for the four system outputs individually as

$$\begin{aligned} R^i: \text{IF } & u_1(kT - 1.59) = A_{11}^i \text{ AND } u_1((k-1)T - 1.59) = A_{12}^i \text{ AND} \\ & u_2(kT - 7.75) = A_{21}^i \text{ AND} \\ & u_3(kT - 3.79) = A_{31}^i \text{ AND} \\ & y_1[(k-1)T] = B_{1y1}^i \text{ AND } y_1[(k-2)T] = B_{1y2}^i, \\ \text{THEN } & y_1(kT) = b_{1y}^i \end{aligned}$$

R^i : IF $u_1(kT - 2.24) = A_{11}^i$ AND $u_1[(k-1)T - 2.24] = A_{12}^i$ AND
 $u_2(kT - 0.71) = A_{21}^i$ AND
 $u_3(kT - 60) = A_{31}^i$ AND
 $y_2[(k-1)T] = B_{2y1}^i$ AND $y_2[(k-2)T] = B_{2y2}^i$,
 THEN $y_2(kT) = b_{2y}^i$

R^i : IF $u_1(kT - 0.42) = A_{11}^i$ AND
 $u_2(kT - 0.59) = A_{21}^i$ AND
 $u_3(kT - 0.68) = A_{31}^i$ AND
 $y_3[(k-1)T] = B_{3y1}^i$ AND $y_3[(k-2)T] = B_{3y2}^i$,
 THEN $y_3(kT) = b_{3y}^i$

R^i : IF $u_1(kT - 1.91) = A_{11}^i$ AND $u_1[(k-1)T - 1.91] = A_{12}^i$ AND
 $u_2(kT - 0.48) = A_{21}^i$ AND
 $u_3(kT - 0.52) = A_{31}^i$ AND
 $y_4[(k-1)T] = B_{4y1}^i$ AND $y_4[(k-2)T] = B_{4y2}^i$,
 THEN $y_4(kT) = b_{4y}^i$

where T is the sampling time specified as 1 min. The steady-state gain matrix is calculated based on Equations 12.20 and 12.21 from the FBFN models as

$$\mathbf{G}(0) = \begin{bmatrix} -9.811 & 0.374 & -11.30 \\ 5.984 & -1.986 & 3.566 \\ 2.380 & 0.020 & -0.330 \\ -11.67 & -0.176 & 4.480 \end{bmatrix}$$

and the RGA for this three-input four-output system is calculated as

$$\mathbf{A} = \mathbf{G}(0) \times [\mathbf{G}(0)^{-1}]^+ = \begin{bmatrix} 0.242 & -0.077 & 0.834 \\ 0.006 & 1.060 & -0.066 \\ 0.028 & 0.001 & 0.002 \\ 0.725 & 0.017 & 0.230 \end{bmatrix}$$

Denote $\tilde{\mathbf{A}}$ as the RGA obtained from the system mathematical model in reference (Chang and Yu, 1990) at this point:

$$\tilde{\mathbf{A}} = \begin{bmatrix} 0.241 & -0.103 & 0.861 \\ 0.006 & 1.094 & -0.100 \\ 0.028 & 0.000 & 0.002 \\ 0.726 & 0.008 & 0.237 \end{bmatrix}$$

This result calculated from the system mathematical model is in a good agreement with the one obtained from the FBFN model, which thus justifies the FBFN-based RGA analysis method as an effective way to derive the multivariable interaction degree without any accurate system analytical model.

12.2.4 CONCLUSION

In multivariable control that is becoming increasingly important in many industrial applications, the cross-coupling effects often exist, that is, when one control variable changes, more than one system outputs will be affected. The presented method focuses on the input–output interaction analysis for MIMO systems based on the FBFN model, which is used to model complex nonlinear multivariable systems when mathematical models are not available. A novel method of determining RGA for the FBFN model has been proposed to represent the multivariable interaction degree based on system steady-state gain around the specific operating point. Since RGA requires only steady-state information, it is a valuable tool for analyzing the controlled systems and can be integrated into real-time multivariable controller design. Two simulation studies demonstrated the efficiency and accuracy of the described method in obtaining the RGA to determine the multivariable interaction effect when the system mathematical model is not available.

12.3 MULTIVARIABLE FUZZY CONTROL DESIGN

A novel proposed multivariable fuzzy controller is constructed with two perpendicular fuzzy control engines. The horizontal fuzzy control engine for each system input–output pair has a hierarchical structure to update the control parameters online and compensate for unknown system variations. The perpendicular fuzzy control engine is designed based on the system RGA to eliminate the multivariable interaction effects. The resultant closed-loop fuzzy control system is proved to be input–output passive stable. Two sets of simulation examples demonstrate that the proposed fuzzy control strategy can be a promising way in controlling multivariable time-varying nonlinear systems with unknown system uncertainties and time-varying parameters.

The structure of the multivariable fuzzy controller is shown in Figure 12.3, where the controller consists of two orthogonal fuzzy control engines. The first

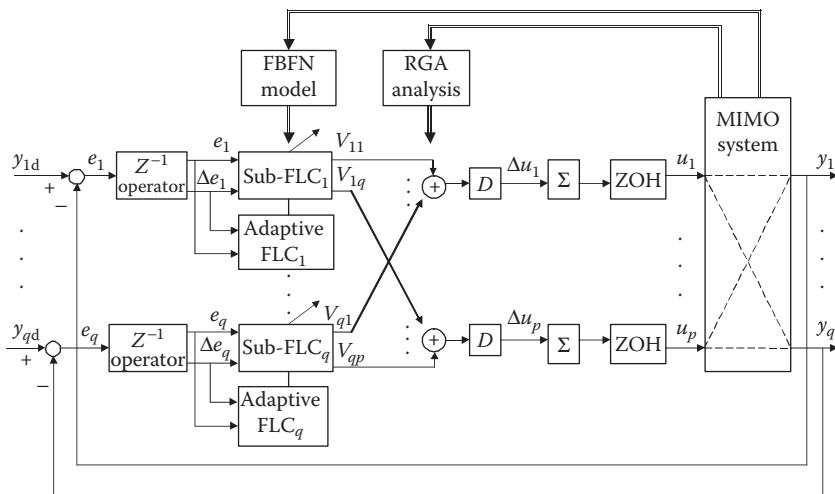


FIGURE 12.3 Multivariable fuzzy controller.

fuzzy control engine is horizontal-wise, which consists of a set of sub-fuzzy logic controllers (sub-FLCs) and adaptive fuzzy logic controllers (adaptive FLCs) for each individual input–output pair. The sub-FLC works similar as a regular fuzzy controller for a single-input single-output system. The inputs to the j th sub-FLC are crisp numbers of the error e_j and the change of the error Δe_j . The only difference is that the obtained control action V_{jr} for the $u_r - y_j$ input–output pair is a fuzzy set instead of a crisp number. In order to correct the imprecision of the control rules in the sub-FLC and compensate for the time-varying parameters or system uncertainties, the adaptive FLC is designed to tune the output MFs of the sub-FLC based on the system performance. After the fuzzy inference calculation from the first fuzzy control engine, the fuzzy control action V_{jr} is obtained and further fed into the second fuzzy control engine, which is designed perpendicular to the first fuzzy control engine in order to eliminate the input–output interaction in the MIMO system. The inputs to the r th perpendicular fuzzy control engine are the fuzzy control actions $V_{1r}, V_{2r}, \dots, V_{qr}$, which are aggregated based on the system RGA to derive the change of the control action ΔU_r for the r th input. The D operator in Figure 12.3 represents the defuzzification calculation that transforms the fuzzy control action into the crisp control action Δu_r . The ultimate control action $u_r(k)$ is obtained in an incremental form through zero-order hold (ZOH) calculation.

12.3.1 HORIZONTAL FUZZY CONTROL ENGINE

The horizontal fuzzy control engine consists of two layers of fuzzy controllers for each individual system input–output pair: sub-FLC and adaptive FLC. The inputs to the sub-FLC are the error e_j and the change of the error Δe_j . The fuzzy control rules are generated based on the system steady-state gain matrix $\mathbf{G}(0)$ in Equation 12.21. The output is the fuzzy control action V_{jr} for the $u_r - y_j$ input–output pair. Triangular membership functions are defined over the universe of discourse for the input and output individually (Xu and Shin, 2005).

The fuzzy control rules in the sub-FLC are in the form of

$$R^i: \text{IF } e_1 \text{ is } E_x^i \text{ AND } e_2 \text{ is } E_y^i, \text{ THEN } V_{jr} \text{ is } V_{n(x,y)}^i \quad (12.24)$$

where

e_1 is the error $e_j(k)$ at time instant k

e_2 is the change of the error $\Delta e_j(k)$

E_x^i and E_y^i are the linguistic variables of e_1 and e_2 , respectively

$V_{n(x,y)}^i$ is the linguistic variable of V_{jr} with $n(x,y)$ representing any function whose value at x and y is an integer (Ying, 1994; Calcev, 1998; Aracil and Gordillo, 2000; Farinwata et al., 2000)

Mamdani's minimum fuzzy implication is adopted and the fuzzy relations are aggregated as

$$R = \bigcup_{i=1}^l R^i = \bigcup_{E_x, E_y, V_{n(x,y)}} (R^1, \dots, R^l) = \bigcup_{i=1}^l [E_x^i \cap E_y^i \cap V_{n(x,y)}^i] \quad (12.25)$$

where

$$i = 1, 2, \dots, l$$

l is the total number of fuzzy rules

\cup is the union operator

\cap is the intersection operator

Given the current fuzzy inputs e_1 and e_2 , the fuzzy control action V_{jr} is obtained by the compositional rule of inferencing:

$$V_{jr} = (e_1 \times e_2) \circ R = (e_1 \times e_2) \circ \bigcup_{i=1}^l \left[E_x^i \cap E_y^i \cap V_{n(x,y)}^i \right] \quad (12.26)$$

where the symbol “ \times ” denotes as the Cartesian min operation between two fuzzy sets, and the symbol “ \circ ” represents the max–min composition operator between the two fuzzy relations.

In order to compensate for unexpected disturbances and time-varying parameters, a self-organizing fuzzy controller (adaptive FLC) is embedded in the sub-FLC and is also designed as a two-input single-output fuzzy controller to simplify the online calculation load. The inputs are designed as the error e_j and the change of the error Δe_j . The output will be used to tune the center of the output MFs of the sub-FLC. The self-organizing fuzzy control rules are in the form of

$$R^i: \text{IF } e_1 \text{ is } E_x^i \text{ AND } e_2 \text{ is } E_y^i, \text{ THEN } c_{jr} \text{ is } C_{m(x,y)}^i \quad (12.27)$$

where

e_1 , e_2 , E_x^i , and E_y^i have the same meanings as the description in the sub-FLC

c_{jr} is the change of the center of the output MFs of the sub-FLC

$C_{m(x,y)}^i$ is the linguistic variable of c_{jr}

Mamdani's minimum fuzzy implication and center average defuzzification are adopted. After the fuzzy rule adaptation, the center of output MFs of the sub-FLC $V_{n(x,y)}^i$ will be tuned as

$$V_{n(x,y)}^i \leftarrow V_{n(x,y)}^i + \frac{\sum_{x,y} \{ [\mu_{E_x^i}(e_1) \wedge \mu_{E_y^i}(e_2)] \cdot C_{m(x,y)}^i \}}{\sum_{x,y} [\mu_{E_x^i}(e_1) \wedge \mu_{E_y^i}(e_2)]} \quad (12.28)$$

where $\mu_{E_x^i}(e_1)$ is the fuzzy membership value of the current error in E_x^i , $\mu_{E_y^i}(e_2)$ the fuzzy membership value of the current change of the error in E_y^i , $\mu_{E_x^i}(e_1) \cap \mu_{E_y^i}(e_2)$ denotes the intersection of two fuzzy memberships $\mu_{E_x^i}(e_1)$ and $\mu_{E_y^i}(e_2)$, which is defined as the *min* operation between two fuzzy sets:

$$\mu_{E_x^i}(e_1) \cap \mu_{E_y^i}(e_2) = \min \left[\mu_{E_x^i}(e_1), \mu_{E_y^i}(e_2) \right]$$

With this adaptation mechanism, each adaptive fuzzy controller has the ability to tune the output MFs of the sub-FLC based on system performance. An improved system performance will be achieved and a limited number of fuzzy rules are adequate for general nonlinear time-varying systems.

12.3.2 PERPENDICULAR FUZZY CONTROL ENGINE

The perpendicular fuzzy control engine is designed to eliminate the input–output interaction effect in the MIMO system. As stated before, the control actions obtained from the horizontal fuzzy control engine are fuzzy sets, which represent the required control action for each individual input–output pair without any consideration on the multivariable interaction. These fuzzy variables are further aggregated in fuzzy domain to derive the individual control action based on the result of RGA analysis. For example, the change of the control action ΔU_r for the r th input is obtained from the aggregation of the q fuzzy control action $V_{1r}, V_{2r}, \dots, V_{qr}$ as

$$\Delta U_r = \bigcup_{j=1}^q V_{jr} \frac{1}{\lambda_{jr}} = \bigcup_{j=1}^q \frac{V_{jr}}{\lambda_{jr}} = \frac{V_{1r}}{\lambda_{1r}} + \frac{V_{2r}}{\lambda_{2r}} + \cdots + \frac{V_{qr}}{\lambda_{qr}} \quad (12.29)$$

where

$j = 1, 2, \dots, q$ is the total number of the system output

λ_{jr} is the (j, r) th element in the system RGA in Equations 12.11 and 12.14 for square systems and nonsquare systems, respectively

As explained in the previous section, the factor $1/\lambda_{jr}$ indicates how much the input–output gain changes for the $u_r - y_j$ pair due to the interaction effect from the other system variables. Term V_{jr} represents the required control action for the r th input from the j th output without interaction effect. Thus, V_{jr}/λ_{jr} is the control action for the r th input from the j th output with multivariable cross-coupling. And the summation of $\bigcup_{j=1}^q \frac{V_{jr}}{\lambda_{jr}}$ represents the required control action for the r th input from all the system outputs.

Center average defuzzification is performed to derive the crisp control action Δu_r as

$$\Delta u_r = \frac{\sum_{\Delta U_r} \mu(\Delta U_r) \Delta U_r}{\sum_{\Delta U_r} \mu(\Delta U_r)} \quad (12.30a)$$

for a discrete universe of discourse, where $\mu(\Delta U_r)$ represents the membership value in the output fuzzy set ΔU_r . If the universe of discourse is continuous, then the control action is calculated as

$$\Delta u_r = \frac{\int_{\Delta U_r} \mu(\Delta U_r) \Delta U_r d(\Delta U_r)}{\int_{\Delta U_r} \mu(\Delta U_r) d(\Delta U_r)} \quad (12.30b)$$

And the control action $u_r(k)$ is obtained in an incremental form as

$$u_r(k) = u_r(k - 1) + \Delta u_r(k) \quad (12.31)$$

The fuzzy controller is performed in discrete-time domain using the zero-order-hold (ZOH), which holds its input signal for a specified sampling period to get a continuous signal out. The p control actions are expressed in the vector form as

$$\mathbf{u}(k) = \mathbf{u}(k - 1) + \Delta \mathbf{u}(k) \quad (12.32)$$

12.4 STABILITY ANALYSIS

From Equations 12.26 and 12.28, the fuzzy control action from the first fuzzy control engine V_{jr} for the $u_r - y_j$ pair is derived as

$$\begin{aligned} V_{jr} &= (e_1 \times e_2) \circ \bigcup_{i=1}^l [E_x^i \cap E_y^i \cap \left(V_{n(x,y)}^i + \frac{\sum_{x,y} \{ [\mu_{E_x^i}(e_1) \cap \mu_{E_y^i}(e_2)] \cdot C_{m(x,y)}^i \}}{\sum_{x,y} [\mu_{E_x^i}(e_1) \cap \mu_{E_y^i}(e_2)]} \right)] \\ &= (e_1 \times e_2) \circ \left[\bigcup_{i=1}^l (E_x^i \cap E_y^i \cap V_{n(x,y)}^i) + \bigcup_{i=1}^l \left(E_x^i \cap E_y^i \cap \frac{\sum_{x,y} \{ [\mu_{E_x^i}(e_1) \cap \mu_{E_y^i}(e_2)] \cdot C_{m(x,y)}^i \}}{\sum_{x,y} [\mu_{E_x^i}(e_1) \cap \mu_{E_y^i}(e_2)]} \right) \right] \\ &= (e_1 \times e_2) \circ \left\{ \bigcup_{i=1}^l [E_x^i \cap E_y^i \cap V_{n(x,y)}^i] \right\} \\ &\quad + (e_1 \times e_2) \circ \left[\bigcup_{i=1}^l \left(E_x^i \cap E_y^i \cap \frac{\sum_{x,y} \{ [\mu_{E_x^i}(e_1) \cap \mu_{E_y^i}(e_2)] \cdot C_{m(x,y)}^i \}}{\sum_{x,y} [\mu_{E_x^i}(e_1) \cap \mu_{E_y^i}(e_2)]} \right) \right] \\ &= (e_1 \times e_2) \circ \left\{ \bigcup_{i=1}^l [E_x^i \cap E_y^i \cap V_{n(x,y)}^i] \right\} \\ &\quad + (e_1 \times e_2) \circ \left(\bigcup_{i=1}^l \{ E_x^i \cap E_y^i \cap (e_1 \times e_2) \circ [E_x^i \cap E_y^i \cap C_{m(x,y)}^i] \} \right) \\ &= (e_1 \times e_2) \circ \left\{ \bigcup_{i=1}^l [E_x^i \cap E_y^i \cap V_{n(x,y)}^i] \right\} + (e_1 \times e_2) \circ \left\{ \bigcup_{i=1}^l [E_x^i \cap E_y^i \cap C_{m(x,y)}^i] \right\} \\ &= (e_1 \times e_2) \circ \left(\bigcup_{i=1}^l \{ E_x^i \cap E_y^i \cap [V_{n(x,y)}^i + C_{m(x,y)}^i] \} \right) \end{aligned}$$

From the derivation and the stability proof in Section 10.2, the horizontal fuzzy control engine, which consists of the sub-FLC and the adaptive FLC for each system input–output pair, is Lyapunov stable in both continuous and discrete domains. And then, the change of control action ΔU_r is derived based on the aggregation of the q fuzzy control actions $V_{1r}, V_{2r}, \dots, V_{qr}$ for the r th input as shown in Equation 12.29:

$$\Delta U_r = \bigcup_{j=1}^q \frac{V_{jr}}{\lambda_{jr}} = \bigcup_{j=1}^q \frac{(e_1 \times e_2) \circ \left(\bigcup_{i=1}^l \left\{ E_x^i \cap E_y^i \cap [V_{n(x,y)}^i + C_{m(x,y)}^i] \right\} \right)}{\lambda_{jr}}$$

From Property 10.1, the parallel combination of the q fuzzy control actions is also stable, with the storage function being the summation of the q storage functions. Furthermore, from Property 9.2, if two passive subsystems are interconnected in a well-defined negative feedback connection, then the closed-loop system is stable. As proved in Section 10.2, the closed-loop multivariable system is Lyapunov stable as long as the augmented open-loop system is input–output stable as

$$V[\mathbf{x}(t_f)] - V[\mathbf{x}(t_0)] \leq \int_{t_0}^{t_f} \left[\int u_r(\sigma) d\sigma \right]^T y_j(\sigma) d\sigma, \quad \forall \mathbf{x}, \forall u_r \quad (12.33)$$

for each $u_r - y_j$ input–output pair in continuous domain and

$$V[\mathbf{x}(k+1)] - V[\mathbf{x}(k)] \leq \left[\sum u_r(i) \right]^T y_j(k), \quad \forall \mathbf{x}(k), \forall u_r(k), \forall k \quad (12.34)$$

in discrete domain, where \mathbf{x} is the state vector.

12.5 SIMULATION EXAMPLES

12.5.1 DISTILLATION COLUMN

Consider a multivariable side-stream distillation column, originally studied by Doukas and Luyben (1978):

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} -9.811e^{-1.59s} & 0.374e^{-7.75s} & -11.3e^{-3.79s} \\ \frac{11.36s+1}{11.36s+1} & \frac{(22.2s+1)^2}{(22.2s+1)^2} & \frac{(21.74s+1)^2}{(21.74s+1)^2} \\ \frac{5.984e^{-2.24s}}{14.29s+1} & \frac{-1.986e^{-0.71s}}{(66.67s+1)^2} & \frac{5.24e^{-60s}}{(400s+1)^2} \\ \frac{2.38e^{-0.42s}}{(1.43s+1)^2} & \frac{0.0204e^{-0.59s}}{(7.14s+1)^2} & \frac{-0.33e^{-0.68s}}{(2.38s+1)^2} \\ \frac{-11.67e^{-1.91s}}{12.19s+1} & \frac{-0.176e^{-0.48s}}{(6.9s+1)^2} & \frac{4.48e^{-0.52s}}{(11.11s+1)^2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad (12.35)$$

The four system outputs are the impurities concentrations: toluene in bottom y_1 , toluene in bottom y_2 , benzene in side draw y_3 , and benzene in side draw y_4 , which are controlled by three controlled variables: reboiler duty u_1 , reflux ratio u_2 , and side-stream flow rate u_3 . The system schematic diagram of this side-stream distillation column is shown in Figure 12.4.

Based on the interaction analysis method described in Sections 12.1 and 12.2, an FBFN model is constructed to model this multivariable system and the steady-state gain matrix is calculated as

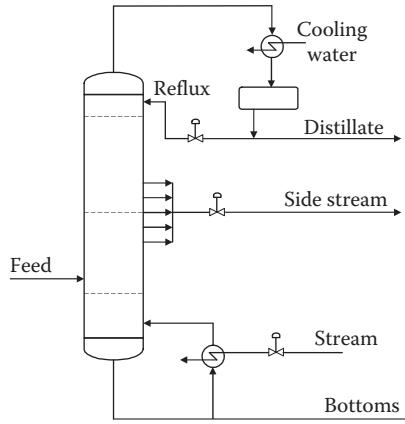


FIGURE 12.4 Schematic diagram of a multivariable side-stream distillation column. (From Doukas, N. and Luyben, W.L., *Anal. Instrum.*, 16, 51, 1978. With permission.)

$$\mathbf{G}(0) = \begin{bmatrix} -9.811 & 0.374 & -11.30 \\ 5.984 & -1.986 & 3.566 \\ 2.380 & 0.020 & -0.330 \\ -11.67 & -0.176 & 4.480 \end{bmatrix}$$

The system RGA is obtained as

$$\Lambda = \mathbf{G}(0) \times [\mathbf{G}(0)^+]^T = \begin{bmatrix} 0.242 & -0.077 & 0.834 \\ 0.006 & 1.060 & -0.066 \\ 0.028 & 0.001 & 0.002 \\ 0.725 & 0.017 & 0.230 \end{bmatrix}$$

It is well known that the RGA is one of the most systematic measures to quantify the multivariable cross-coupling effect. Each element represents the interaction effect of each input–output pair. From the description in Section 12.2, it is obvious that the third system output will be difficult to control since all elements in the third row of the array are small numbers, which are almost zero. This indicates that each system input has little influence on the third system output. The following simulation results will also illustrate the poor controllability of the third system output.

In order to benchmark this multivariable fuzzy controller, a nonsquare internal model controller (IMC) (Chang and Yu, 1990) is also executed to control this system (Figure 12.5), where the output vector $\mathbf{Y}(s)$ is expressed as

$$\mathbf{Y}(s) = \mathbf{C}(s)\{\mathbf{I} + \mathbf{C}(s)[\mathbf{G}(s) - \tilde{\mathbf{G}}(s)]\}^{-1}\mathbf{G}(s)\mathbf{Y}_d(s)$$

where

$\mathbf{Y}_d(s)$ represents the desired output vector
 $\tilde{\mathbf{G}}(s)$ is the system model

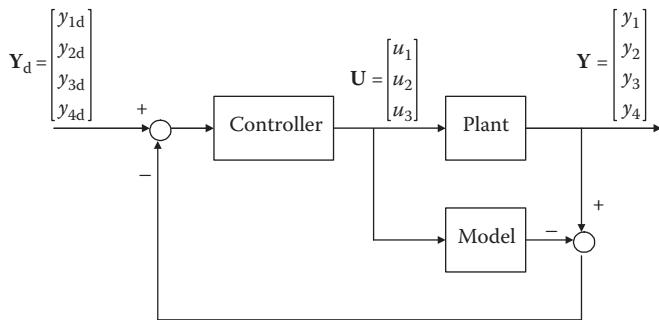


FIGURE 12.5 IMC for the distilled column. (From Chang, J.-W. and Yu, C.-C., *Chem. Eng. Sci.*, 45, 1309, 1990. With permission.)

Assume there is no plant-model mismatch, then

$$\mathbf{Y}(s) = \mathbf{C}(s)\mathbf{G}(s)\mathbf{Y}_d(s)$$

The closed-loop error is then calculated as

$$\mathbf{e}(s) = [\mathbf{I} - \mathbf{C}(s)\mathbf{G}(s)]\mathbf{Y}_d(s)$$

In order to minimize the steady-state error vector, the least-square perfect controller requires

$$\mathbf{C}(s) = [\mathbf{G}^T(0)\mathbf{G}(0)]^{-1}\mathbf{G}^T(0) = \mathbf{G}^+(0)$$

And the control actions under the least-square perfect control are calculated as

$$\mathbf{U}(s) = \mathbf{C}(s)\mathbf{Y}_d(s)$$

Next, simulation examples will be presented in three different cases to compare the control performance of the nonsquare IMC and the proposed multivariable fuzzy controller described in Section 12.3.

Case 12.1 Normal Operating Condition with a Unit Step Change in Each System Output

In this case, the normal operating condition with a unit step change is made for the four desired outputs y_{1d} , y_{2d} , y_{3d} , and y_{4d} , individually. The control performance of the nonsquare IMC and the proposed multivariable fuzzy controller is compared for each system output set-point change.

The upper plot in Figure 12.6 shows the output responses with a unit step change in y_{1d} , where the dashed lines are the desired outputs as $y_{1d} = 1$, $y_{2d} = 0$, $y_{3d} = 0$, and $y_{4d} = 0$, the dash-dot lines are the actual outputs with the nonsquare IMC and the solid

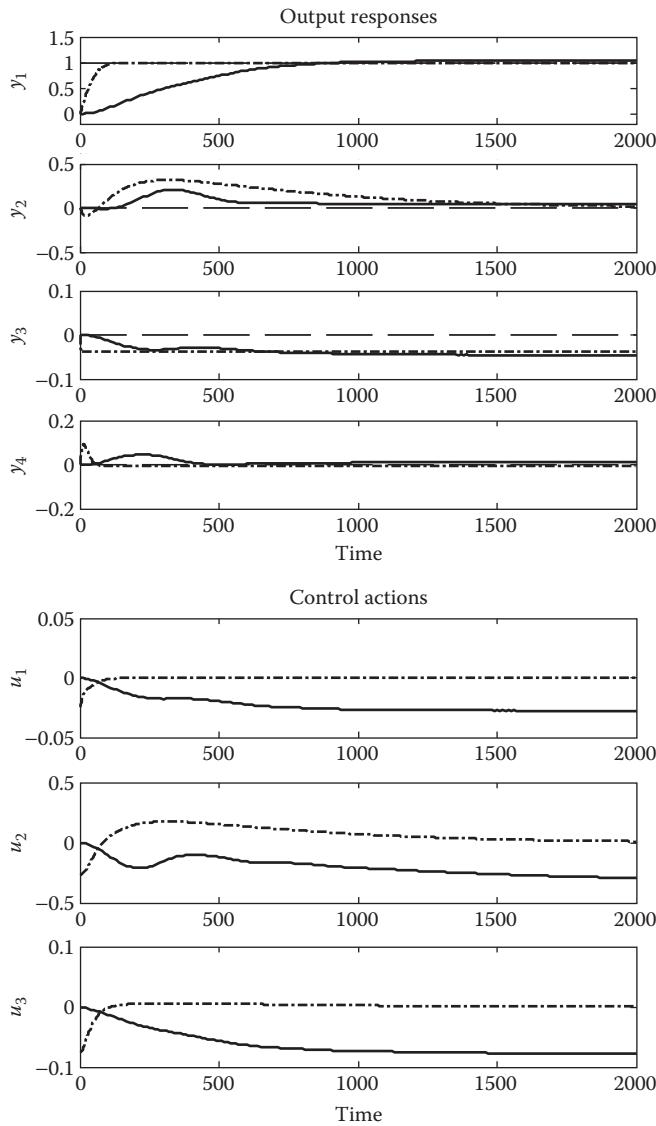


FIGURE 12.6 Control performance comparison for a unit step change in y_{1d} (dash-dot lines: nonsquare IMC, solid lines: multivariable fuzzy controller).

lines are the actual outputs with the multivariable fuzzy controller. The horizontal axes denote the simulation time in second and the vertical axes are the four system output signals. From the comparison, it can be seen that the actual outputs y_1 and y_4 approach the desired values in shorter transient time with the nonsquare IMC than with the multivariable fuzzy controller. However, the second output y_2 stabilizes to the desired

value y_{2d} in relative longer transient time with the nonsquare IMC, which is around 2000 s. For the third output y_3 , as stated previously, each element in the third row of the RGA is almost zero. Thus, y_3 has the worst output response, where the constant steady-state offset to the desired value is about -0.04 with both controllers. The lower plot in [Figure 12.6](#) is the required control actions, where the dash-dot lines represent those of the nonsquare IMC and the solid lines are those from the multivariable fuzzy controller.

Since the nonsquare IMC is designed in order to minimize the error vector in steady state, it is natural that the two-norm of the error signals with the nonsquare IMC is smaller than with the proposed multivariable fuzzy controller, which can be observed in the output responses in the upper plot in [Figure 12.6](#). However, the design of the nonsquare IMC depends on the availability of the system model, which is not always obtainable for complex real-world systems. In comparison, the advantage of the multivariable fuzzy controller is that no mathematical system model is required and the control laws are designed and tuned automatically based on system performance. The resultant transient time and steady-state error are at an acceptable level compared with the nonsquare IMC perfect controller. Later, further simulation examples will be provided to illustrate the superiority of the proposed multivariable fuzzy controller in compensating for the unknown system model variations.

[Figure 12.7](#) shows the comparison of the system performance of the two controllers with a unit step change in the desired value of the second output y_{2d} , which are denoted as the dashed line $y_{1d} = 0$, $y_{2d} = 1$, $y_{3d} = 0$, and $y_{4d} = 0$ in the upper plot. The output responses are similar for these two controllers except that the second output y_2 takes little longer time to stabilize to the desired value with the multivariable fuzzy controller. The other three actual output responses are about the same and the steady-state errors are around zero.

The next case is when a unit step change is designated to the third output y_{3d} as shown in [Figure 12.8](#). As indicated in the third row of the RGA of this multivariable system, the output y_3 is difficult to control with both controllers, where the steady-state error is around $1 - 0.03 = 0.97$ with the nonsquare IMC, while it is about $1 - 0.05 = 0.95$ with the multivariable fuzzy controller as shown in the third figure of the output responses. The significant errors of the actual output y_3 with the desired output y_{3d} exist due to the poor controllability of this output variable. The magnitudes of the other three steady-state errors are similar between these two controllers and in a reasonable range compared to the third case.

The fourth case is made with a unit step change in y_{4d} and is shown in [Figure 12.9](#). In this case, both the nonsquare IMC and multivariable fuzzy controller can stabilize the fourth output y_4 with negligible error in a short transient time. The magnitudes of the steady-state errors of the other three output variables are similar and within an acceptable range.

In all of the simulation results in this case, little steady-state errors can be observed in y_1 , y_2 , and y_4 when unit step changes are made in these desired outputs. However, a significant error exists in y_3 for a unit step set-point change. This phenomenon is indicated by the RGA elements and can be generalized to general multivariable systems.

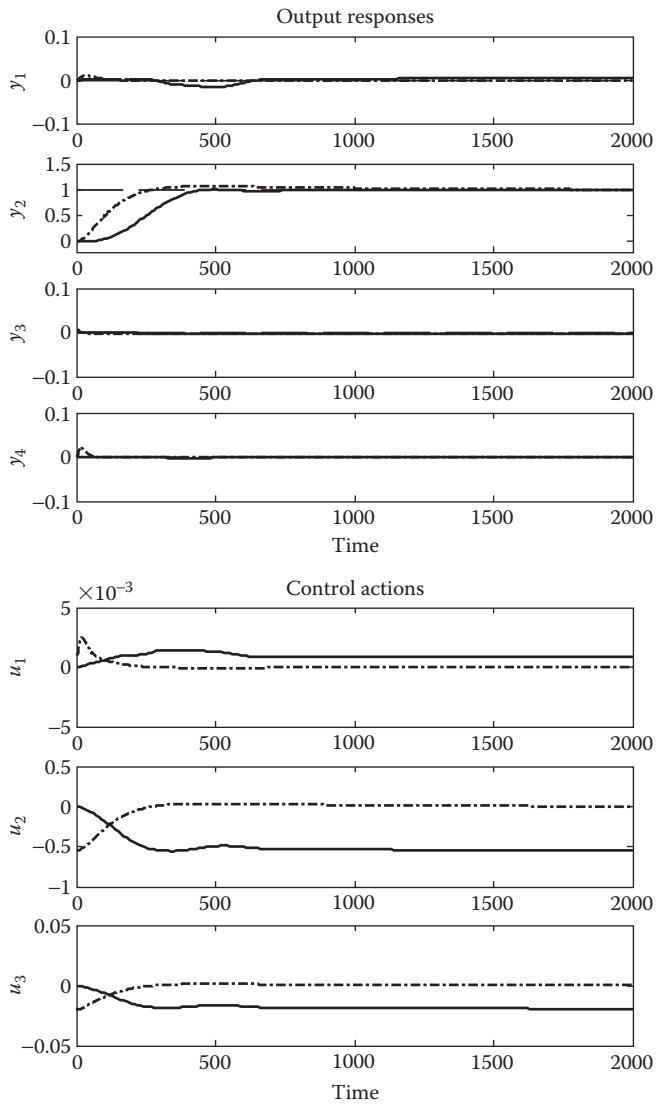


FIGURE 12.7 Control performance comparison for a unit step change in y_{2d} (dash-dot lines: nonsquare IMC, solid lines: multivariable fuzzy controller).

Case 12.2 Normal Operating Condition with a Unit Step Change in All System Outputs

In this case, the simulation is run under the normal operating condition and the system outputs are prescribed as $y_{1d} = 1$, $y_{2d} = 1$, $y_{3d} = 1$, and $y_{4d} = 1$, which are shown as the dashed lines in the upper plot in Figure 12.10. Both controllers can stabilize y_1 and y_2 with negligible steady-state error values, except that the multivariable fuzzy controller takes a relatively longer transient time than the nonsquare

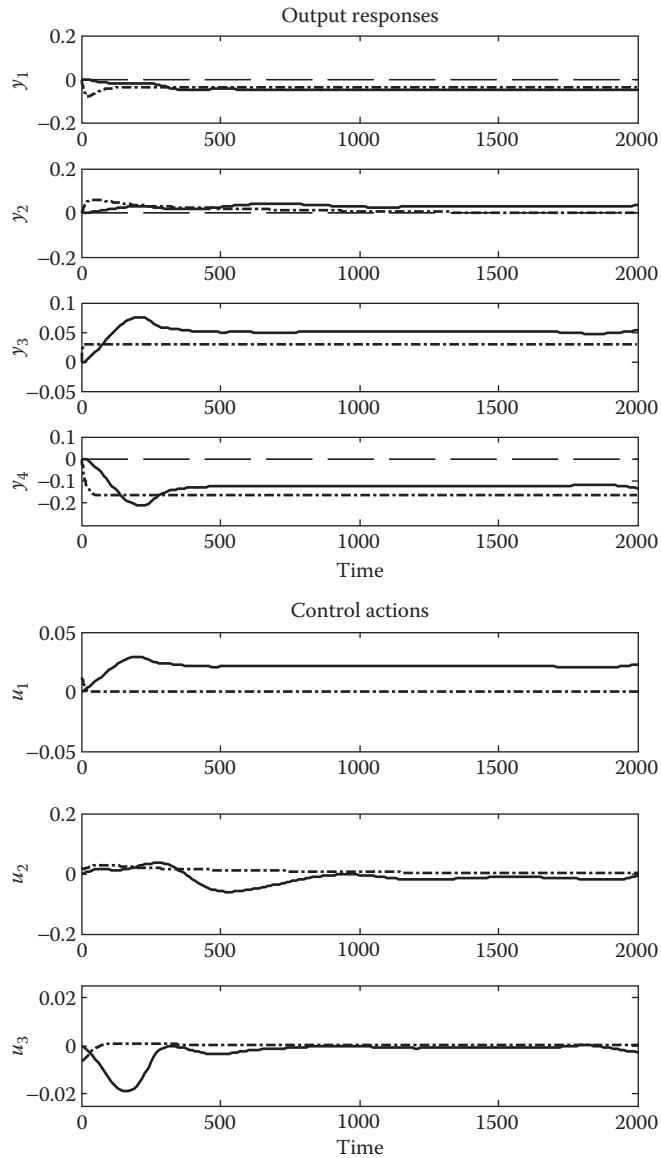


FIGURE 12.8 Control performance comparison for a unit step change in y_{3d} (dash-dot lines: nonsquare IMC, solid lines: multivariable fuzzy controller).

IMC. The third system output y_3 is poorly controlled and the steady-state errors are about 1.2 for both the controllers, as is shown in the third figure of the output responses. With the nonsquare IMC, there exists a steady-state error for the fourth output y_4 and the magnitude is around 0.2. In contrast, the multivariable fuzzy controller can stabilize the fourth output y_4 in a short transient time with about zero steady-state error.

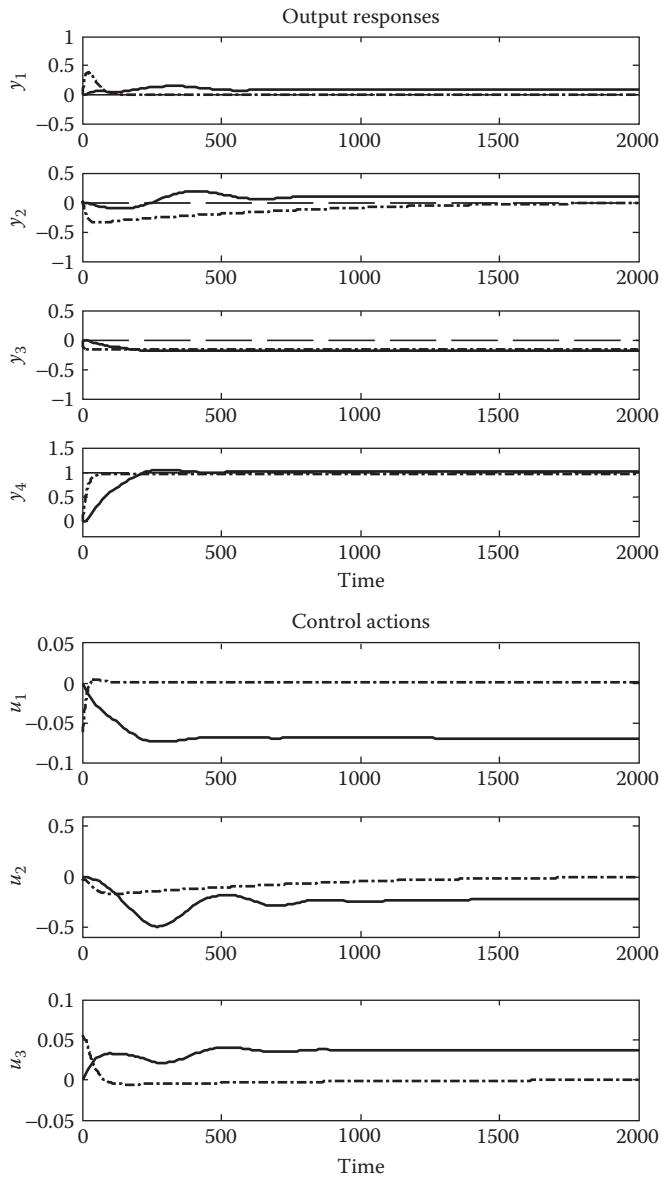


FIGURE 12.9 Control performance comparison for a unit step change in y_{4d} (dash-dot lines: nonsquare IMC, solid lines: multivariable fuzzy controller).

Case 12.3 Model Variations on Input–Output Gains with a Unit Step Change in All System Outputs

The design of the nonsquare IMC depends on the availability of the system mathematical model, which is not always obtainable for complex real-world systems.

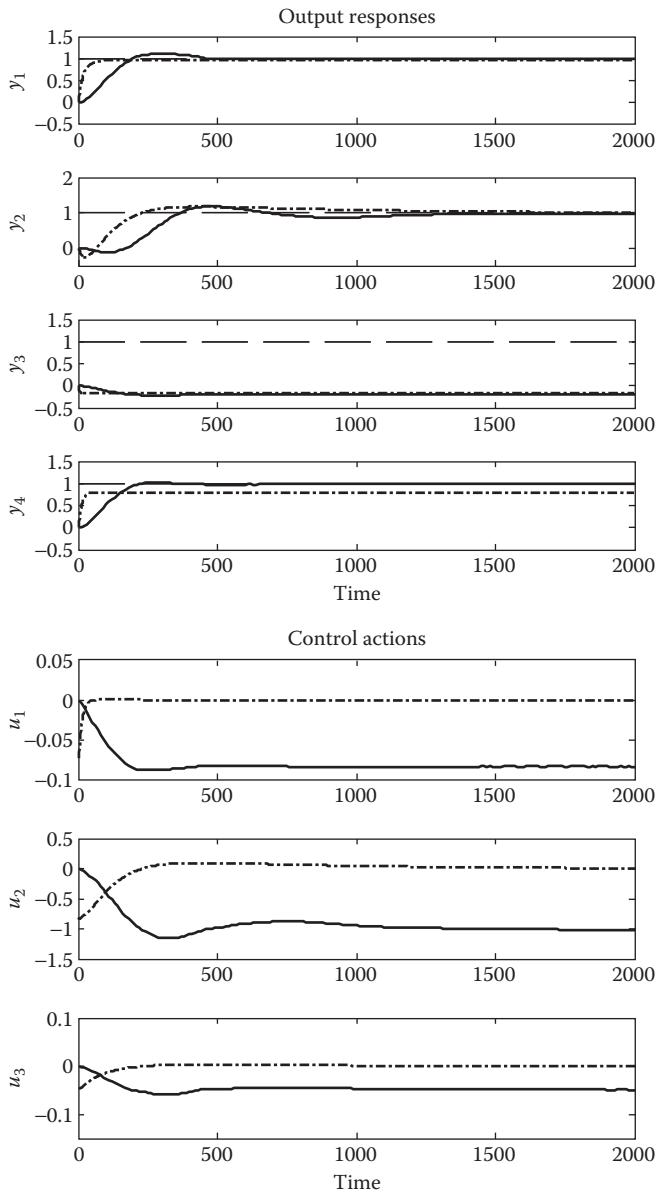


FIGURE 12.10 Control performance comparison for unit step changes in y_{1d} , y_{2d} , y_{3d} , and y_{4d} (dash-dot lines: nonsquare IMC, solid lines: multivariable fuzzy controller).

In comparison, the advantage of the multivariable fuzzy controller is that no mathematical model is required and the control rules are determined and tuned automatically based on system performance. The following two cases illustrate the effectiveness of the proposed multivariable fuzzy controller in compensating for unknown system variations.

As described by Doukas and Luyben (1978), the transfer functions of this multi-variable distillation column (Equation 12.35) are obtained by approximately fitting low-order transfer functions to the Bode plots generated by Fourier transformation of the impulse response data. Suppose some discrepancies exist in the system input-output gain parameters and the system transfer function matrix is identified as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \frac{-13e^{-1.59s}}{11.36s + 1} & \frac{0.5e^{-7.75s}}{(22.2s + 1)^2} & \frac{-14.7e^{-3.79s}}{(21.74s + 1)^2} \\ \frac{7.8e^{-2.24s}}{14.29s + 1} & \frac{-2.6e^{-0.71s}}{(66.67s + 1)^2} & \frac{6.81e^{-60s}}{(400s + 1)^2} \\ \frac{3.1e^{-0.42s}}{(1.43s + 1)^2} & \frac{0.1e^{-0.59s}}{(7.14s + 1)^2} & \frac{-0.5e^{-0.68s}}{(2.38s + 1)^2} \\ \frac{-15.17e^{-1.91s}}{12.19s + 1} & \frac{-0.229e^{-0.48s}}{(6.9s + 1)^2} & \frac{5.82e^{-0.52s}}{(11.11s + 1)^2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad (12.36)$$

The desired system outputs are specified as $y_{1d} = 1$, $y_{2d} = 1$, $y_{3d} = 1$, and $y_{4d} = 1$ as in the previous case. The system performance of both controllers is shown in [Figure 12.11](#). Since the nonsquare IMC is designed based on the mathematical model of the multivariable system under normal condition, it is incapable of compensating for the unknown system variations. The steady-state errors of y_1 , y_2 , and y_4 are obviously larger than the errors with the multivariable fuzzy controller. Owing to the embedded self-tuning ability of the proposed multivariable fuzzy controller, the unknown model variations on input-output gains can be successfully compensated, where the steady-state errors of y_1 , y_2 , and y_4 are around zero. Due to the poor controllability of the third system output, the actual output y_3 cannot be regulated to the desired value y_{3d} and the value of the steady-state error is around 1.3 for both controllers.

Case 12.4 Model Variations on System Poles and Input–Output Delays with a Unit Step Change in All System Outputs

This case is designed to demonstrate the adaptability of the proposed multivariable fuzzy control scheme under a drastic change in system dynamics. Assume the system model variations exist in both system pole locations and input–output delays, and the system transfer function is described as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \frac{-9.811e^{-2s}}{8.7s + 1} & \frac{0.374e^{-10s}}{(17s + 1)^2} & \frac{-11.3e^{-5s}}{(16.7s + 1)^2} \\ \frac{5.984e^{-2.9s}}{10s + 1} & \frac{-1.986e^{-0.92s}}{(50s + 1)^2} & \frac{5.24e^{-78s}}{(310s + 1)^2} \\ \frac{2.38e^{-0.55s}}{(1.1s + 1)^2} & \frac{0.0204e^{-0.8s}}{(5.5s + 1)^2} & \frac{-0.33e^{-s}}{(1.8s + 1)^2} \\ \frac{-11.67e^{-2.5s}}{10s + 1} & \frac{-0.176e^{-0.62s}}{(5.3s + 1)^2} & \frac{4.48e^{-0.68s}}{(8.5s + 1)^2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad (12.37)$$

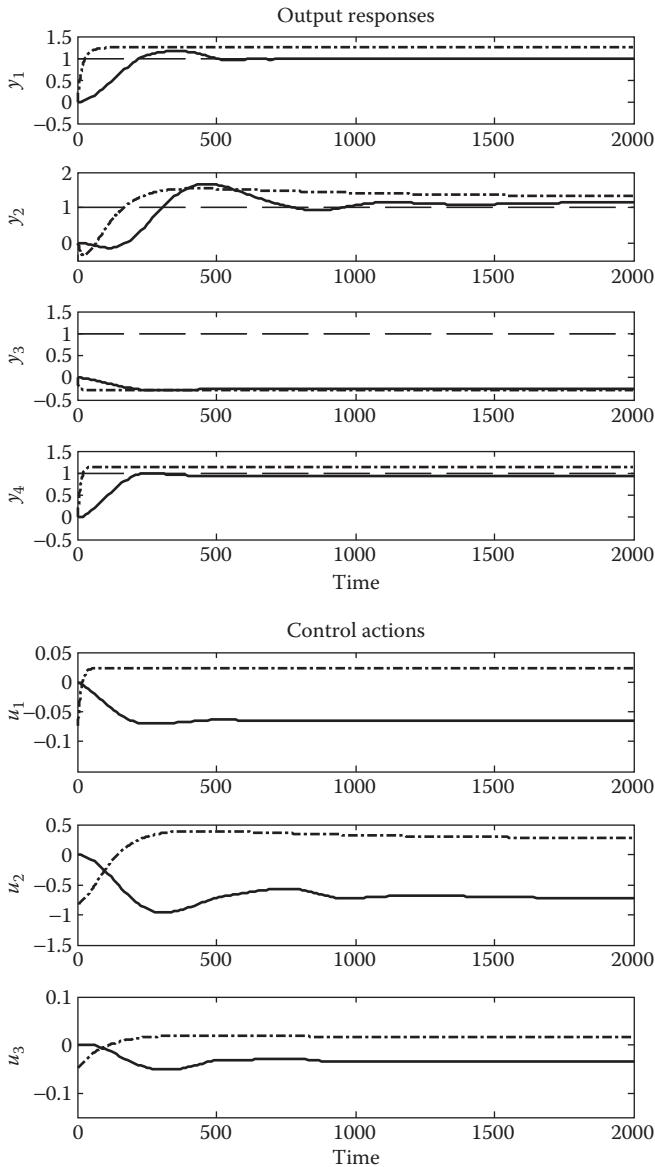


FIGURE 12.11 Control performance comparison for unit step changes in y_{1d} , y_{2d} , y_{3d} , and y_{4d} with model variations on system input–output gain (dash-dot lines: nonsquare IMC, solid lines: multivariable fuzzy controller).

Same as in the previous case, the desired system outputs are set as $y_{1d} = 1$, $y_{2d} = 1$, $y_{3d} = 1$, and $y_{4d} = 1$. The output responses and the control actions are shown in [Figure 12.12](#). The first output response y_1 with the nonsquare IMC has a higher oscillation

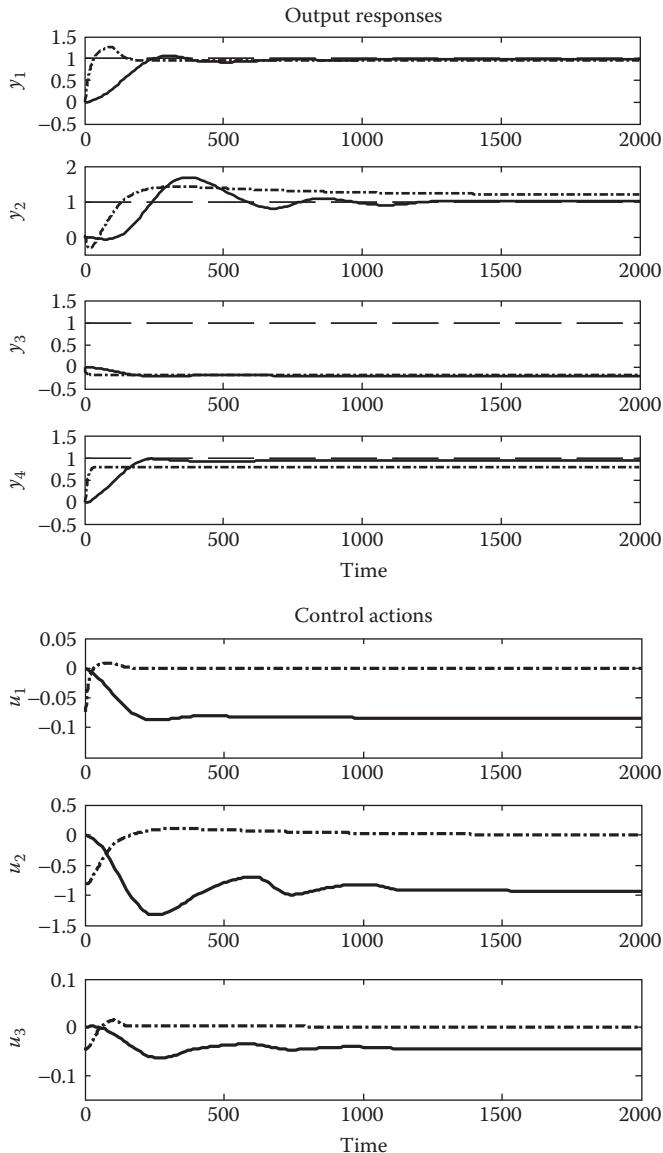


FIGURE 12.12 Control performance comparison for unit step changes in y_{1d} , y_{2d} , y_{3d} , and y_{4d} with model variations on system poles and input–output delays (dash-dot lines: nonsquare IMC, solid lines: multivariable fuzzy controller).

peak value than that with the multivariable fuzzy controller. The steady-state errors for y_2 and y_4 with the nonsquare IMC are larger than the errors with the multivariable fuzzy controller. The third output y_3 cannot be controlled with both controllers and the steady-state error is around 1.2.

12.5.2 CHEMICAL PRESSURE TANK SYSTEM

This example is a multivariable pressure tank system, where the air flows with a regulated supply rate. The system outputs are the pressure in the tank and the outlet flow rate. Control valves are installed on both the inlet and outlet of the tank. The system inputs are the fractional openings of the inlet valve l_i and of the outlet valve l_o . The system equations are identified as

$$\begin{aligned}\dot{P}_v &= \frac{K_{vi}l_i}{V}\sqrt{\frac{RT}{M}}\sqrt{P_i(P_i - P_v)} - \frac{K_{vo}l_o}{V}\sqrt{\frac{RT}{M}}\sqrt{P_o(P_v - P_o)} \\ \dot{V}_o &= K_{vol_o}\sqrt{\frac{RT}{M}}\sqrt{\frac{P_v - P_o}{P_o}}\end{aligned}\quad (12.38)$$

where

P_v , P_i , and P_o are the pressures in the pressure tank, inlet, and outlet, respectively ($P_i = 373$ kPa, $P_o = 0.101$ kPa)

V_o is the flow volume at the outlet

V is the tank volume ($V = 0.000215$ m³)

K_{vi} is the flow parameter for the inlet valve

K_{vo} is the flow parameter for the outlet valve

l_i is the fractional opening of the inlet valve

l_o is the fractional opening of the outlet valve

R , T , and M are the gas constant, temperature, and molecular weight of the gas in the tank, respectively ($R = 8314.4$ J/kmol K, $T = 1^\circ\text{C}$, $M = 29$ kg/kmol)

The model parameters are given in Table 12.1.

An FBFN model is constructed to represent this multivariable pressure tank system and the steady-state gain matrix is calculated as

$$\mathbf{G}(0) = \begin{bmatrix} 899.7901 & -451.1720 \\ 0.1245 & 0.0011 \end{bmatrix}$$

The system RGA is obtained as

$$\mathbf{A} = \mathbf{G}(0) \times [\mathbf{G}(0)^+]^T = \begin{bmatrix} 0.0177 & 0.9823 \\ 0.9823 & 0.0177 \end{bmatrix}\quad (12.39)$$

The system schematic diagram is shown in [Figure 12.13](#).

TABLE 12.1
Model Parameters for the Chemical Pressure Tank System

P_i	Pressure in the inlet	373	(kPa)
P_o	Pressure in the outlet	0.101	(kPa)
V	Tank volume	0.000215	(m ³)
R	Gas constant	8314.4	(J/kmol)
T	Temperature	1	(°C)
M	Molecular weight	29	(kg/kmol)

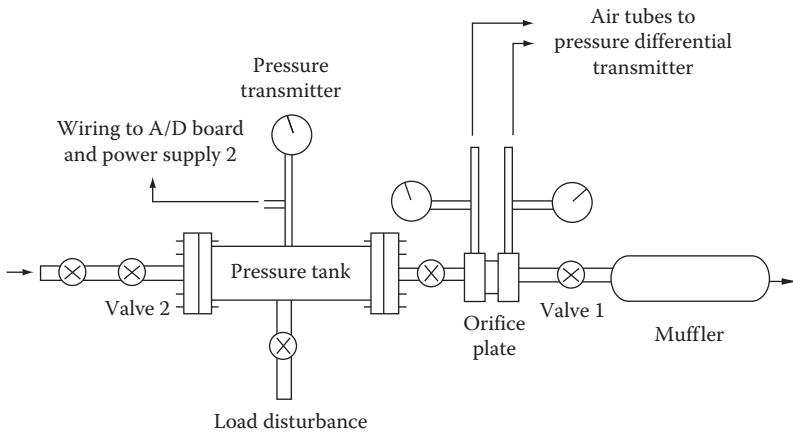


FIGURE 12.13 Chemical pressure tank system. (From Guo, B., Jiang, A., Hua, X., and Jutan, A., *Chem. Eng. Sci.*, 56, 6781, 2001. With permission.)

The relationships between the system outputs and the valve openings are highly nonlinear and interactive. In order to compare the control performance with the multivariable fuzzy controller, a nonlinear adaptive control strategy (Guo et al., 2001) is used to control this multivariable system as well. This nonlinear control scheme combines the generic model control (GMC) algorithm with an online nonlinear observer to estimate the system parameter changes. The control actions are obtained as

$$\begin{bmatrix} l_i \\ l_o \end{bmatrix} = G^{-1} \left[K_1 \left(\begin{bmatrix} P_v \\ V_o \end{bmatrix}_{sp} - \begin{bmatrix} P_v \\ V_o \end{bmatrix} \right) + K_2 \int_0^t \left(\begin{bmatrix} P_v \\ V_o \end{bmatrix}_{sp} - \begin{bmatrix} P_v \\ V_o \end{bmatrix} \right) d\tau \right]$$

with $G = \begin{bmatrix} \frac{K_{vi}}{V} \sqrt{\frac{RT}{M}} \sqrt{P_i(P_i - P_v)} & -\frac{K_{vo}}{V} \sqrt{\frac{RT}{M}} \sqrt{P_o(P_v - P_o)} \\ 0 & K_{vo} \sqrt{\frac{RT}{M}} \sqrt{\frac{P_v - P_o}{P_o}} \end{bmatrix}$

(12.40)

where

$\begin{bmatrix} l_i \\ l_o \end{bmatrix}$ are the system control actions

$\begin{bmatrix} P_v \\ V_o \end{bmatrix}_{sp}$ are the desired system outputs

$\begin{bmatrix} P_v \\ V_o \end{bmatrix}$ are the actual system outputs

$K_1 = \begin{bmatrix} k_{11} & 0 \\ 0 & k_{12} \end{bmatrix}$ and $K_2 = \begin{bmatrix} k_{21} & 0 \\ 0 & k_{22} \end{bmatrix}$ are the control gain matrices, which are determined based on the desired system closed-loop performance, such as requirements on transient time and oscillation degree

Since the valve hysteresis is not directly measurable, some mismatch may exist between the actual system and the model, for example, the values of K_{vi} and K_{vo} change with the position variations of the inlet and outlet valves. The process/model mismatch will inevitably deteriorate the control performance. Thus, a nonlinear online observer is developed to estimate the model parameters as

$$\begin{bmatrix} \dot{\hat{P}}_v \\ \dot{\hat{V}}_o \\ \dot{\hat{K}}_{vi} \\ \dot{\hat{K}}_{vo} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{l_i}{V} \sqrt{\frac{RT}{M}} \sqrt{P_i(P_i - P_v)} & -\frac{l_o}{V} \sqrt{\frac{RT}{M}} \sqrt{P_o(P_v - P_o)} \\ 0 & 0 & 0 & l_o \sqrt{\frac{RT}{M}} \sqrt{\frac{P_v - P_o}{P_o}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{P}_v \\ \hat{V}_o \\ \hat{K}_{vi} \\ \hat{K}_{vo} \end{bmatrix} - \begin{bmatrix} 2\lambda & 0 \\ 0 & 2\lambda \\ \frac{V}{l_i} \sqrt{\frac{M}{RT}} \frac{\lambda^2}{\sqrt{P_i(P_i - P_v)}} & \sqrt{\frac{M}{RT}} \frac{P_o \lambda^2}{\sqrt{P_i(P_i - P_v)}} \\ 0 & \frac{\lambda^2}{l_o} \sqrt{\frac{M}{RT}} \sqrt{\frac{P_o}{P_v - P_o}} \end{bmatrix} \begin{bmatrix} \hat{P}_v - P_v \\ \hat{V}_o - V_o \end{bmatrix} \quad (12.41)$$

where

\hat{P}_v , \hat{V}_o , \hat{K}_{vi} , and \hat{K}_{vo} are the estimated values of P_v , V_o , K_{vi} , and K_{vo}
 $\lambda = 0.95$ is the design parameter for the observer, which is a compromise
between the output fast convergence and system performance sensitivity to
noise

The two diagonal matrices in the GMC algorithm are determined as
 $K_1 = \begin{bmatrix} 1.1111 & 0 \\ 0 & 0.086 \end{bmatrix}$ and $K_2 = \begin{bmatrix} 0.1667 & 0 \\ 0 & 0.01929 \end{bmatrix}$. The developed nonlinear adaptive control structure is shown in Figure 12.14. The sampling time Δt in the simulation is specified as 1 s for both controllers.

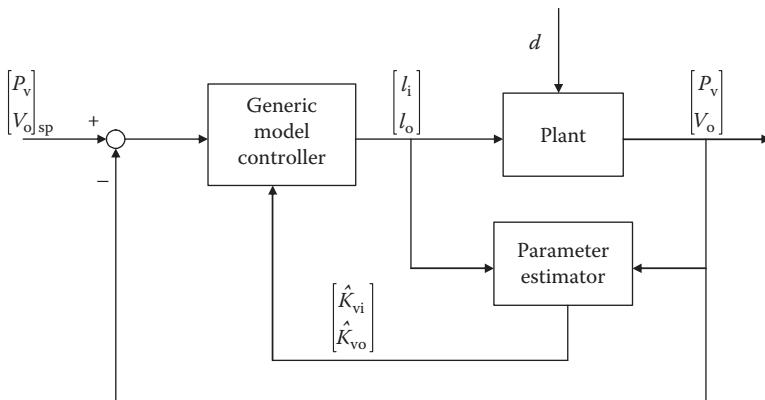


FIGURE 12.14 Nonlinear adaptive control system.

Case 12.5 Normal Operating Condition

In this case, the set point for the tank pressure varies from 13.7 to 11.3 kPa at $t = 250$ s and then back to 13.7 kPa at $t = 800$ s. The set point for the outlet flow rate also changes from 30 to 50 m^3/ks at $t = 250$ s and then back to 30 m^3/ks at $t = 800$ s. This case intends to compare the ability of these two controllers in achieving set-point tracking performance during normal operating condition. Figure 12.15 shows the

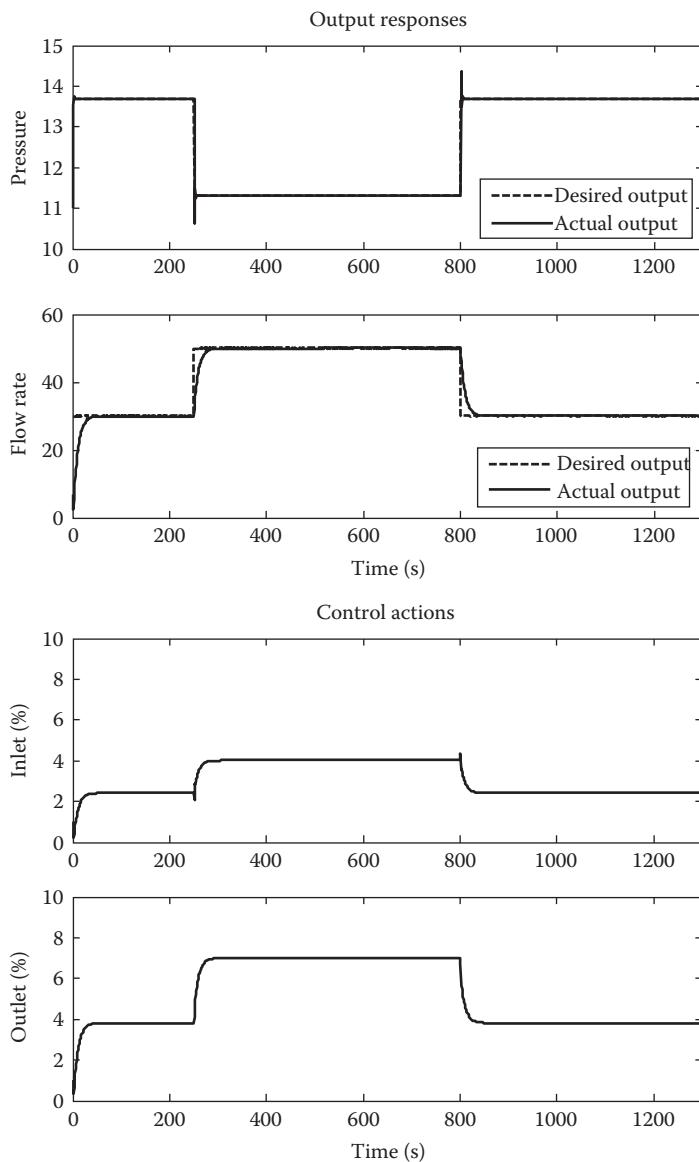


FIGURE 12.15 Nonlinear adaptive control result in Case 12.5.

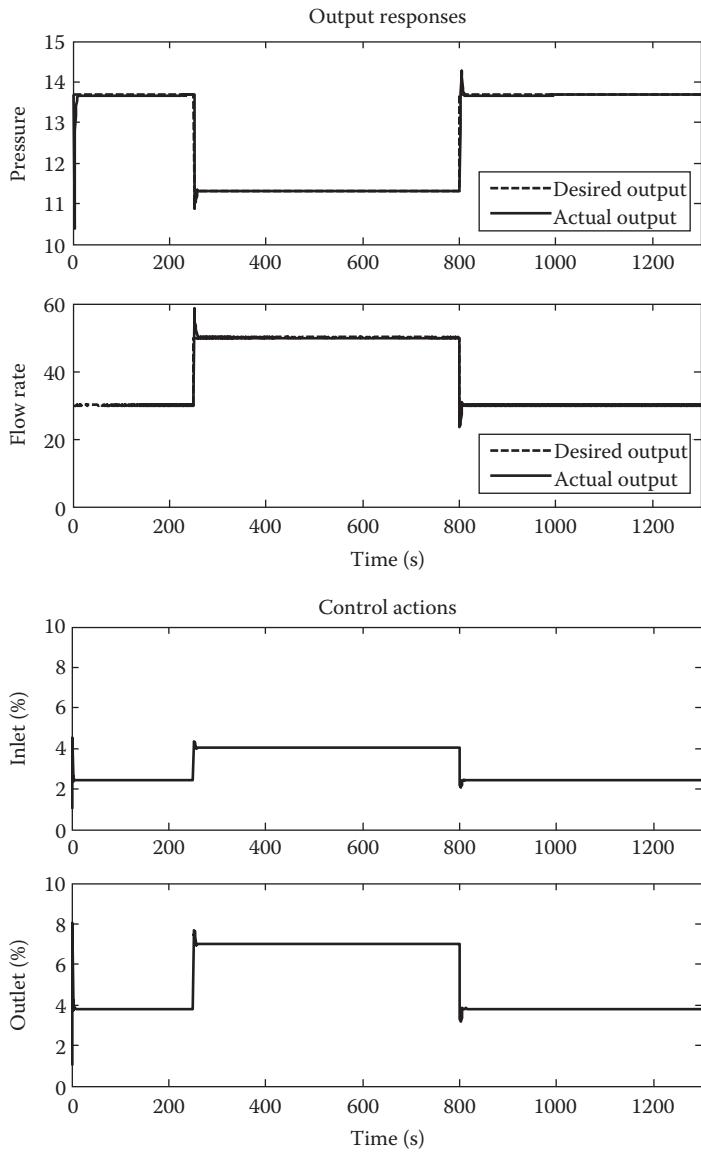


FIGURE 12.16 Multivariable fuzzy control result in Case 12.5.

control result with the nonlinear adaptive controller and Figure 12.16 is the result with the multivariable fuzzy controller. The control results illustrate the satisfactory decoupling performance of both controllers, where the system output signals are able to follow the desired set-point variations with both controllers. From the comparison, it can be seen that the transient time of the outlet flow rate with the multivariable

fuzzy controller is shorter than the case with the nonlinear adaptive controller, which demonstrates the superiority ability of the multivariable fuzzy controller in controlling a multivariable nonlinear system with cross-coupling and keeping all the system outputs follow the desired set-point values.

Case 12.6 Load Disturbance Rejection

This example case is designed to compare the controllers' ability in rejecting unknown external load disturbance during process operation. Two load disturbances are introduced in the inlet and outlet of the tank as sinusoidal signals with different frequencies as

$$\begin{aligned} d_1(t) &= 0.01 \times \sin(0.02\pi t) \\ d_2(t) &= 0.07 \times \sin(0.06\pi t) \end{aligned} \quad (12.42)$$

The output responses in [Figure 12.17](#) are the simulation results with the nonlinear adaptive controller, which exhibit large oscillations in the tank pressure output and detectable variations in the outlet flow rate. In comparison, [Figure 12.18](#) is the result with the multivariable fuzzy controller, which demonstrates its superiority in rejecting the load disturbances due to its embedded adaptation mechanism. When the desired output signals change, there is a short-term oscillation in the output responses. And then, the multivariable fuzzy controller brings the system back to the set points rather quickly and there is no oscillation during the steady-state operation. This result illustrates that the multivariable fuzzy control system is robust and converges fast to the desired trajectory under the existence of unknown exterior disturbances.

Case 12.7 Sensor Noise Elimination

This example case considers the system operation with sensor noises in the output measurements, where the sensor noises are added to the system outputs as the feedback signal as

$$\begin{aligned} s_1(t) &= 0.1 \times (2 \times \text{rand} - 1) \\ s_2(t) &= 0.1 \times (2 \times \text{rand} - 1) \end{aligned} \quad (12.43)$$

The simulation results are shown in [Figures 12.19](#) and [12.20](#) for the nonlinear adaptive controller and the multivariable fuzzy controller, respectively, where both system responses are able to follow the desired reference output signals. For the second output of the outlet flow rate, the transient time with the multivariable fuzzy controller is shorter than that with the nonlinear adaptive controller, and the signal noisy variation is smaller with the multivariable fuzzy controller compared to that with the nonlinear adaptive controller. For the first output of the tank pressure, the signal noisy variations are similar with both controllers. This simulation example demonstrates the effectiveness of the multivariable fuzzy controller in handling

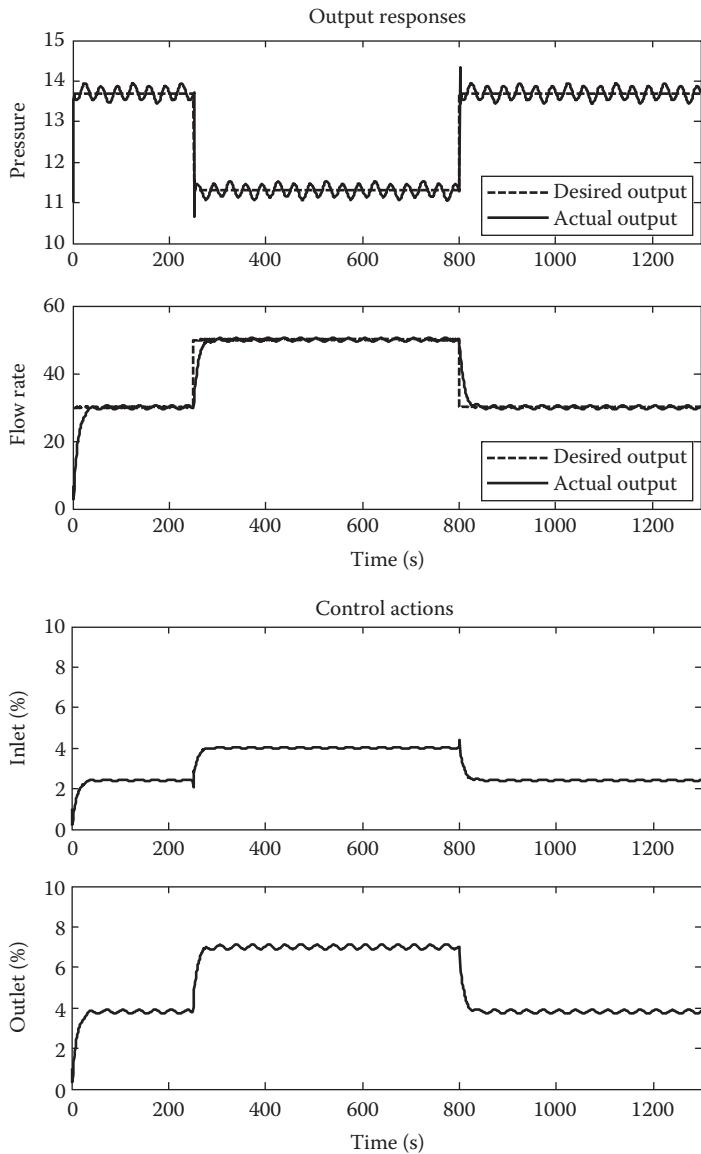


FIGURE 12.17 Nonlinear adaptive control result in Case 12.6.

multivariable interaction and following multiple output trajectories under the condition with unexpected sensor noise. Due to the random and nonperiodical properties of the sensor noise, the control ability of the multivariable fuzzy controller in sensor noise elimination is limited and there still exist detectable noisy fluctuations in the output signals.

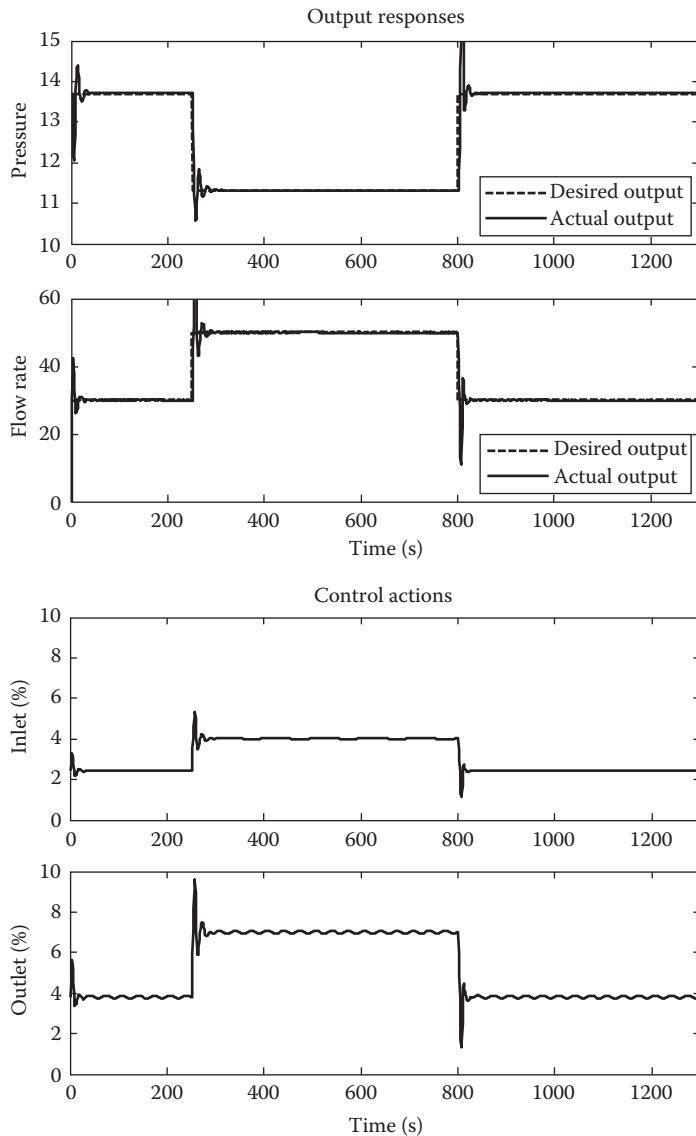


FIGURE 12.18 Multivariable fuzzy control result in Case 12.6.

12.6 CONCLUSION

In this chapter, a multivariable fuzzy control strategy has been developed for a general MIMO nonlinear system with no mathematical model requirement, as long as each system's input–output pair has an input–output monotonic relationship or a piecewise monotonic relationship. The system dynamics was modeled in fuzzy

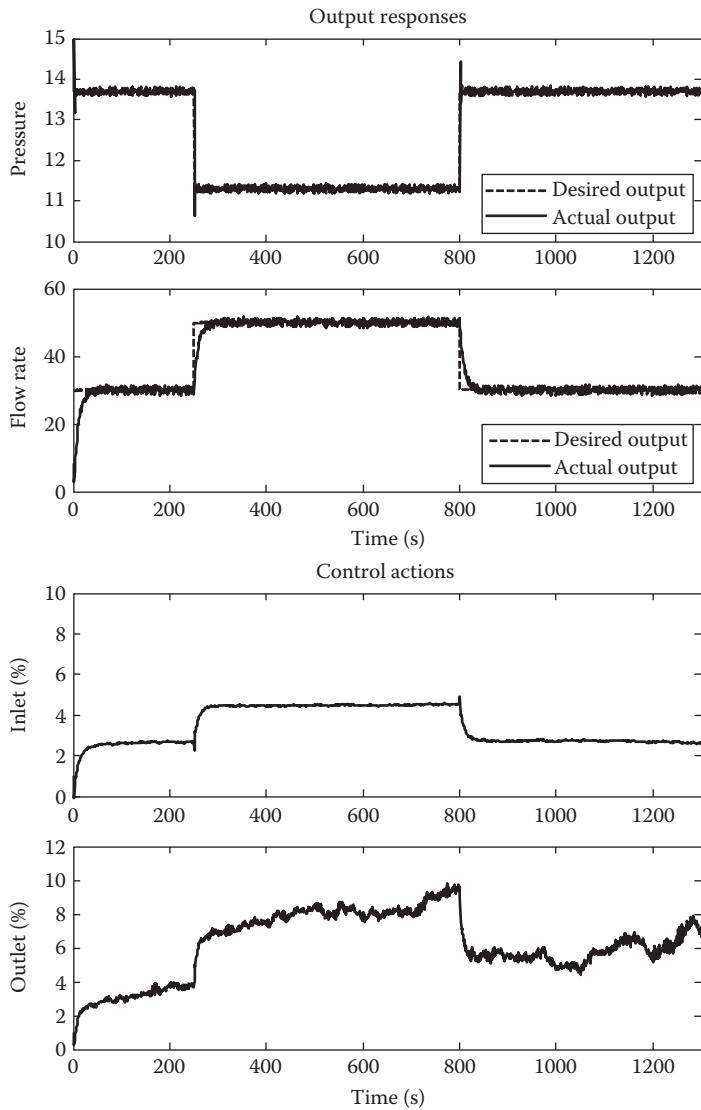


FIGURE 12.19 Nonlinear adaptive control result in Case 12.7.

domain and the multivariable cross-coupling effect was analyzed. Based on the obtained fuzzy model and the interaction degree, a multivariable fuzzy controller was designed with an embedded hierarchical structure for each input–output pair to compensate for system uncertainties and time-varying parameters. The multivariable interaction was also eliminated through the aggregation of the multiple control actions from each system input–output pair based on the multivariable interaction analysis. The stability of the closed-loop fuzzy control system was proved using the

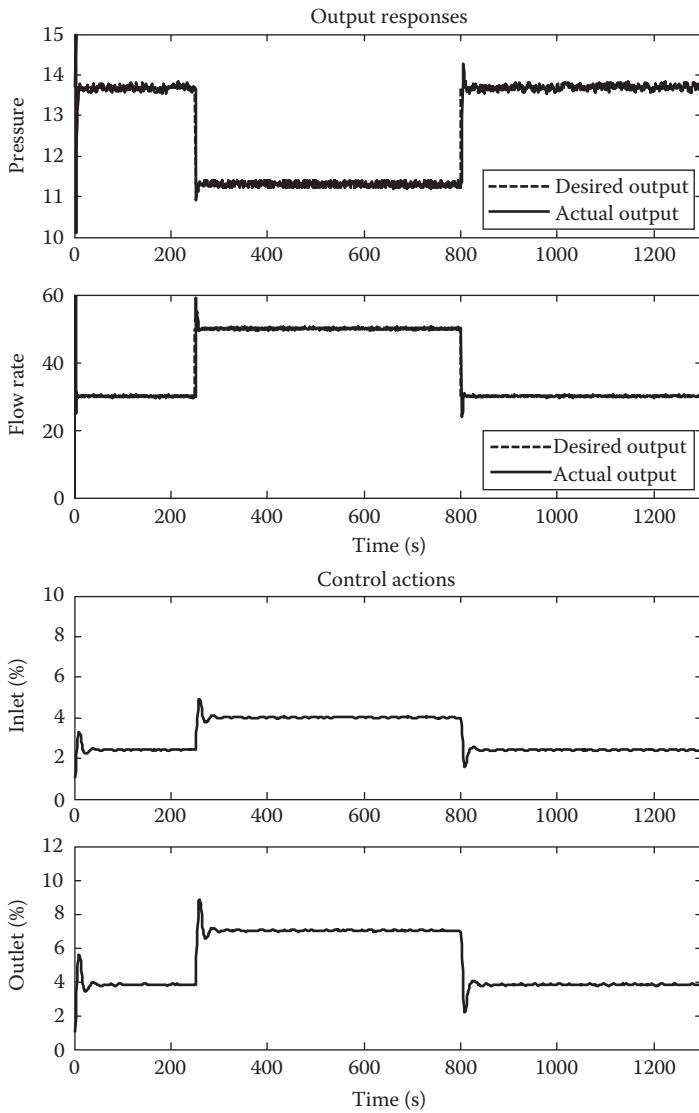


FIGURE 12.20 Multivariable fuzzy control result in Case 12.7.

input–output passivity theory and can be guaranteed for a general system whose augmented open-loop system is input–output passive. Finally, the multivariate fuzzy control strategy was applied to two simulation examples, a distillation column and a chemical pressure tank, and satisfactory control results were obtained in both cases, which demonstrate the effectiveness of the multivariable fuzzy control strategy under various operating conditions.

REFERENCES

- Aracil, J. and Gordillo, F., *Stability Issues in Fuzzy Control*, Physica-Verlag, New York, 2000.
- Bristol, E., On a new measure of interaction for multivariable process control, *IEEE Transaction on Automatic Control*, 11(1): 133–134, 1966.
- Calcev, G., Some remarks in the stability of Mamdani fuzzy control systems, *IEEE Transactions on Fuzzy Systems*, 6(3): 436–442, 1998.
- Chang, J.-W. and Yu, C.-C., The relative gain for non-square multivariable systems, *Chemical Engineering Science*, 45(5): 1309–1323, 1990.
- Deshpande, P.B., *Multivariable Process Control*, Instrument Society of America, Research Triangle Park, NC, 1989.
- Doukas, N. and Luyben, W.L., Control of sidestream columns separating ternary mixture, *Analysis Instrumentation*, 16: 51–58, 1978.
- Farinwata, S.S., Filev, D., and Langari, R., *Fuzzy Control Synthesis and Analysis*, John Wiley & Sons Ltd., New York, 2000.
- Graybill, F.A., *Introduction to Matrices with Applications in Statistics*, Wadsworth, Belmont, CA, 1969.
- Grossdidier, P., Morari, M., and Holt, B.R., Closed-loop properties from steady-state gain information, *Industrial and Engineering Chemistry Process Design and Development*, 24: 221–235, 1985.
- Guo, B., Jiang, A., Hua, X., and Jutan, A., Nonlinear adaptive control for multivariable chemical processes, *Chemical Engineering Science*, 56: 6781–6791, 2001.
- Kinnaert, M., Interaction measures and pairing of controlled and manipulated variables for multi-input-multi-output systems: A survey, *Journal A*, 36(4): 15–23, 1995.
- Lee, C.W. and Shin, Y.C., Construction of fuzzy systems using least-squares method and genetic algorithm, *Fuzzy Sets and Systems*, 13: 297–323, 2003.
- McAvoy, T.J., *Interaction Analysis—Principles and Applications*, Instrument Society of America, Research Triangle Park, NC, 1983.
- Morari, M. and Zafiriou, E., *Robust Process Control*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- Reeves, D.E. and Arkun, Y., Interaction measures for nonsquare decentralized control structures, *American Institute of Chemical Engineers Journal*, 35(4): 603–613, 1988.
- Shinskey, F.G., *Process Control Systems Application, Design and Tuning*, McGraw-Hill, New York, 1996.
- Skodestad, S., Dynamics and control of distillation columns, *Chemical Engineering Research and Design, Transactions of the Institute of Chemical Engineers*, 75: 539–562, 1997.
- Skogestad, S. and Morari, M., Implications of large RGA elements on control performance, *Industrial and Engineering Chemistry Research*, 26: 2029–2330, 1987.
- Wang, L.X. and Mendel, J.M., Fuzzy basis functions, universal approximation, and orthogonal least-squares learning, *IEEE Transactions on Neural Networks*, 3(5): 807–814, 1992.
- Xu, C. and Shin, Y.C., Design of a multilevel fuzzy controller for nonlinear systems and stability analysis, *IEEE Transactions on Fuzzy Systems*, 13(6): 761–778, 2005.
- Xu, C. and Shin, Y.C., Interaction analysis for MIMO nonlinear systems based on a fuzzy basis function network model, *Fuzzy Sets and Systems*, 158: 2013–2025, 2007.
- Ying, H., Practical design of nonlinear fuzzy controllers with stability analysis for regulating processes with unknown mathematical models, *Automatica*, 30(7): 1185–1195, 1994.