Fuzzy Control

Kevin M. Passino
Department of Electrical Engineering
The Ohio State University

Stephen Yurkovich
Department of Electrical Engineering
The Ohio State University

ADDISON-WESLEY

An Imprint of Addison-Wesley Longman, Inc.

Menlo Park, California • Reading, Massachusetts • Harlow, England • Berkeley, California
Don Mills, Ontaria • Sydney • Bonn • Amsterdam • Mexico City

Assistant Editor: Laura Cheu
Editorial Assistant: Royden Tonomura
Senior Production Editor: Teri Hyde
Marketing Manager: Rob Merino
Manufacturing Supervisor: Janet Weaver
Art and Design Manager: Kevin Berry
Cover Design: Yvo Riezebos (technical drawing by K. Passino)
Text Design: Peter Vacek
Design Macro Writer: William Erik Baxter
Copyeditor: Brian Jones
Proofreader: Holly McLean-Aldis

Library of Congress Cataloging-in-Publication Data

```
Passino, Kevin M.
Fuzzy control / Kevin M. Passino and Stephen Yurkovich.
p. cm.
Includes bibliographical references and index.
ISBN 0-201-18074-X
1. Automatic control. 2. Control theory. 3. Fuzzy systems.
I. Yurkovich, Stephen. II. Title.
TJ213.P317    1997                                          97-14003
629.8'9--DC21                                                   CIP
```


About the Cover: An explanation of the technical drawing is given in Chapter 2 on page 50.

ISBN 0–201–18074–X
1 2 3 4 5 6 7 8 9 10—CRW—01 00 99 98 97

Addison Wesley Longman, Inc., 2725 Sand Hill Road, Menlo Park, California 94025

To Annie and Juliana (K.M.P)

To Tricia, B.J., and James (S.Y.)

Preface

Fuzzy control is a practical alternative for a variety of challenging control applications since it provides a convenient method for constructing nonlinear controllers via the use of heuristic information. Such heuristic information may come from an operator who has acted as a "human-in-the-loop" controller for a process. In the fuzzy control design methodology, we ask this operator to write down a set of rules on how to control the process, then we incorporate these into a fuzzy controller that emulates the decision-making process of the human. In other cases, the heuristic information may come from a control engineer who has performed extensive mathematical modeling, analysis, and development of control algorithms for a particular process. Again, such expertise is loaded into the fuzzy controller to automate the reasoning processes and actions of the expert. Regardless of where the heuristic control knowledge comes from, fuzzy control provides a user-friendly formalism for representing and implementing the ideas we have about how to achieve high-performance control.

In this book we provide a control-engineering perspective on fuzzy control. We are concerned with both the construction of nonlinear controllers for challenging real-world applications and with gaining a fundamental understanding of the dynamics of fuzzy control systems so that we can mathematically verify their properties (e.g., stability) before implementation. We emphasize engineering evaluations of performance and comparative analysis with conventional control methods. We introduce adaptive methods for identification, estimation, and control. We examine numerous examples, applications, and design and implementation case studies throughout the text. Moreover, we provide introductions to neural networks, genetic algorithms, expert and planning systems, and intelligent autonomous control, and explain how these topics relate to fuzzy control.

Overall, we take a pragmatic engineering approach to the design, analysis, performance evaluation, and implementation of fuzzy control systems. We are not concerned with whether the fuzzy controller is "artificially intelligent" or with investigating the mathematics of fuzzy sets (although some of the exercises do), but

rather with whether the fuzzy control methodology can help solve challenging real-world problems.

Overview of the Book

The book is basically broken into three parts. In Chapters 1–4 we cover the basics of "direct" fuzzy control (i.e., the nonadaptive case). In Chapters 5–7 we cover adaptive fuzzy systems for estimation, identification, and control. Finally, in Chapter 8 we briefly cover the main areas of intelligent control and highlight how the topics covered in this book relate to these areas. Overall, we largely focus on what one could call the "heuristic approach to fuzzy control" as opposed to the more recent mathematical focus on fuzzy control where stability analysis is a major theme.

In Chapter 1 we provide an overview of the general methodology for conventional control system design. Then we summarize the fuzzy control system design process and contrast the two. Next, we explain what this book is about via a simple motivating example. In Chapter 2 we first provide a tutorial introduction to fuzzy control via a two-input, one-output fuzzy control design example. Following this we introduce a general mathematical characterization of fuzzy systems and study their fundamental properties. We use a simple inverted pendulum example to illustrate some of the most widely used approaches to fuzzy control system design. We explain how to write a computer program to simulate a fuzzy control system, using either a high-level language or Matlab[1]. In the web and ftp pages for the book we provide such code in C and Matlab. In Chapter 3 we use several case studies to show how to design, simulate, and implement a variety of fuzzy control systems. In these case studies we pay particular attention to comparative analysis with conventional approaches. In Chapter 4 we show how to perform stability analysis of fuzzy control systems using Lyapunov methods and frequency domain–based stability criteria. We introduce nonlinear analysis methods that can be used to predict and eliminate steady-state tracking error and limit cycles. We then show how to use the analysis approaches in fuzzy control system design. The overall focus for these nonlinear analysis methods is on understanding fundamental problems that can be encountered in the design of fuzzy control systems and how to avoid them.

In Chapter 5 we introduce the basic "function approximation problem" and show how identification, estimation, prediction, and some control design problems are a special case of it. We show how to incorporate heuristic information into the function approximator. We show how to form rules for fuzzy systems from data pairs and show how to train fuzzy systems from input-output data with least squares, gradient, and clustering methods. And we show how one clustering method from fuzzy pattern recognition can be used in conjunction with least squares methods to construct a fuzzy model from input-output data. Moreover, we discuss hybrid approaches that involve a combination of two or more of these methods. In Chapter 6 we introduce adaptive fuzzy control. First, we introduce several methods for automatically synthesizing and tuning a fuzzy controller, and then we illustrate their application via several design and implementation case studies. We also show how

1. MATLAB is a registered trademark of The MathWorks, Inc.

to tune a fuzzy model of the plant and use the parameters of such a model in the on-line design of a controller. In Chapter 7 we introduce fuzzy supervisory control. We explain how fuzzy systems can be used to automatically tune proportional-integral-derivative (PID) controllers, how fuzzy systems provide a methodology for constructing and implementing gain schedulers, and how fuzzy systems can be used to coordinate the application and tuning of conventional controllers. Following this, we show how fuzzy systems can be used to tune direct and adaptive fuzzy controllers. We provide case studies in the design and implementation of fuzzy supervisory control.

In Chapter 8 we summarize our control engineering perspective on fuzzy control, provide an overview of the other areas of the field of "intelligent control," and explain how these other areas relate to fuzzy control. In particular, we briefly cover neural networks, genetic algorithms, knowledge-based control (expert systems and planning systems), and hierarchical intelligent autonomous control.

Examples, Applications, and Design and Implementation Case Studies

We provide several design and implementation case studies for a variety of applications, and many examples are used throughout the text. The basic goals of these case studies and examples are as follows:

- To help illustrate the theory.
- To show how to apply the techniques.
- To help illustrate design procedures in a concrete way.
- To show what practical issues are encountered in the development and implementation of a fuzzy control system.

Some of the more detailed applications that are studied in the chapters and their accompanying homework problems are the following:

- *Direct fuzzy control:* Translational inverted pendulum, fuzzy decision-making systems, two-link flexible robot, rotational inverted pendulum, and machine scheduling (Chapters 2 and 3 homework problems: translational inverted pendulum, automobile cruise control, magnetic ball suspension system, automated highway system, single-link flexible robot, rotational inverted pendulum, machine scheduling, motor control, cargo ship steering, base braking control system, rocket velocity control, acrobot, and fuzzy decision-making systems).
- *Nonlinear analysis:* Inverted pendulum, temperature control, hydrofoil controller, underwater vehicle control, and tape drive servo (Chapter 4 homework problems: inverted pendulum, magnetic ball suspension system, temperature control, and hydrofoil controller design).

- *Fuzzy identification and estimation:* Engine intake manifold failure estimation, and failure detection and identification for internal combustion engine calibration faults (Chapter 5 homework problems: tank identification, engine friction estimation, and cargo ship failures estimation).
- *Adaptive fuzzy control:* Two-link flexible robot, cargo ship steering, fault tolerant aircraft control, magnetically levitated ball, rotational inverted pendulum, machine scheduling, and level control in a tank (Chapter 6 homework problems: tanker and cargo ship steering, liquid level control in a tank, rocket velocity control, base braking control system, magnetic ball suspension system, rotational inverted pendulum, and machine scheduling).
- *Supervisory fuzzy control:* Two-link flexible robot, and fault-tolerant aircraft control (Chapter 7 homework problems: liquid level control, and cargo and tanker ship steering).

Some of the applications and examples are dedicated to illustrating one idea from the theory or one technique. Others are used in several places throughout the text to show how techniques build on one another and compare to each other. Many of the applications show how fuzzy control techniques compare to conventional control methodologies.

World Wide Web Site and FTP Site: Computer Code Available

The following information is available electronically:

- Various versions of C and Matlab code for simulation of fuzzy controllers, fuzzy control systems, adaptive fuzzy identification and estimation methods, and adaptive fuzzy control systems (e.g., for some examples and homework problems in the text).
- Other special notes of interest, including an errata sheet if necessary.

You can access this information via the web site:

http://www.awl.com/cseng/titles/0-201-18074-X

or you can access the information directly via anonymous ftp to

ftp://ftp.aw.com/cseng/authors/passino/fc

For anonymous ftp, log into the above machine with a username "anonymous" and use your e-mail address as a password.

Organization, Prerequisites, and Usage

Each chapter includes an overview, a summary, and a section "For Further Study" that explains how the reader can continue study in the topical area of the chapter. At the end of each chapter overview, we explain how the chapter is related to the

others. This includes an outline of what must be covered to be able to understand the later chapters and what may be skipped on a first reading. The summaries at the end of each chapter provide a list of all major topics covered in that chapter so that it is clear what should be learned in each chapter.

Each chapter also includes a set of exercises or design problems and often both. Exercises or design problems that are particularly challenging (considering how far along you are in the text) or that require you to help define part of the problem are designated with a star ("$\star$") after the title of the problem. In addition to helping to solidify the concepts discussed in the chapters, the problems at the ends of the chapters are sometimes used to introduce new topics. We require the use of computer-aided design (CAD) for fuzzy controllers in many of the design problems at the ends of the chapters (e.g., via the use of Matlab or some high-level language).

The necessary background for the book includes courses on differential equations and classical control (root locus, Bode plots, Nyquist theory, lead-lag compensation, and state feedback concepts including linear quadratic regulator design). Courses on nonlinear stability theory and adaptive control would be helpful but are not necessary. Hence, much of the material can be covered in an undergraduate course. For instance, one could easily cover Chapters 1–3 in an undergraduate course as they require very little background besides a basic understanding of signals and systems including Laplace and z-transform theory (one application in Chapter 3 does, however, require a cursory knowledge of the linear quadratic regulator). Also, many parts of Chapters 5–7 can be covered once a student has taken a first course in control (a course in nonlinear control would be helpful for Chapter 4 but is not necessary). One could cover the basics of fuzzy control by adding parts of Chapter 2 to the end of a standard undergraduate or graduate course on control. Basically, however, we view the book as appropriate for a first-level graduate course in fuzzy control.

We have used the book for a portion (six weeks) of a graduate-level course on intelligent control and for undergraduate independent studies and design projects. In addition, portions of the text have been used for short courses and workshops on fuzzy control where the focus has been directed at practicing engineers in industry.

Alternatively, the text could be used for a course on intelligent control. In this case, the instructor could cover the material in Chapter 8 on neural networks and genetic algorithms after Chapter 2 or 3, then explain their role in the topics covered in Chapters 5, 6, and 7 while these chapters are covered. For instance, in Chapter 5 the instructor would explain how gradient and least squares methods can be used to train neural networks. In Chapter 6 the instructor could draw analogies between neural control via the radial basis function neural network and the fuzzy model reference learning controller. Also, for indirect adaptive control, the instructor could explain how, for instance, the multilayer perceptron or radial basis function neural networks can be used as the nonlinearity that is trained to act like the plant. In Chapter 7 the instructor could explain how neural networks can be trained to serve as gain schedulers. After Chapter 7 the instructor could then cover the material on expert control, planning systems, and intelligent autonomous control in Chapter 8. Many more details on strategies for teaching the material in a fuzzy or intelligent

control course are given in the instructor's manual, which is described below.

Engineers and scientists working in industry will find that the book will serve nicely as a "handbook" for the development of fuzzy control systems, and that the design, simulation, and implementation case studies will provide very good insights into how to construct fuzzy controllers for specific applications. Researchers in academia and elsewhere will find that this book will provide an up-to-date view of the field, show the major approaches, provide good references for further study, and provide a nice outlook for thinking about future research directions.

Instructor's Manual

An Instructor's Manual to accompany this textbook is available (to instructors only) from Addison Wesley Longman. The Instructor's Manual contains the following:

- Strategies for teaching the material.
- Solutions to end-of-chapter exercises and design problems.
- A description of a laboratory course that has been taught several times at The Ohio State University which can be run in parallel with a lecture course that is taught out of this book.
- An electronic appendix containing the computer code (e.g., C and Matlab code) for solving many exercises and design problems.

Sales Specialists at Addison Wesley Longman will make the instructor's manual available to qualified instructors. To find out who your Addison Wesley Longman Sales Specialist is please see the web site:

http://www.aw.com/cseng/

or send an email to:

cseng@aw.com

Feedback on the Book

It is our hope that we will get the opportunity to correct any errors in this book; hence, we encourage you to provide a precise description of any errors you may find. We are also open to your suggestions on how to improve the textbook. For this, please use either e-mail (passino@ee.eng.ohio-state.edu) or regular mail to the first author: Kevin M. Passino, Dept. of Electrical Engineering, The Ohio State University, 2015 Neil Ave., Columbus, OH 43210-1272.

Acknowledgments

No book is written in a vacuum, and this is especially true for this one. We must emphasize that portions of the book appeared in earlier forms as conference papers, journal papers, theses, or project reports with our students here at Ohio

State. Due to this fact, these parts of the text are sometimes a combination of our words and those of our students (which are very difficult to separate at times). In every case where we use such material, the individuals have given us permission to use it, and we provide the reader with a reference to the original source since this will typically provide more details than what are covered here. While we always make it clear where the material is taken from, it is our pleasure to highlight these students' contributions here as well. In particular, we drew heavily from work with the following students and papers written with them (in alphabetical order): Anthony Angsana [4], Scott C. Brown [27], David L. Jenkins [83], Waihon Andrew Kwong [103, 104, 144], Eric G. Laukonen [107, 104], Jeffrey R. Layne [110, 113, 112, 114, 111], William K. Lennon [118], Sashonda R. Morris [143], Vivek G. Moudgal [145, 144], Jeffrey T. Spooner [200, 196], and Moeljono Widjaja [235, 244]. These students, and Mehmet Akar, Mustafa K. Guven, Min-Hsiung Hung, Brian Klinehoffer, Duane Marhefka, Matt Moore, Hazem Nounou, Jeff Palte, and Jerry Troyer helped by providing solutions to several of the exercises and design problems and these are contained in the instructor's manual for this book. Manfredi Maggiore helped by proofreading the manuscript. Scott C. Brown and Raúl Ordóñez assisted in the development of the associated laboratory course at OSU.

We would like to gratefully acknowledge the following publishers for giving us permission to use figures that appeared in some of our past publications: The Institute of Electrical and Electronic Engineers (IEEE), John Wiley and Sons, Hemisphere Publishing Corp., and Kluwer Academic Publishers. In each case where we use a figure from a past publication, we give the full reference to the original paper, and indicate in the caption of the figure that the copyright belongs to the appropriate publisher (via, e.g., "© IEEE").

We have benefited from many technical discussions with many colleagues who work in conventional and intelligent control (too many to list here); most of these persons are mentioned by referencing their work in the bibliography at the end of the book. We would, however, especially like to thank Zhiqiang Gao and Oscar R. González for class-testing this book. Moreover, thanks go to the following persons who reviewed various earlier versions of the manuscript: D. Aaronson, M.A. Abidi, S.P. Colombano, Z. Gao, O. González, A.S. Hodel, R. Langari, M.S. Stachowicz, and G. Vachtsevanos.

We would like to acknowledge the financial support of National Science Foundation grants IRI-9210332 and EEC-9315257, the second of which was for the development of a course and laboratory for intelligent control. Moreover, we had additional financial support from a variety of other sponsors during the course of the development of this textbook, some of whom gave us the opportunity to apply some of the methods in this text to challenging real-world applications, and others where one or both of us gave a course on the topics covered in this book. These sponsors include Air Products and Chemicals Inc., Amoco Research Center, Battelle Memorial Institute, Delphi Chassis Division of General Motors, Ford Motor Company, General Electric Aircraft Engines, The Center for Automotive Research (CAR) at The Ohio State University, The Center for Intelligent Transportation

Research (CITR) at The Ohio State University, and The Ohio Aerospace Institute (in a teamed arrangement with Rockwell International Science Center and Wright Laboratories).

We would like to thank Tim Cox, Laura Cheu, Royden Tonomura, Teri Hyde, Rob Merino, Janet Weaver, Kevin Berry, Yvo Riezebos, Peter Vacek, William Erik Baxter, Brian Jones, and Holly McLean-Aldis for all their help in the production and editing of this book. Finally, we would most like to thank our wives, who have helped set up wonderful supportive home environments that we value immensely.

Kevin Passino
Steve Yurkovich
Columbus, Ohio
July 1997

Contents

CHAPTER 1

Introduction

It is not only old and early impressions
that deceive us; the charms of novelty
have the same power.
–Blaise Pascal

1.1 Overview

When confronted with a control problem for a complicated physical process, a control engineer generally follows a relatively systematic design procedure. A simple example of a control problem is an automobile "cruise control" that provides the automobile with the capability of regulating its own speed at a driver-specified set-point (e.g., 55 mph). One solution to the automotive cruise control problem involves adding an electronic controller that can sense the speed of the vehicle via the speedometer and actuate the throttle position so as to regulate the vehicle speed as close as possible to the driver-specified value (the design objective). Such speed regulation must be accurate even if there are road grade changes, head winds, or variations in the number of passengers or amount of cargo in the automobile.

After gaining an intuitive understanding of the plant's dynamics and establishing the design objectives, the control engineer typically solves the cruise control problem by doing the following:

1. Developing a model of the automobile dynamics (which may model vehicle and power train dynamics, tire and suspension dynamics, the effect of road grade variations, etc.).

2. Using the mathematical model, or a simplified version of it, to design a controller (e.g., via a linear model, develop a linear controller with techniques from classical control).

3. Using the mathematical model of the closed-loop system and mathematical or simulation-based analysis to study its performance (possibly leading to redesign).

4. Implementing the controller via, for example, a microprocessor, and evaluating the performance of the closed-loop system (again, possibly leading to redesign).

This procedure is concluded when the engineer has demonstrated that the control objectives have been met, and the controller (the "product") is approved for manufacturing and distribution.

In this book we show how the fuzzy control design methodology can be used to construct fuzzy controllers for challenging real-world applications. As opposed to "conventional" control approaches (e.g., proportional-integral-derivative (PID), lead-lag, and state feedback control) where the focus is on modeling and the use of this model to construct a controller that is described by differential equations, in fuzzy control we focus on gaining an intuitive understanding of how to best control the process, then we load this information directly into the fuzzy controller.

For instance, in the cruise control example we may gather rules about how to regulate the vehicle's speed from a human driver. One simple rule that a human driver may provide is "If speed is lower than the set-point, then press down further on the accelerator pedal." Other rules may depend on the rate of the speed error increase or decrease, or may provide ways to adapt the rules when there are significant plant parameter variations (e.g., if there is a significant increase in the mass of the vehicle, tune the rules to press harder on the accelerator pedal). For more challenging applications, control engineers typically have to gain a very good understanding of the plant to specify complex rules that dictate how the controller should react to the plant outputs and reference inputs.

Basically, while differential equations are the language of conventional control, heuristics and "rules" about how to control the plant are the language of fuzzy control. This is not to say that differential equations are not needed in the fuzzy control methodology. Indeed, one of the main focuses of this book will be on how "conventional" the fuzzy control methodology really is and how many ideas from conventional control can be quite useful in the analysis of this new class of control systems.

In this chapter we first provide an overview of the standard approach to constructing a control system and identify a wide variety of relevant conventional control ideas and techniques (see Section 1.2). We assume that the reader has at least some familiarity with conventional control. Our focus in this book is not only on introducing a variety of approaches to fuzzy control but also on comparing these to conventional control approaches to determine when fuzzy control offers advantages over conventional methods. Hence, to *fully* understand this book you need to understand several ideas from conventional control (e.g., classical control, state-space based design, the linear quadratic regulator, stability analysis, feedback linearization, adaptive control, etc.). The reader not familiar with conventional control to this extent will still find the book quite useful. In fact, we expect to whet the

appetite of such readers so that they become interested in learning more about conventional control. At the end of this chapter we will provide a list of books that can serve to teach such readers about these areas.

Following our overview of conventional control, in Section 1.3 we outline a "philosophy" of fuzzy control where we explain the design methodology for fuzzy controllers, relate this to the conventional control design methodology, and highlight the importance of analysis and verification of the behavior of closed-loop fuzzy control systems.

We highly recommend that you take the time to study this chapter (even if you already understand conventional control or even the basics of fuzzy control) as it will set the tone for the remainder of the book and provide a sound methodology for approaching the sometimes "overhyped" field of fuzzy control. Moreover, in Section 1.4 we provide a more detailed overview of this book than we provided in the Preface, and you will find this useful in deciding what topics to study closely and which ones you may want to skip over on a first reading.

1.2 Conventional Control System Design

A basic control system is shown in Figure 1.1. The process (or "plant") is the object to be controlled. Its inputs are $u(t)$, its outputs are $y(t)$, and the reference input is $r(t)$. In the cruise control problem, $u(t)$ is the throttle input, $y(t)$ is the speed of the vehicle, and $r(t)$ is the desired speed that is specified by the driver. The plant is the vehicle itself. The controller is the computer in the vehicle that actuates the throttle based on the speed of the vehicle and the desired speed that was specified. In this section we provide an overview of the steps taken to design the controller shown in Figure 1.1. Basically, these are modeling, controller design, and performance evaluation.

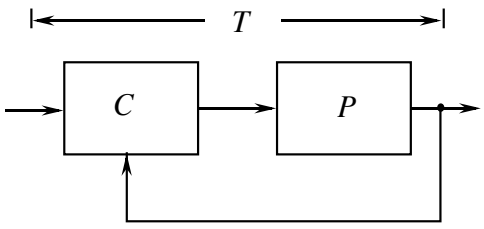


FIGURE 1.1 Control system.

1.2.1 Mathematical Modeling

When a control engineer is given a control problem, often one of the first tasks that she or he undertakes is the development of a mathematical model of the process to be controlled, in order to gain a clear understanding of the problem. Basically, there are only a few ways to actually generate the model. We can use first principles of

physics (e.g., $F = ma$) to write down a model. Another way is to perform "system identification" via the use of real plant data to produce a model of the system. Sometimes a combined approach is used where we use physics to write down a general differential equation that we believe represents the plant behavior, and then we perform experiments on the plant to determine certain model parameters or functions.

Often, more than one mathematical model is produced. A "truth model" is one that is developed to be as accurate as possible so that it can be used in simulation-based evaluations of control systems. It must be understood, however, that there is *never* a perfect mathematical model for the plant. The mathematical model is an *abstraction* and hence cannot perfectly represent all possible dynamics of any physical process (e.g., certain noise characteristics or failure conditions). This is not to say that we cannot produce models that are "accurate enough" to closely represent the behavior of a physical system. Usually, control engineers keep in mind that for control design they only need to use a model that is accurate enough to be able to design a controller that will work. Then, they often also need a very accurate model to test the controller in simulation (e.g., the truth model) before it is tested in an experimental setting. Hence, lower-order "design models" are also often developed that may satisfy certain assumptions (e.g., linearity or the inclusion of only certain forms of nonlinearities) yet still capture the essential plant behavior. Indeed, it is quite an art (and science) to produce good low-order models that satisfy these constraints. We emphasize that the reason we often need simpler models is that the synthesis techniques for controllers often require that the model of the plant satisfy certain assumptions (e.g., linearity) or these methods generally cannot be used.

Linear models such as the one in Equation (1.1) have been used extensively in the past and the control theory for linear systems is quite mature.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{1.1}$$

In this case u is the m-dimensional input; x is the n-dimensional state ($\dot{x} = \frac{dx(t)}{dt}$); y is the p dimensional output; and A, B, C, and D are matrices of appropriate dimension. Such models, or transfer functions ($G(s) = C(sI - A)^{-1}B + D$ where s is the Laplace variable), are appropriate for use with frequency domain design techniques (e.g., Bode plots and Nyquist plots), the root-locus method, state-space methods, and so on. Sometimes it is assumed that the parameters of the linear model are constant but unknown, or can be perturbed from their nominal values (then techniques for "robust control" or adaptive control are developed).

Much of the current focus in control is on the development of controllers using nonlinear models of the plant of the form

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= g(x, u) \end{aligned} \tag{1.2}$$

where the variables are defined as for the linear model and f and g are nonlinear functions of their arguments. One form of the nonlinear model that has received significant attention is

$$\dot{x} = f(x) + g(x)u \tag{1.3}$$

since it is possible to exploit the structure of this model to construct nonlinear controllers (e.g., in feedback linearization or nonlinear adaptive control). Of particular interest with both of the above nonlinear models is the case where f and g are not completely known and subsequent research focuses on robust control of nonlinear systems.

Discrete time versions of the above models are also used, and stochastic effects are often taken into account via the addition of a random input or other stochastic effects. Under certain assumptions you can linearize the nonlinear model in Equation (1.2) to obtain a linear one. In this case we sometimes think of the nonlinear model as the truth model, and the linear models that are generated from it as control design models. We will have occasion to work with all of the above models in this book.

There are certain properties of the plant that the control engineer often seeks to identify early in the design process. For instance, the stability of the plant may be analyzed (e.g., to see if certain variables remain bounded). The effects of certain nonlinearities are also studied. The engineer may want to determine if the plant is "controllable" to see, for example, if the control inputs will be able to properly affect the plant; and "observable" to see, for example, if the chosen sensors will allow the controller to observe the critical plant behavior so that it can be compensated for, or if it is "nonminimum phase." These properties will have a fundamental impact on our ability to design effective controllers for the system. In addition, the engineer will try to make a general assessment of how the plant behaves under various conditions, how the plant dynamics may change over time, and what random effects are present. Overall, this analysis of the plant's behavior gives the control engineer a fundamental understanding of the plant dynamics. This will be very valuable when it comes time to synthesize a controller.

1.2.2 Performance Objectives and Design Constraints

Controller design entails constructing a controller to meet the specifications. Often the first issue to address is whether to use open- or closed-loop control. If you can achieve your objectives with open-loop control, why turn to feedback control? Often, you need to pay for a sensor for the feedback information and there needs to be justification for this cost. Moreover, feedback can destabilize the system. Do not develop a feedback controller just because you are used to developing feedback controllers; you may want to consider an open-loop controller since it may provide adequate performance.

Assuming you use feedback control, the closed-loop specifications (or "performance objectives") can involve the following factors:

- *Disturbance rejection properties* (e.g., for the cruise control problem, that the control system will be able to dampen out the effects of winds or road grade variations). Basically, the need for disturbance rejection creates the need for feedback control over open-loop control; for many systems it is simply impossible to achieve the specifications without feedback (e.g., for the cruise control problem, if you had no measurement of vehicle velocity, how well could you regulate the velocity to the driver's set-point?).
- *Insensitivity to plant parameter variations* (e.g., for the cruise control problem, that the control system will be able to compensate for changes in the total mass of the vehicle that may result from varying the numbers of passengers or the amount of cargo).
- *Stability* (e.g., in the cruise control problem, to guarantee that on a level road the actual speed will converge to the desired set-point).
- *Rise-time* (e.g., in the cruise control problem, a measure of how long it takes for the actual speed to get close to the desired speed when there is a step change in the set-point speed).
- *Overshoot* (e.g., in the cruise control problem, when there is a step change in the set-point, how much the speed will increase above the set-point).
- *Settling time* (e.g., in the cruise control problem, how much time it takes for the speed to reach to within 1% of the set-point).
- *Steady-state error* (e.g., in the cruise control problem, if you have a level road, can the error between the set-point and actual speed actually go to zero; or if there is a long positive road grade, can the cruise controller eventually achieve the set-point).

While these factors are used to characterize the *technical conditions* that indicate whether or not a control system is performing properly, there are other issues that must be considered that are often of equal or greater importance. These include the following:

- *Cost:* How much money will it take to implement the controller, or how much time will it take to develop the controller?
- *Computational complexity:* How much processor power and memory will it take to implement the controller?
- *Manufacturability:* Does your controller have any extraordinary requirements with regard to manufacturing the hardware that is to implement it?
- *Reliability:* Will the controller always perform properly? What is its "mean time between failures?"

- *Maintainability:* Will it be easy to perform maintenance and routine adjustments to the controller?
- *Adaptability:* Can the same design be adapted to other similar applications so that the cost of later designs can be reduced? In other words, will it be easy to modify the cruise controller to fit on different vehicles so that the development can be done just once?
- *Understandability:* Will the right people be able to understand the approach to control? For example, will the people that implement it or test it be able to fully understand it?
- *Politics:* Is your boss biased against your approach? Can you sell your approach to your colleagues? Is your approach too novel and does it thereby depart too much from standard company practice?

Most often not only must a particular approach to control satisfy the basic technical conditions for meeting the performance objectives, but the above issues must also be taken into consideration — and these can often force the control engineer to make some very practical decisions that can significantly affect how, for example, the ultimate cruise controller is designed. It is important then that the engineer has these issues in mind early in the design process.

1.2.3 Controller Design

Conventional control has provided numerous methods for constructing controllers for dynamic systems. Some of these are listed below, and we provide a list of references at the end of this chapter for the reader who is interested in learning more about any one of these topics.

- *Proportional-integral-derivative (PID) control:* Over 90% of the controllers in operation today are PID controllers (or at least some form of PID controller like a P or PI controller). This approach is often viewed as simple, reliable, and easy to understand. Often, like fuzzy controllers, heuristics are used to tune PID controllers (e.g., the Zeigler-Nichols tuning rules).
- *Classical control:* Lead-lag compensation, Bode and Nyquist methods, root-locus design, and so on.
- *State-space methods:* State feedback, observers, and so on.
- *Optimal control:* Linear quadratic regulator, use of Pontryagin's minimum principle or dynamic programming, and so on.
- *Robust control:* H_2 or H_∞ methods, quantitative feedback theory, loop shaping, and so on.

- *Nonlinear methods:* Feedback linearization, Lyapunov redesign, sliding mode control, backstepping, and so on.
- *Adaptive control:* Model reference adaptive control, self-tuning regulators, nonlinear adaptive control, and so on.
- *Stochastic control:* Minimum variance control, linear quadratic gaussian (LQG) control, stochastic adaptive control, and so on.
- *Discrete event systems:* Petri nets, supervisory control, infinitesimal perturbation analysis, and so on.

Basically, these conventional approaches to control system design offer a variety of ways to utilize information from mathematical models on how to do good control. Sometimes they do not take into account certain heuristic information early in the design process, but use heuristics when the controller is implemented to tune it (tuning is invariably needed since the model used for the controller development is not perfectly accurate). Unfortunately, when using some approaches to conventional control, some engineers become somewhat removed from the control problem (e.g., when they do not fully understand the plant and just take the mathematical model as given), and sometimes this leads to the development of unrealistic control laws. Sometimes in conventional control, useful heuristics are ignored because they do not fit into the proper mathematical framework, and this can cause problems.

1.2.4 Performance Evaluation

The next step in the design process is to perform analysis and performance evaluation. Basically, we need performance evaluation to test that the control system that we design does in fact meet the closed-loop specifications (e.g., for "commissioning" the control system). This can be particularly important in safety-critical applications such as a nuclear power plant control or in aircraft control. However, in some consumer applications such as the control of a washing machine or an electric shaver, it may not be as important in the sense that failures will not imply the loss of life (just the possible embarrassment of the company and cost of warranty expenses), so some of the rigorous evaluation methods can sometimes be ignored. Basically, there are three general ways to verify that a control system is operating properly: (1) mathematical analysis based on the use of formal models, (2) simulation-based analysis that most often uses formal models, and (3) experimental investigations on the real system.

Mathematical Analysis

In mathematical analysis you may seek to prove that the system is stable (e.g., stable in the sense of Lyapunov, asymptotically stable, or bounded-input bounded-output (BIBO) stable), that it is controllable, or that other closed-loop specifications such as disturbance rejection, rise-time, overshoot, settling time, and steady-state errors have been met. Clearly, however, there are several limitations to mathe-

matical analysis. First, it always relies on the accuracy of the mathematical model, which is never a perfect representation of the plant, so the conclusions that are reached from the analysis are in a sense only as accurate as the model that they were developed from (the reader should never forget that mathematical analysis proves that properties hold *for the mathematical model,* not for the real physical system). And, second, there is a need for the development of analysis techniques for even more sophisticated nonlinear systems since existing theory is somewhat lacking for the analysis of complex nonlinear (e.g., fuzzy) control systems, particularly when there are significant nonlinearities, a large number of inputs and outputs, and stochastic effects. These limitations do not make mathematical analysis useless for all applications, however. Often it can be viewed as one more method to enhance our confidence that the closed-loop system will behave properly, and sometimes it helps to uncover fundamental problems with a control design.

Simulation-Based Analysis

In simulation-based analysis we seek to develop a simulation model of the physical system. This can entail using physics to develop a mathematical model and perhaps real data can be used to specify some of the parameters of the model (e.g., via system identification or direct parameter measurement). The simulation model can often be made quite accurate, and you can even include the effects of implementation considerations such as finite word length restrictions. As discussed above, often the simulation model ("truth model") will be more complex than the model that is used for control design because this "design model" needs to satisfy certain assumptions for the control design methodology to apply (e.g., linearity or linearity in the controls). Often, simulations are developed on digital computers, but there are occasions where an analog computer is still quite useful (particularly for real-time simulation of complex systems or in certain laboratory settings).

Regardless of the approach used to develop the simulation, there are always limitations on what can be achieved in simulation-based analysis. First, as with the mathematical analysis, the model that is developed will never be perfectly accurate. Also, some properties simply cannot be fully verified via simulation studies. For instance, it is impossible to verify the asymptotic stability of an ordinary differential equation via simulations since a simulation can only run for a finite amount of time and only a finite number of initial conditions can be tested for these finite-length trajectories. Basically, however, simulation-based studies can enhance our confidence that properties of the closed-loop system hold, and can offer valuable insights into how to redesign the control system before you spend time implementing the control system.

Experimental Investigations

To conduct an experimental investigation of the performance of a control system, you implement the control system for the plant and test it under various conditions. Clearly, implementation can require significant resources (e.g., time, hardware), and for some plants you would not even consider doing an implementation

until extensive mathematical and simulation-based investigations have been performed. However, the experimental evaluation does shed some light on some other issues involved in control system design such as cost of implementation, reliability, and perhaps maintainability. The limitations of experimental evaluations are, first, problems with the repeatability of experiments, and second, variations in physical components, which make the verification only approximate for other plants that are manufactured at other times. On the other hand, experimental studies can go a long way toward enhancing our confidence that the system will actually work since if you can get the control system to operate, you will see one real example of how it can perform.

Regardless of whether you choose to use one or all three of the above approaches to performance evaluation, it is important to keep in mind that there are two basic reasons we do such analysis. First, we seek to verify that the designed control system will perform properly. Second, if it does not perform properly, then we hope that the analysis will suggest a way to improve the performance so that the controller can be redesigned and the closed-loop specifications met.

1.3 Fuzzy Control System Design

What, then, is the motivation for turning to fuzzy control? Basically, the difficult task of modeling and simulating complex real-world systems for control systems development, especially when implementation issues are considered, is well documented. Even if a relatively accurate model of a dynamic system can be developed, it is often too complex to use in controller development, especially for many conventional control design procedures that require restrictive assumptions for the plant (e.g., linearity). It is for this reason that in practice conventional controllers are often developed via simple models of the plant behavior that satisfy the necessary assumptions, and via the ad hoc tuning of relatively simple linear or nonlinear controllers. Regardless, it is well understood (although sometimes forgotten) that heuristics enter the conventional control design process as long as you are concerned with the actual implementation of the control system. It must be acknowledged, moreover, that conventional control engineering approaches that use appropriate heuristics to tune the design have been relatively successful. You may ask the following questions: How much of the success can be attributed to the use of the mathematical model and conventional control design approach, and how much should be attributed to the clever heuristic tuning that the control engineer uses upon implementation? And if we exploit the use of heuristic information throughout the entire design process, can we obtain higher performance control systems?

Fuzzy control provides a formal methodology for representing, manipulating, and implementing a human's heuristic knowledge about how to control a system. In this section we seek to provide a philosophy of how to approach the design of fuzzy controllers. This will lead us to provide a motivation for, and overview of, the entire book.

The fuzzy controller block diagram is given in Figure 1.2, where we show a fuzzy controller embedded in a closed-loop control system. The plant outputs are

denoted by $y(t)$, its inputs are denoted by $u(t)$, and the reference input to the fuzzy controller is denoted by $r(t)$.

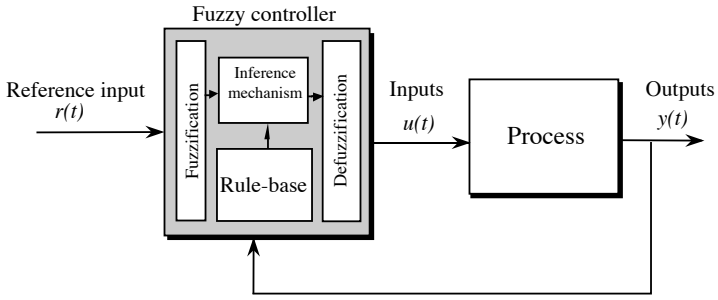


FIGURE 1.2 Fuzzy controller architecture.

The fuzzy controller has four main components: (1) The "rule-base" holds the knowledge, in the form of a set of rules, of how best to control the system. (2) The inference mechanism evaluates which control rules are relevant at the current time and then decides what the input to the plant should be. (3) The fuzzification interface simply modifies the inputs so that they can be interpreted and compared to the rules in the rule-base. And (4) the defuzzification interface converts the conclusions reached by the inference mechanism into the inputs to the plant.

Basically, you should view the fuzzy controller as an artificial decision maker that operates in a closed-loop system in real time. It gathers plant output data $y(t)$, compares it to the reference input $r(t)$, and then decides what the plant input $u(t)$ should be to ensure that the performance objectives will be met.

To design the fuzzy controller, the control engineer must gather information on how the artificial decision maker should act in the closed-loop system. Sometimes this information can come from a human decision maker who performs the control task, while at other times the control engineer can come to understand the plant dynamics and write down a set of rules about how to control the system without outside help. These "rules" basically say, "If the plant output and reference input are behaving in a certain manner, then the plant input should be some value." A whole set of such "If-Then" rules is loaded into the rule-base, and an inference strategy is chosen, then the system is ready to be tested to see if the closed-loop specifications are met.

This brief description provides a very high-level overview of how to design a fuzzy control system. Below we will expand on these basic ideas and provide more details on this procedure and its relationship to the conventional control design procedure.

1.3.1 Modeling Issues and Performance Objectives

People working in fuzzy control often say that "a model is not needed to develop a fuzzy controller, and this is the main advantage of the approach." However, will a proper understanding of the plant dynamics be obtained without trying to use first principles of physics to develop a mathematical model? And will a proper understanding of how to control the plant be obtained without simulation-based evaluations that also need a model? We always know roughly what process we are controlling (e.g., we know whether it is a vehicle or a nuclear reactor), and it is often possible to produce at least an approximate model, so why not do this? For a safety-critical application, if you do not use a formal model, then it is not possible to perform mathematical analysis or simulation-based evaluations. Is it wise to ignore these analytical approaches for such applications? Clearly, there will be some applications where you can simply "hack" together a controller (fuzzy or conventional) and go directly to implementation. In such a situation there is no need for a formal model of the process; however, is this type of control problem really so challenging that fuzzy control is even needed? Could a conventional approach (such as PID control) or a "table look-up" scheme work just as well or better, especially considering implementation complexity?

Overall, when you carefully consider the possibility of ignoring the information that is frequently available in a mathematical model, it is clear that it will often be unwise to do so. Basically, then, the role of modeling in fuzzy control design is quite similar to its role in conventional control system design. In fuzzy control there is a more significant emphasis on the use of heuristics, but in many control approaches (e.g., PID control for process control) there is a similar emphasis. Basically, in fuzzy control there is a focus on the use of rules to represent how to control the plant rather than ordinary differential equations (ODE). This approach can offer some advantages in that the representation of knowledge in rules seems more lucid and natural to some people. For others, though, the use of differential equations is more clear and natural. Basically, there is simply a "language difference" between fuzzy and conventional control: ODEs are the language of conventional control, and rules are the language of fuzzy control.

The performance objectives and design constraints are the same as the ones for conventional control that we summarized above, since we still want to meet the same types of closed-loop specifications. The fundamental limitations that the plant provides affect our ability to achieve high-performance control, and these are still present just as they were for conventional control (e.g., nonminimum phase or unstable behavior still presents challenges for fuzzy control).

1.3.2 Fuzzy Controller Design

Fuzzy control system design essentially amounts to (1) choosing the fuzzy controller inputs and outputs, (2) choosing the preprocessing that is needed for the controller inputs and possibly postprocessing that is needed for the outputs, and (3) designing each of the four components of the fuzzy controller shown in Figure 1.2. As you will see in the next chapter, there are standard choices for the fuzzification and

defuzzification interfaces. Moreover, most often the designer settles on an inference mechanism and may use this for many different processes. Hence, the main part of the fuzzy controller that we focus on for design is the rule-base.

The rule-base is constructed so that it represents a human expert "in-the-loop." Hence, the information that we load into the rules in the rule-base may come from an actual human expert who has spent a long time learning how best to control the process. In other situations there is no such human expert, and the control engineer will simply study the plant dynamics (perhaps using modeling and simulation) and write down a set of control rules that makes sense. As an example, in the cruise control problem discussed above it is clear that anyone who has experience driving a car can practice regulating the speed about a desired set-point and load this information into a rule-base. For instance, one rule that a human driver may use is "If the speed is lower than the set-point, then press down further on the accelerator pedal." A rule that would represent even more detailed information about how to regulate the speed would be "If the speed is lower than the set-point AND the speed is approaching the set-point very fast, then release the accelerator pedal by a small amount." This second rule characterizes our knowledge about how to make sure that we do not overshoot our desired goal (the set-point speed). Generally speaking, if we load very detailed expertise into the rule-base, we enhance our chances of obtaining better performance.

1.3.3 Performance Evaluation

Each and every idea presented in Section 1.2.4 on performance evaluation for conventional controllers applies here as well. The basic reason for this is that a fuzzy controller *is* a nonlinear controller — so many conventional modeling, analysis (via mathematics, simulation, or experimentation), and design ideas apply directly.

Since fuzzy control is a relatively new technology, it is often quite important to determine what value it has relative to conventional methods. Unfortunately, few have performed detailed comparative analyses between conventional and intelligent control that have taken into account a wide array of available conventional methods (linear, nonlinear, adaptive, etc.); fuzzy control methods (direct, adaptive, supervisory); theoretical, simulation, and experimental analyses; computational issues; and so on.

Moreover, most work in fuzzy control to date has focused only on its *advantages* and has not taken a critical look at what possible *disadvantages* there could be to using it (hence the reader should be cautioned about this when reading the literature). For example, the following questions are cause for concern when you employ a strategy of gathering heuristic control knowledge:

- Will the behaviors that are observed by a human expert and used to construct the fuzzy controller include all situations that can occur due to disturbances, noise, or plant parameter variations?
- Can the human expert realistically and reliably foresee problems that could arise from closed-loop system instabilities or limit cycles?

- Will the human expert be able to effectively incorporate stability criteria and performance objectives (e.g., rise-time, overshoot, and tracking specifications) into a rule-base to ensure that reliable operation can be obtained?

These questions may seem even more troublesome (1) if the control problem involves a safety-critical environment where the failure of the control system to meet performance objectives could lead to loss of human life or an environmental disaster, or (2) if the human expert's knowledge implemented in the fuzzy controller is somewhat inferior to that of the very experienced specialist we would expect to design the control system (different designers have different levels of expertise).

Clearly, then, for some applications there is a need for a methodology to develop, implement, and evaluate fuzzy controllers to ensure that they are reliable in meeting their performance specifications. This is the basic theme and focus of this book.

1.3.4 Application Areas

Fuzzy systems have been used in a wide variety of applications in engineering, science, business, medicine, psychology, and other fields. For instance, in engineering some potential application areas include the following:

- *Aircraft/spacecraft:* Flight control, engine control, avionic systems, failure diagnosis, navigation, and satellite attitude control.
- *Automated highway systems:* Automatic steering, braking, and throttle control for vehicles.
- *Automobiles:* Brakes, transmission, suspension, and engine control.
- *Autonomous vehicles:* Ground and underwater.
- *Manufacturing systems:* Scheduling and deposition process control.
- *Power industry:* Motor control, power control/distribution, and load estimation.
- *Process control:* Temperature, pressure, and level control, failure diagnosis, distillation column control, and desalination processes.
- *Robotics:* Position control and path planning.

This list is only representative of the range of possible applications for the methods of this book. Others have already been studied, while still others are yet to be identified.

1.4 What This Book Is About

In this section we will provide an overview of the techniques of this book by using an automotive cruise control problem as a motivational example. Moreover, we will state the basic objectives of the book.

1.4.1 What the Techniques Are Good For: An Example

In Chapter 2 we will introduce the basics of fuzzy control by explaining how the fuzzy controller processes its inputs to produce its outputs. In doing this, we explain all the details of rule-base construction, inference mechanism design, fuzzification, and defuzzification methods. This will show, for example, how for the cruise control application you can implement a set of rules about how to regulate vehicle speed. In Chapter 2 we also discuss the basics of fuzzy control system design and provide several design guidelines that have been found to be useful for practical applications such as cruise controller development. Moreover, we will show, by providing psuedocode, how to simulate a fuzzy control system, and will discuss issues that you encounter when seeking to implement a fuzzy control system. This will help you bridge the gap between theory and application so that you can quickly implement a fuzzy controller for your own application.

In Chapter 3 we perform several "case studies" in how to design fuzzy control systems. We pay particular attention to how these perform relative to conventional controllers and provide actual implementation results for several applications. It is via Chapter 3 that we solidify the reader's knowledge about how to design, simulate, and implement a fuzzy control system. In addition, we show examples of how fuzzy systems can be used as more general decision-making systems, not just in closed-loop feedback control.

In Chapter 4 we will show how conventional nonlinear analysis can be used to study, for example, the stability of a fuzzy control system. This sort of analysis is useful, for instance, to show that the cruise control system will always achieve the desired speed. For example, we will show how to verify that no matter what the actual vehicle speed is when the driver sets a desired speed, and no matter what terrain the vehicle is traveling over, the actual vehicle speed will stay close to the desired speed. We will also show that the actual speed will converge to the desired speed and not oscillate around it. While this analysis is important to help verify that the cruise controller is operating properly, it also helps to show the problems that can be encountered if you are not careful in the design of the fuzzy controller's rule-base.

Building on the basic fuzzy control approach that is covered in Chapters 2–4, in the remaining chapters of the book we show how fuzzy systems can be used for more advanced control and signal processing methods, sometimes via the implementation of more sophisticated intelligent reasoning strategies.

First, in Chapter 5 we show how to construct a fuzzy system from plant data so that it can serve as a model of the plant. Using the same techniques, we show how to construct fuzzy systems that are parameter estimators. In the cruise control problem such a "fuzzy estimator" could estimate the current combined mass of the vehicle and its occupants so that this parameter could be used by a control algorithm to achieve high-performance control even if there are significant mass changes (if the mass is increased, rules may be tuned to provide increased throttle levels). Other times, we can use these "fuzzy identification" techniques to construct (or design) a fuzzy controller from data we have gathered about how a human

expert (or some other system) performs a control problem. Chapter 5 also includes several case studies to show how to construct fuzzy systems from system data.

In Chapter 6 we further build on these ideas by showing how to construct "adaptive fuzzy controllers" that can automatically synthesize and, if necessary, tune a fuzzy controller using data from the plant. Such an adaptive fuzzy controller can be quite useful for plants where it is difficult to generate detailed a priori knowledge on how to control a plant, or for plants where there will be significant changes in its dynamics that result in inadequate performance if only a fixed fuzzy controller were used. For the cruise control example, an adaptive fuzzy controller may be particularly useful if there are failures in the engine that result in somewhat degraded engine performance. In this case, the adaptation mechanism would try to tune the rules of the fuzzy controller so that if, for example, the speed was lower than the set-point, the controller would open the throttle even more than it would with a nondegraded engine. If the engine failure is intermittent, however, and the engine stops performing poorly, then the adaptation mechanism would tune the rules so that the controller would react in the same way as normal. In Chapter 6 we introduce several approaches for adaptive fuzzy control and provide several case studies that help explain how to design, simulate, and implement adaptive fuzzy control systems.

In Chapter 7 we study another approach to specifying adaptive fuzzy controllers for the case where there is a priori heuristic knowledge available about how a fuzzy or conventional controller should be tuned. We will load such knowledge about how to supervise the fuzzy controller into what we will call a "fuzzy supervisory controller." For the cruise control example, suppose that we have an additional input to the system that allows the driver to specify how the vehicle is to respond to speed set-point changes. This input will allow the driver to specify if he or she wants the cruise controller to be very aggressive (i.e., act like a sports car) or very conservative (i.e., more like a family car). This information could be an input to a fuzzy supervisor that would tune the rules used for regulating the speed so that they would result in either fast or slow responses (or anything in between) to set-point changes. In Chapter 7 we will show several approaches to fuzzy supervisory control where we supervise either conventional or fuzzy controllers. Moreover, we provide several case studies to help show how to design, simulate, and implement fuzzy supervisory controllers.

In the final chapter of this book we highlight the issues involved in choosing fuzzy versus conventional controllers that were brought up throughout the book and provide a brief overview of other "intelligent control" methods that offer different perspectives on fuzzy control. These other methods include neural networks, genetic algorithms, expert systems, planning systems, and hierarchical intelligent autonomous controllers. We will introduce the multilayer perceptron and radial basis function neural network, explain their relationships to fuzzy systems, and explain how techniques from neural networks and fuzzy systems can cross-fertilize the two fields. We explain the basics of genetic algorithms, with a special focus on how these can be used in the design and tuning of fuzzy systems. We will explain how "expert controllers" can be viewed as a general type of fuzzy controller. We high-

light the additional functionalities often used in planning systems to reason about control, and discuss the possibility of using these in fuzzy control. Finally, we offer a broad view of the whole area of intelligent control by providing a functional architecture for an intelligent autonomous controller. We provide a brief description of the operation of the autonomous controller and explain how fuzzy control can fit into this architecture.

1.4.2 Objectives of This Book

Overall, the goals of this book are the following:

1. To introduce a variety of fuzzy control methods (fixed, adaptive, and supervisory) and show how they can utilize a wide diversity of heuristic knowledge about how to achieve good control.

2. To compare fuzzy control methods with conventional ones to try to determine the advantages and disadvantages of each.

3. To show how techniques and ideas from conventional control are quite useful in fuzzy control (e.g., methods for verifying that the closed-loop system performs according to the specifications and provides for stable operation).

4. To show how a fuzzy system is a tunable nonlinearity, various methods for tuning fuzzy systems, and how such approaches can be used in system identification, estimation, prediction, and adaptive and supervisory control.

5. To illustrate each of the fuzzy control approaches on a variety of challenging applications, to draw clear connections between the theory and application of fuzzy control (in this way we hope that you will be able to quickly apply the techniques described in this book to your own control problems).

6. To illustrate how to construct general fuzzy decision-making systems that can be used in a variety of applications.

7. To show clear connections between the field of fuzzy control and the other areas in intelligent control, including neural networks, genetic algorithms, expert systems, planning systems, and general hierarchical intelligent autonomous control.

The book includes many examples, applications, and case studies; and it is our hope that these will serve to show both how to develop fuzzy control systems and how they perform relative to conventional approaches. The problems at the ends of the chapters provide exercises and a variety of interesting (and sometimes challenging) design problems, and are sometimes used to introduce additional topics.

1.5 Summary

In this chapter we have provided an overview of the approaches to conventional and fuzzy control system design and have showed how they are quite similar in many respects. In this book our focus will be not only on introducing the basics of fuzzy control, but also on performance evaluation of the resulting closed-loop systems. Moreover, we will pay particular attention to the problem of assessing what advantages fuzzy control methods have over conventional methods. Generally, this must be done by careful comparative analyses involving modeling, mathematical analysis, simulation, implementation, and a full engineering cost-benefit analysis (which involves issues of cost, reliability, maintainability, flexibility, lead-time to production, etc.). Some of our comparisons will involve many of these dimensions while others will necessarily be more cursory.

Although it is not covered in this book, we would expect the reader to have as prerequisite knowledge a good understanding of the basic ideas in conventional control (at least, those typically covered in a first course on control). Upon completing this chapter, the reader should then understand the following:

- The distinction between a "truth model" and a "design model."
- The basic definitions of performance objectives (e.g., stability and overshoot).
- The general procedure used for the design of conventional and fuzzy control systems, which often involves modeling, analysis, and performance evaluation.
- The importance of using modeling information in the design of fuzzy controllers and when such information can be ignored.
- The idea that mathematical analysis provides proofs about the properties of the mathematical model and not the physical control system.
- The importance, roles, and limitations of mathematical analysis, simulation-based analysis, and experimental evaluations of performance for conventional and fuzzy control systems.
- The basic components of the fuzzy controller and fuzzy control system.
- The need to incorporate more sophisticated reasoning strategies in controllers and the subsequent motivation for adaptive and supervisory fuzzy control.

Essentially, this is a checklist for the major topics of this chapter. The reader should be sure to understand each of the above concepts before proceeding to later chapters, where the techniques of fuzzy control are introduced. We find that if you have a solid high-level view of the design process and philosophical issues involved, you will be more effective in developing control systems.

1.6 For Further Study

The more that you understand about conventional control, the more you will be able to appreciate some of the finer details of the operation of fuzzy control systems. We realize that all readers may not be familiar with all areas of control, so next we provide a list of books from which the major topics can be learned. There are many good texts on classical control [54, 102, 55, 45, 41, 10]. State-space methods and optimal and multivariable control can be studied in several of these texts and also in [56, 31, 3, 12, 132]. Robust control is treated in [46, 249]. Nonlinear control is covered in [90, 223, 13, 189, 217, 80]; stability analysis in [141, 140]; and adaptive control in [77, 99, 180, 11, 60, 149]. System identification is treated in [127] (and in the adaptive control texts), and optimal estimation and stochastic control are covered in [101, 123, 122, 63]. A relatively complete treatment of the field of control is in [121].

For more recent work in all these areas, see the proceedings of the IEEE Conference on Decision and Control, the American Control Conference, the European Control Conference, the International Federation on Automatic Control World Congress, and certain conferences in chemical, aeronautical, and mechanical engineering. Major journals to keep an eye on include the *IEEE Transactions on Automatic Control*, *IEEE Transactions on Control Systems Technology*, *IEEE Control Systems Magazine*, *Systems and Control Letters*, *Automatica*, *Control Engineering Practice*, *International Journal of Control*, and many others. Extensive lists of references for fuzzy and intelligent control are provided at the ends of Chapters 2–8.

1.7 Exercises

Exercise 1.1 (Modeling): This problem focuses on issues in modeling dynamic systems.

(a) What do we mean by model complexity and representation accuracy? List model features that affect the complexity of a model.

(b) What issues are of concern when determining how complex of a model to develop for a plant that is to be controlled?

(c) Are stochastic effects always present in physical systems? Explain.

(d) Why do we use discrete-time models?

(e) What are the advantages and disadvantages of representing a system with a linear model?

(f) Is a linear model of a physical system perfectly accurate? A nonlinear model? Explain.

Exercise 1.2 (Control System Properties): In this problem you will define the basic properties of systems that are used to quantify plant and closed-loop system dynamics and hence some performance specifications.

(a) Define, in words, bounded-input bounded-output (BIBO) stability, stability in the sense of Lyapunov, asymptotic stability, controllability, observability, rise-time, overshoot, and steady-state error (see [54, 31, 90] if you are unfamiliar with some of these concepts).

(b) Give examples of the properties in (a) for the following systems: cruise control for an automobile, aircraft altitude control, and temperature control in a house.

(c) Explain what disturbance rejection and sensitivity to plant parameter variations are, and identify disturbances and plant parameter variations for each of the systems in (b) (to do this you should describe the process, draw the control system for the process, show where the disturbance or plant parameter variation enters the system, and describe its effects on the closed-loop system). (See, for example, [45] if you are unfamiliar with these concepts.)

Exercise 1.3 (Fuzzy Control Design Philosophy): In this problem we will focus on the fuzzy control system design methodology.

(a) Is a model used in fuzzy control system design? If it is, when is it used, and what type of model is it? Should a model be used? Why? Why not?

(b) Explain the roles of knowledge acquisition, modeling, analysis, and past control designs in the construction of fuzzy control systems.

(c) What role does nonlinear analysis of stability play in fuzzy control system design?

Exercise 1.4 (Analysis): In this problem we will focus on performance analysis of control systems.

(a) Why are control engineers concerned with verifying that a control system will meet its performance specifications?

(b) How do they make sure that they are met? Is there any way to be 100% certain that the performance specifications can be met?

(c) What are the limitations of mathematical analysis, simulation-based analysis, and experimental analysis? What are the advantages of each of these?

Exercise 1.5 (Control Engineering Cost-Benefit Analysis): In this problem we will focus on engineering cost-benefit analysis for control systems.

(a) List all of the issues that must be considered in deciding what is the best approach to use for the control of a system (include in your list such issues as cost, marketing, etc.).

(b) Which of these issues is most important and why? In what situations? Rank the issues that must be considered in the order of priority for consideration, and justify your order.

Exercise 1.6 (Relations to Biological Intelligent Systems)*:[1] In this problem you will be asked to relate systems and control concepts to intelligent biological systems.

(a) The fuzzy controller represents, very crudely, the human deductive process. What features of the human deductive process seem to be ignored? Are these important for controller emulation? How could they be incorporated?

(b) Define the human brain as a dynamic system with inputs and outputs (what are they?). Define controllability, observability, and stability for both neurological (bioelectrical) activity and cognitive activities (i.e., the hardware and software of our brain).

(c) Do you think that it is possible to implement artificial intelligence in a current microcomputer and hence achieve intelligent control? On any computer or at any time in the future?

1. Reminder: Exercises or design problems that are particularly challenging (sometimes simply considering how far along you are in the text) or that require you to help define part of the problem are designated with a star ("$\star$").

CHAPTER 2

Fuzzy Control: The Basics

A few strong instincts and a few plain rules suffice us.

–Ralph Waldo Emerson

2.1 Overview

The primary goal of control engineering is to distill and apply knowledge about how to control a process so that the resulting control system will reliably and safely achieve high-performance operation. In this chapter we show how fuzzy logic provides a methodology for representing and implementing our knowledge about how best to control a process.

We begin in Section 2.2 with a "gentle" (tutorial) introduction, where we focus on the construction and basic mechanics of operation of a two-input one-output fuzzy controller with the most commonly used fuzzy operations. Building on our understanding of the two-input one-output fuzzy controller, in Section 2.3 we provide a mathematical characterization of general fuzzy systems with many inputs and outputs, and general fuzzification, inference, and defuzzification strategies. In Section 2.4 we illustrate some typical steps in the fuzzy control design process via a simple inverted pendulum control problem. We explain how to write a computer program that will simulate the actions of a fuzzy controller in Section 2.5. Moreover, we discuss various issues encountered in implementing fuzzy controllers in Section 2.6.

Then, in Chapter 3, after providing an overview of some design methodologies for fuzzy controllers and computer-aided design (CAD) packages for fuzzy system construction, we present several design case studies for fuzzy control systems. It is these case studies that the reader will find most useful in learning the finer

points about the fuzzy controller's operation and design. Indeed, the best way to really learn fuzzy control is to design your own fuzzy controller for one of the plants studied in this or the next chapter, and simulate the fuzzy control system to evaluate its performance. Initially, we recommend coding this fuzzy controller in a high-level language such as C, Matlab, or Fortran. Later, after you have acquired a firm understanding of the fuzzy controller's operation, you can take shortcuts by using a (or designing your own) CAD package for fuzzy control systems.

After completing this chapter, the reader should be able to design and simulate a fuzzy control system. This will move the reader a long way toward implementation of fuzzy controllers since we provide pointers on how to overcome certain practical problems encountered in fuzzy control system design and implementation (e.g., coding the fuzzy controller to operate in real-time, even with large rule-bases).

This chapter provides a foundation on which the remainder of the book rests. After our case studies in direct fuzzy controller design in Chapter 3, we will use the basic definition of the fuzzy control system and study its fundamental dynamic properties, including stability, in Chapter 4. We will use the same plants, and others, to illustrate the techniques for fuzzy identification, fuzzy adaptive control, and fuzzy supervisory control in Chapters 5, 6, and 7, respectively. It is therefore important for the reader to have a firm grasp of the concepts in this and the next chapter before moving on to these more advanced chapters.

Before skipping any sections or chapters of this book, we recommend that the reader study the chapter summaries at the end of each chapter. In these summaries we will highlight all the major concepts, approaches, and techniques that are covered in the chapter. These summaries also serve to remind the reader what should be learned in each chapter.

2.2 Fuzzy Control: A Tutorial Introduction

A block diagram of a fuzzy control system is shown in Figure 2.1. The fuzzy controller[1] is composed of the following four elements:

1. *A rule-base* (a set of If-Then rules), which contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control.

2. *An inference mechanism* (also called an "inference engine" or "fuzzy inference" module), which emulates the expert's decision making in interpreting and applying knowledge about how best to control the plant.

3. *A fuzzification interface*, which converts controller inputs into information that the inference mechanism can easily use to activate and apply rules.

4. *A defuzzification interface*, which converts the conclusions of the inference mechanism into actual inputs for the process.

1. Sometimes a fuzzy controller is called a "fuzzy logic controller" (FLC) or even a "fuzzy linguistic controller" since, as we will see, it uses fuzzy logic in the quantification of linguistic descriptions. In this book we will avoid these phrases and simply use "fuzzy controller."

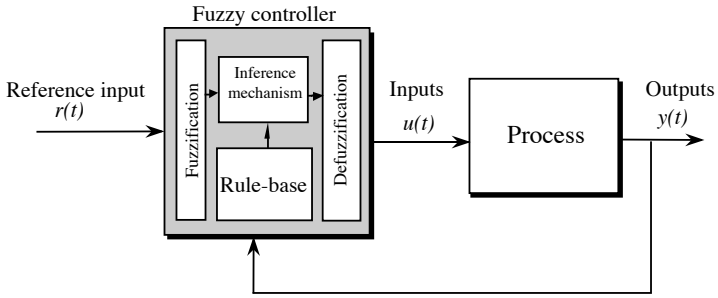


FIGURE 2.1 Fuzzy controller.

We introduce each of the components of the fuzzy controller for a simple problem of balancing an inverted pendulum on a cart, as shown in Figure 2.2. Here, y denotes the angle that the pendulum makes with the vertical (in radians), l is the half-pendulum length (in meters), and u is the force input that moves the cart (in Newtons). We will use r to denote the desired angular position of the pendulum. The goal is to balance the pendulum in the upright position (i.e., $r = 0$) when it initially starts with some nonzero angle off the vertical (i.e., $y \neq 0$). This is a very simple and academic nonlinear control problem, and many good techniques already exist for its solution. Indeed, for this standard configuration, a simple PID controller works well even in implementation.

In the remainder of this section, we will use the inverted pendulum as a convenient problem to illustrate the design and basic mechanics of the operation of a fuzzy control system. We will also use this problem in Section 2.4 to discuss much more general issues in fuzzy control system design that the reader will find useful for more challenging applications (e.g., the ones in the next chapter).

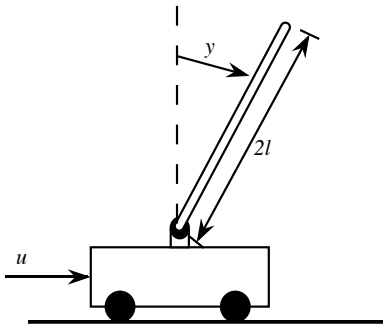


FIGURE 2.2 Inverted pendulum on a cart.

2.2.1 Choosing Fuzzy Controller Inputs and Outputs

Consider a human-in-the-loop whose responsibility is to control the pendulum, as shown in Figure 2.3. The fuzzy controller is to be designed to automate how a human expert who is successful at this task would control the system. First, the expert tells us (the designers of the fuzzy controller) what information she or he will use as inputs to the decision-making process. Suppose that for the inverted pendulum, the expert (this could be you!) says that she or he will use

$$e(t) = r(t) - y(t)$$

and

$$\frac{d}{dt}e(t)$$

as the variables on which to base decisions. Certainly, there are many other choices (e.g., the integral of the error e could also be used) but this choice makes good intuitive sense. Next, we must identify the controlled variable. For the inverted pendulum, we are allowed to control only the force that moves the cart, so the choice here is simple.

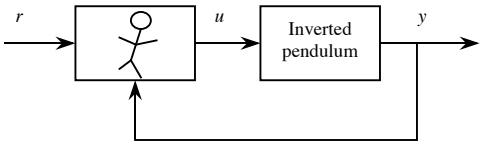


FIGURE 2.3 Human controlling an inverted pendulum on a cart.

For more complex applications, the choice of the inputs to the controller and outputs of the controller (inputs to the plant) can be more difficult. Essentially, you want to make sure that the controller will have the proper information available to be able to make good decisions and have proper control inputs to be able to steer the system in the directions needed to be able to achieve high-performance operation. Practically speaking, access to information and the ability to effectively control the system often cost money. If the designer believes that proper information is not available for making control decisions, he or she may have to invest in another sensor that can provide a measurement of another system variable. Alternatively, the designer may implement some filtering or other processing of the plant outputs. In addition, if the designer determines that the current actuators will not allow for the precise control of the process, he or she may need to invest in designing and implementing an actuator that can properly affect the process. Hence, while in some academic problems you may be given the plant inputs and outputs, in many practical situations you may have some flexibility in their choice. These choices

affect what information is available for making on-line decisions about the control of a process and hence affect how we design a fuzzy controller.

Once the fuzzy controller inputs and outputs are chosen, you must determine what the reference inputs are. For the inverted pendulum, the choice of the reference input $r = 0$ is clear. In some situations, however, you may want to choose r as some nonzero constant to balance the pendulum in the off-vertical position. To do this, the controller must maintain the cart at a constant acceleration so that the pendulum will not fall.

After all the inputs and outputs are defined for the fuzzy controller, we can specify the fuzzy control system. The fuzzy control system for the inverted pendulum, with our choice of inputs and outputs, is shown in Figure 2.4. Now, *within this framework* we seek to obtain a description of how to control the process. We see then that the choice of the inputs and outputs of the controller places certain constraints on the remainder of the fuzzy control design process. If the proper information is not provided to the fuzzy controller, there will be little hope for being able to design a good rule-base or inference mechanism. Moreover, even if the proper information is available to make control decisions, this will be of little use if the controller is not able to properly affect the process variables via the process inputs. It must be understood that the choice of the controller inputs and outputs is a fundamentally important part of the control design process. We will revisit this issue several times throughout the remainder of this chapter (and book).

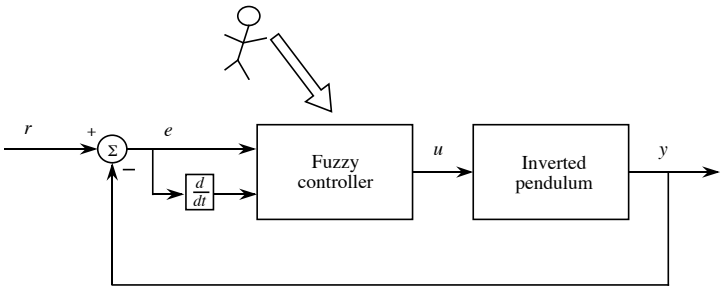


FIGURE 2.4 Fuzzy controller for an inverted pendulum on a cart.

2.2.2 Putting Control Knowledge into Rule-Bases

Suppose that the human expert shown in Figure 2.3 provides a description of how best to control the plant in some natural language (e.g., English). We seek to take this "linguistic" description and load it into the fuzzy controller, as indicated by the arrow in Figure 2.4.

Linguistic Descriptions

The linguistic description provided by the expert can generally be broken into several parts. There will be "linguistic variables" that describe each of the time-varying fuzzy controller inputs and outputs. For the inverted pendulum,

"error" describes $e(t)$
"change-in-error" describes $\frac{d}{dt}e(t)$
"force" describes $u(t)$

Note that we use quotes to emphasize that certain words or phrases are linguistic descriptions, and that we have added the time index to, for example, $e(t)$, to emphasize that generally e varies with time. There are many possible choices for the linguistic descriptions for variables. Some designers like to choose them so that they are quite descriptive for documentation purposes. However, this can sometimes lead to long descriptions. Others seek to keep the linguistic descriptions as short as possible (e.g., using "$e(t)$" as the linguistic variable for $e(t)$), yet accurate enough so that they adequately represent the variables. Regardless, the choice of the linguistic variable has no impact on the way that the fuzzy controller operates; it is simply a notation that helps to facilitate the construction of the fuzzy controller via fuzzy logic.

Just as $e(t)$ takes on a value of, for example, 0.1 at $t = 2$ ($e(2) = 0.1$), linguistic variables assume "linguistic values." That is, the values that linguistic variables take on over time change dynamically. Suppose for the pendulum example that "error," "change-in-error," and "force" take on the following values:

"neglarge"
"negsmall"
"zero"
"possmall"
"poslarge"

Note that we are using "negsmall" as an abbreviation for "negative small in size" and so on for the other variables. Such abbreviations help keep the linguistic descriptions short yet precise. For an even shorter description we could use integers:

"-2" to represent "neglarge"
"-1" to represent "negsmall"
"0" to represent "zero"
"1" to represent "possmall"
"2" to represent "poslarge"

This is a particularly appealing choice for the linguistic values since the descriptions are short and nicely represent that the variable we are concerned with has a numeric quality. We are not, for example, associating "-1" with any particular number of radians of error; the use of the numbers for linguistic descriptions simply quantifies the sign of the error (in the usual way) and indicates the size in relation to the

other linguistic values. We shall find the use of this type of linguistic value quite convenient and hence will give it the special name, "linguistic-numeric value."

The linguistic variables and values provide a language for the expert to express her or his ideas about the control decision-making process in the context of the framework established by our choice of fuzzy controller inputs and outputs. Recall that for the inverted pendulum $r = 0$ and $e = r - y$ so that

$$e = -y$$

and

$$\frac{d}{dt}e = -\frac{d}{dt}y$$

since $\frac{d}{dt}r = 0$. First, we will study how we can quantify certain dynamic behaviors with linguistics. In the next subsection we will study how to quantify knowledge about how to control the pendulum using linguistic descriptions.

For the inverted pendulum each of the following statements quantifies a different configuration of the pendulum (refer back to Figure 2.2 on page 25):

- The statement "error is poslarge" can represent the situation where the pendulum is at a significant angle to the *left* of the vertical.
- The statement "error is negsmall" can represent the situation where the pendulum is just slightly to the right of the vertical, but not too close to the vertical to justify quantifying it as "zero" and not too far away to justify quantifying it as "neglarge."
- The statement "error is zero" can represent the situation where the pendulum is very near the vertical position (a linguistic quantification is not precise, hence we are willing to accept any value of the error around $e(t) = 0$ as being quantified linguistically by "zero" since this can be considered a better quantification than "possmall" or "negsmall").
- The statement "error is poslarge **and** change-in-error is possmall" can represent the situation where the pendulum is to the left of the vertical and, since $\frac{d}{dt}y < 0$, the pendulum is moving *away* from the upright position (note that in this case the pendulum is moving counterclockwise).
- The statement "error is negsmall **and** change-in-error is possmall" can represent the situation where the pendulum is slightly to the right of the vertical and, since $\frac{d}{dt}y < 0$, the pendulum is moving *toward* the upright position (note that in this case the pendulum is also moving counterclockwise).

It is important for the reader to study each of the cases above to understand how the expert's linguistics quantify the dynamics of the pendulum (actually, each partially quantifies the pendulum's state).

Overall, we see that to quantify the dynamics of the process we need to have a good understanding of the physics of the underlying process we are trying to control. While for the pendulum problem, the task of coming to a good understanding of the dynamics is relatively easy, this is not the case for many physical processes. Quantifying the process dynamics with linguistics is not always easy, and certainly a better understanding of the process dynamics generally leads to a better linguistic quantification. Often, this will naturally lead to a better fuzzy controller *provided* that you can adequately measure the system dynamics so that the fuzzy controller can make the right decisions at the proper time.

Rules

Next, we will use the above linguistic quantification to specify a set of rules (a rule-base) that captures the expert's knowledge about how to control the plant. In particular, for the inverted pendulum in the three positions shown in Figure 2.5, we have the following rules (notice that we drop the quotes since the whole rule is linguistic):

1. **If** error is neglarge **and** change-in-error is neglarge **Then** force is poslarge

   This rule quantifies the situation in Figure 2.5(a) where the pendulum has a large positive angle and is moving clockwise; hence it is clear that we should apply a strong positive force (to the right) so that we can try to start the pendulum moving in the proper direction.

2. **If** error is zero **and** change-in-error is possmall **Then** force is negsmall

   This rule quantifies the situation in Figure 2.5(b) where the pendulum has nearly a zero angle with the vertical (a linguistic quantification of zero does not imply that $e(t) = 0$ exactly) and is moving counterclockwise; hence we should apply a small negative force (to the left) to counteract the movement so that it moves toward zero (a positive force could result in the pendulum overshooting the desired position).

3. **If** error is poslarge **and** change-in-error is negsmall **Then** force is negsmall

   This rule quantifies the situation in Figure 2.5(c) where the pendulum is far to the left of the vertical and is moving clockwise; hence we should apply a small negative force (to the left) to assist the movement, but not a big one since the pendulum is already moving in the proper direction.

Each of the three rules listed above is a "linguistic rule" since it is formed solely from linguistic variables and values. Since linguistic values are not precise representations of the underlying quantities that they describe, linguistic rules are not precise either. They are simply abstract ideas about how to achieve good control that could mean somewhat different things to different people. They are, however, at

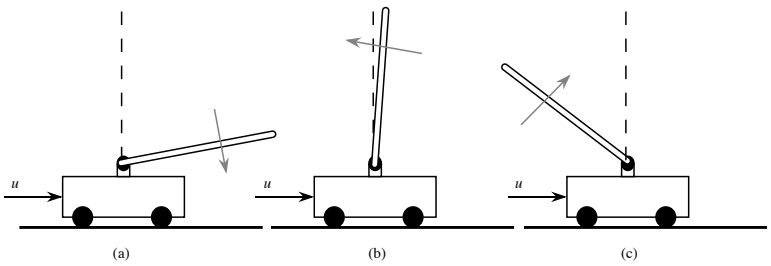


FIGURE 2.5 Inverted pendulum in various positions.

a level of abstraction that humans are often comfortable with in terms of specifying how to control a process.

The general form of the linguistic rules listed above is

If premise **Then** consequent

As you can see from the three rules listed above, the premises (which are sometimes called "antecedents") are associated with the fuzzy controller inputs and are on the left-hand-side of the rules. The consequents (sometimes called "actions") are associated with the fuzzy controller outputs and are on the right-hand-side of the rules. Notice that each premise (or consequent) can be composed of the conjunction of several "terms" (e.g., in rule 3 above "error is poslarge **and** change-in-error is negsmall" is a premise that is the conjunction of two terms). The number of fuzzy controller inputs and outputs places an upper limit on the number of elements in the premises and consequents. Note that there does not need to be a premise (consequent) term for each input (output) in each rule, although often there is.

Rule-Bases

Using the above approach, we could continue to write down rules for the pendulum problem for all possible cases (the reader should do this for practice, at least for a few more rules). Note that since we only specify a finite number of linguistic variables and linguistic values, there is only a finite number of possible rules. For the pendulum problem, with two inputs and five linguistic values for each of these, there are at most $5^2 = 25$ possible rules (all possible combinations of premise linguistic values for two inputs).

A convenient way to list all possible rules for the case where there are not too many inputs to the fuzzy controller (less than or equal to two or three) is to use a tabular representation. A tabular representation of one possible set of rules for the inverted pendulum is shown in Table 2.1. Notice that the body of the table lists the linguistic-numeric consequents of the rules, and the left column and top row of the table contain the linguistic-numeric premise terms. Then, for instance, the $(2, -1)$ position (where the "2" represents the row having "2" for a numeric-linguistic value and the "-1" represents the column having "-1" for a numeric-linguistic value) has a -1 ("negsmall") in the body of the table and represents the rule

If error is poslarge **and** change-in-error is negsmall **Then** force is negsmall

which is rule 3 above. Table 2.1 represents abstract knowledge that the expert has about how to control the pendulum given the error and its derivative as inputs.

TABLE 2.1 Rule Table for the Inverted Pendulum

"force" u		"change-in-error" $\dot{e}$				
		-2	-1	0	1	2
"error" e	-2	2	2	2	1	0
	-1	2	2	1	0	-1
	0	2	1	0	-1	-2
	1	1	0	-1	-2	-2
	2	0	-1	-2	-2	-2

The reader should convince him- or herself that the other rules are also valid and take special note of the pattern of rule consequents that appears in the body of the table: Notice the diagonal of zeros and viewing the body of the table as a matrix we see that it has a certain symmetry to it. This symmetry that emerges when the rules are tabulated is no accident and is actually a representation of abstract knowledge about how to control the pendulum; it arises due to a symmetry in the system's dynamics. We will actually see later that similar patterns will be found when constructing rule-bases for more challenging applications, and we will show how to exploit this symmetry in implementing fuzzy controllers.

2.2.3 Fuzzy Quantification of Knowledge

Up to this point we have only quantified, in an abstract way, the knowledge that the human expert has about how to control the plant. Next, we will show how to use fuzzy logic to fully quantify the meaning of linguistic descriptions so that we may automate, in the fuzzy controller, the control rules specified by the expert.

Membership Functions

First, we quantify the meaning of the linguistic values using "membership functions." Consider, for example, Figure 2.6. This is a plot of a function μ versus $e(t)$ that takes on special meaning. The function μ quantifies the *certainty*[2] that $e(t)$ can be classified linguistically as "possmall." To understand the way that a membership function works, it is best to perform a case analysis where we show how to interpret it for various values of $e(t)$:

2. The reader should not confuse the term "certainty" with "probability" or "likelihood." The membership function is not a probability density function, and there is no underlying probability space. By "certainty" we mean "degree of truth." The membership function does not quantify random behavior; it simply makes more accurate (less fuzzy) the meaning of linguistic descriptions.

- If $e(t) = -\pi/2$ then $\mu(-\pi/2) = 0$, indicating that we are certain that $e(t) = -\pi/2$ is *not* "possmall."
- If $e(t) = \pi/8$ then $\mu(\pi/8) = 0.5$, indicating that we are halfway certain that $e(t) = \pi/8$ is "possmall" (we are only halfway certain since it could also be "zero" with some degree of certainty—this value is in a "gray area" in terms of linguistic interpretation).
- If $e(t) = \pi/4$ then $\mu(\pi/4) = 1.0$, indicating that we are absolutely certain that $e(t) = \pi/4$ is what we mean by "possmall."
- If $e(t) = \pi$ then $\mu(\pi) = 0$, indicating that we are certain that $e(t) = \pi$ is not "possmall" (actually, it is "poslarge").

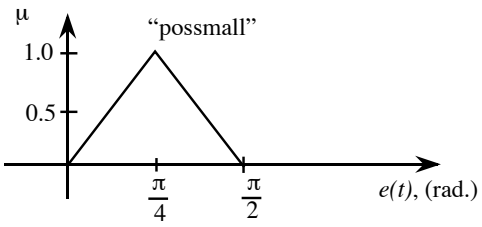


FIGURE 2.6 Membership function for linguistic value "possmall."

The membership function quantifies, in a continuous manner, whether values of $e(t)$ belong to (are members of) the set of values that are "possmall," and hence it quantifies the meaning of the linguistic statement "error is possmall." This is why it is called a membership function. It is important to recognize that the membership function in Figure 2.6 is only one possible definition of the meaning of "error is possmall"; you could use a bell-shaped function, a trapezoid, or many others.

For instance, consider the membership functions shown in Figure 2.7. For some application someone may be able to argue that we are absolutely certain that any value of $e(t)$ near $\frac{\pi}{4}$ is still "possmall" and only when you get sufficiently far from $\frac{\pi}{4}$ do we lose our confidence that it is "possmall." One way to characterize this understanding of the meaning of "possmall" is via the trapezoid-shaped membership function in Figure 2.7(a). For other applications you may think of membership in the set of "possmall" values as being dictated by the Gaussian-shaped membership function (not to be confused with the Gaussian probability density function) shown in Figure 2.7(b). For still other applications you may not readily accept values far away from $\frac{\pi}{4}$ as being "possmall," so you may use the membership function in Figure 2.7(c) to represent this. Finally, while we often think of symmetric characterizations of the meaning of linguistic values, we are not restricted to these

symmetric representations. For instance, in Figure 2.7(d) we represent that we believe that as $e(t)$ moves to the left of $\frac{\pi}{4}$ we are very quick to reduce our confidence that it is "possmall," but if we move to the right of $\frac{\pi}{4}$ our confidence that $e(t)$ is "possmall," diminishes at a slower rate.

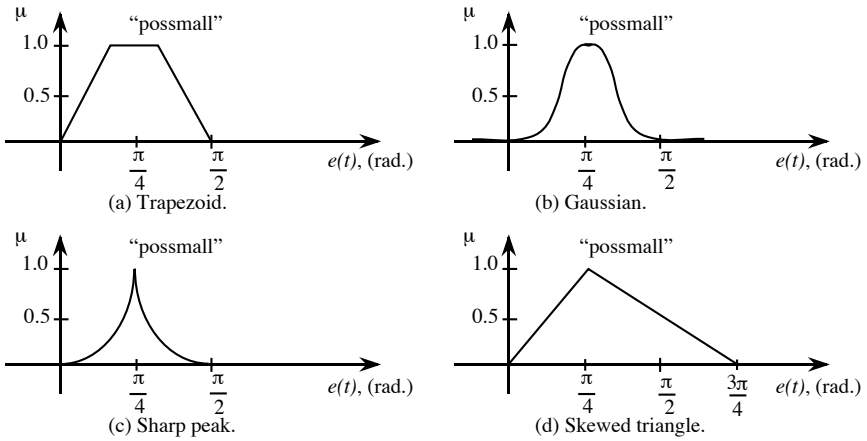


FIGURE 2.7 A few membership function choices for representing "error is possmall."

In summary, we see that depending on the application and the designer (expert), many different choices of membership functions are possible. We will further discuss other ways to define membership functions in Section 2.3.2 on page 55. It is important to note here, however, that for the most part the definition of a membership function is subjective rather than objective. That is, we simply quantify it in a manner that makes sense to us, but others may quantify it in a different manner.

The set of values that is described by μ as being "positive small" is called a "fuzzy set." Let A denote this fuzzy set. Notice that from Figure 2.6 we are absolutely certain that $e(t) = \frac{\pi}{4}$ is an element of A, but we are less certain that $e(t) = \frac{\pi}{16}$ is an element of A. Membership in the set, as specified by the membership function, is fuzzy; hence we use the term "fuzzy set." We will give a more precise description of a fuzzy set in Section 2.3.2 on page 55.

A "crisp" (as contrasted to "fuzzy") quantification of "possmall" can also be specified, but via the membership function shown in Figure 2.8. This membership function is simply an alternative representation for the interval on the real line $\pi/8 \leq e(t) \leq 3\pi/8$, and it indicates that this interval of numbers represents "possmall." Clearly, this characterization of crisp sets is simply another way to represent a normal interval (set) of real numbers.

While the vertical axis in Figure 2.6 represents certainty, the horizontal axis is also given a special name. It is called the "universe of discourse" for the input $e(t)$ since it provides the range of values of $e(t)$ that can be quantified with linguistics

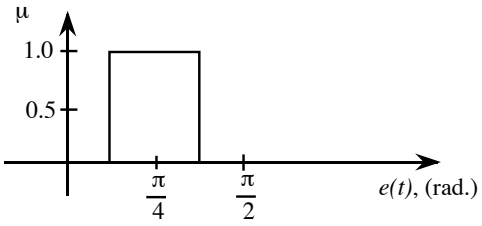


FIGURE 2.8 Membership function for a crisp set.

and fuzzy sets. In conventional terminology, a universe of discourse for an input or output of a fuzzy system is simply the range of values the inputs and outputs can take on.

Now that we know how to specify the meaning of a linguistic value via a membership function (and hence a fuzzy set), we can easily specify the membership functions for all 15 linguistic values (five for each input and five for the output) of our inverted pendulum example. See Figure 2.9 for one choice of membership functions.

Notice that (for our later convenience) we list both the linguistic values and the linguistic-numeric values associated with each membership function. Hence, we see that the membership function in Figure 2.6 for "possmall" is embedded among several others that describe other sizes of values (so that, for instance, the membership function to the right of the one for "possmall" is the one that represents "error is poslarge"). Note that other similarly shaped membership functions make sense (e.g., bell-shaped membership functions). We will discuss the multitude of choices that are possible for membership functions in Section 2.3.2 on page 55.

The membership functions at the outer edges in Figure 2.9 deserve special attention. For the inputs $e(t)$ and $\frac{d}{dt}e(t)$ we see that the outermost membership functions "saturate" at a value of one. This makes intuitive sense as at some point the human expert would just group all large values together in a linguistic description such as "poslarge." The membership functions at the outermost edges appropriately characterize this phenomenon since they characterize "greater than" (for the right side) and "less than" (for the left side). Study Figure 2.9 and convince yourself of this.

For the output u, the membership functions at the outermost edges cannot be saturated for the fuzzy system to be properly defined (more details on this point will be provided in Section 2.2.6 on page 44 and Section 2.3.5 on page 65). The basic reason for this is that in decision-making processes of the type we study, we seek to take actions that specify an exact value for the process input. We do not generally indicate to a process actuator, "any value bigger than, say, 10, is acceptable."

It is important to have a clear picture in your mind of how the values of the membership functions change as, for example, $e(t)$ changes its value over time. For instance, as $e(t)$ changes from $-\pi/2$ to $\pi/2$ we see that various membership

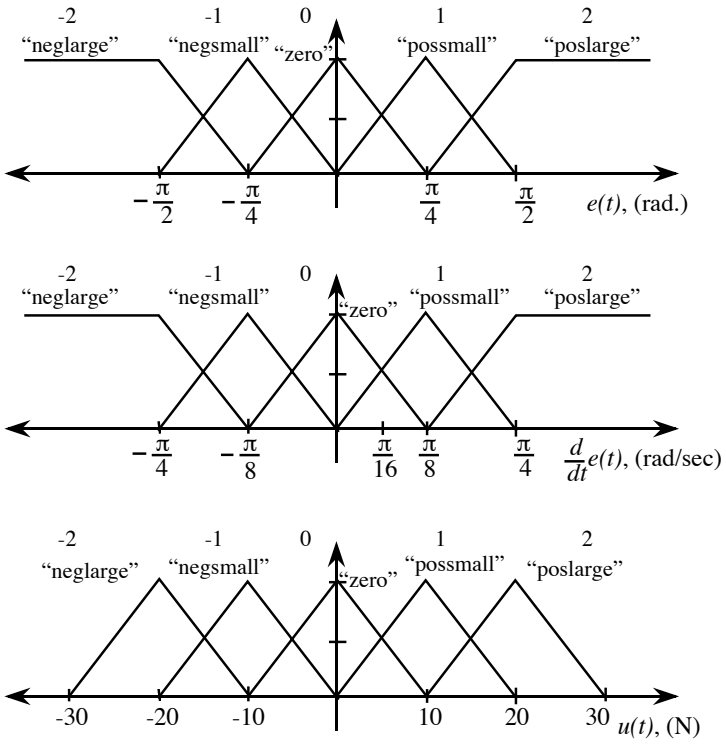


FIGURE 2.9 Membership functions for an inverted pendulum on a cart.

functions will take on zero and nonzero values indicating the degree to which the linguistic value appropriately describes the current value of $e(t)$. For example, at $e(t) = -\pi/2$ we are certain that the error is "neglarge," and as the value of $e(t)$ moves toward $-\pi/4$ we become less certain that it is "neglarge" and more certain that it is "negsmall." We see that the membership functions quantify the meaning of linguistic statements that describe time-varying signals.

Finally, note that often we will draw all the membership functions for one input or output variable on one graph; hence, we often omit the label for the vertical axis with the understanding that the plotted functions are membership functions describing the meaning of their associated linguistic values. Also, we will use the notation μ_{zero} to represent the membership function associated with the linguistic value "zero" and a similar notation for the others.

The rule-base of the fuzzy controller holds the linguistic variables, linguistic values, their associated membership functions, and the set of all linguistic rules (shown in Table 2.1 on page 32), so we have completed the description of the simple inverted pendulum. Next we describe the fuzzification process.

Fuzzification

It is actually the case that for most fuzzy controllers the fuzzification block in Figure 2.1 on page 25 can be ignored since this process is so simple. In Section 2.3.3 on page 61 we will explain the exact operations of the fuzzification process and also explain why it can be simplified and under certain conditions virtually ignored. For now, the reader should simply think of the fuzzification process as the act of obtaining a value of an input variable (e.g., $e(t)$) and finding the numeric values of the membership function(s) that are defined for that variable. For example, if $e(t) = \pi/4$ and $\frac{d}{dt}e(t) = \pi/16$, the fuzzification process amounts to finding the values of the input membership functions for these. In this case

$$\mu_{possmall}(e(t)) = 1$$

(with all others zero) and

$$\mu_{zero}\left(\frac{d}{dt}e(t)\right) = \mu_{possmall}\left(\frac{d}{dt}e(t)\right) = 0.5.$$

Some think of the membership function values as an "encoding" of the fuzzy controller numeric input values. The encoded information is then used in the fuzzy inference process that starts with "matching."

2.2.4 Matching: Determining Which Rules to Use

Next, we seek to explain how the inference mechanism in Figure 2.1 on page 25 operates. The inference process generally involves two steps:

1. The premises of all the rules are compared to the controller inputs to determine which rules apply to the current situation. This "matching" process involves determining the certainty that each rule applies, and typically we will more strongly take into account the recommendations of rules that we are more certain apply to the current situation.

2. The conclusions (what control actions to take) are determined using the rules that have been determined to apply at the current time. The conclusions are characterized with a fuzzy set (or sets) that represents the certainty that the input to the plant should take on various values.

We will cover step 1 in this subsection and step 2 in the next.

Premise Quantification via Fuzzy Logic

To perform inference we must first quantify each of the rules with fuzzy logic. To do this we first quantify the meaning of the premises of the rules that are composed of several terms, each of which involves a fuzzy controller input. Consider Figure 2.10, where we list two terms from the premise of the rule

If error is zero **and** change-in-error is possmall **Then** force is negsmall

Above, we had quantified the meaning of the linguistic terms "error is zero" and "change-in-error is possmall" via the membership functions shown in Figure 2.9. Now we seek to quantify the linguistic premise "error is zero **and** change-in-error is possmall." Hence, the main item to focus on is how to quantify the logical "and" operation that combines the meaning of two linguistic terms. While we could use standard Boolean logic to combine these linguistic terms, since we have quantified them more precisely with fuzzy sets (i.e., the membership functions), we can use these.

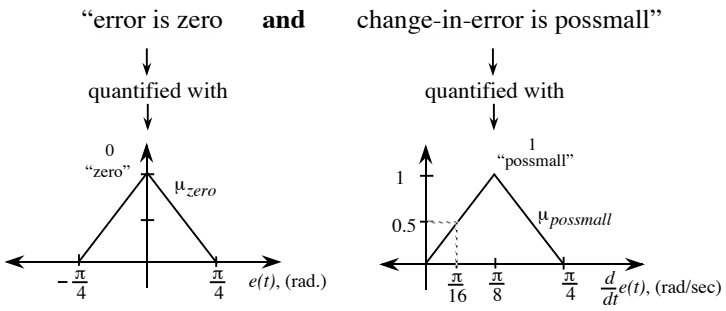


FIGURE 2.10 Membership functions of premise terms.

To see how to quantify the "and" operation, begin by supposing that $e(t) = \pi/8$ and $\frac{d}{dt}e(t) = \pi/32$, so that using Figure 2.9 (or Figure 2.10) we see that

$$\mu_{zero}(e(t)) = 0.5$$

and

$$\mu_{possmall}\left(\frac{d}{dt}e(t)\right) = 0.25$$

What, for these values of $e(t)$ and $\frac{d}{dt}e(t)$, is the certainty of the statement

"error is zero **and** change-in-error is possmall"

that is the premise from the above rule? We will denote this certainty by $\mu_{premise}$. There are actually several ways to define it:

- *Minimum:* Define $\mu_{premise} = \min\{0.5, 0.25\} = 0.25$, that is, using the minimum of the two membership values.
- *Product:* Define $\mu_{premise} = (0.5)(0.25) = 0.125$, that is, using the product of the two membership values.

Do these quantifications make sense? Notice that both ways of quantifying the "and" operation in the premise indicate that you can be no more certain about

the conjunction of two statements than you are about the individual terms that make them up (note that $0 \leq \mu_{premise} \leq 1$ for either case). If we are not very certain about the truth of one statement, how can we be any more certain about the truth of that statement "and" the other statement? It is important that you convince yourself that the above quantifications make sense. To do so, we recommend that you consider other examples of "anding" linguistic terms that have associated membership functions.

While we have simply shown how to quantify the "and" operation for one value of $e(t)$ and $\frac{d}{dt}e(t)$, if we consider all possible $e(t)$ and $\frac{d}{dt}e(t)$ values, we will obtain a multidimensional membership function $\mu_{premise}\left(e(t), \frac{d}{dt}e(t)\right)$ that is a function of $e(t)$ and $\frac{d}{dt}e(t)$ for each rule. For our example, if we choose the minimum operation to represent the "and" in the premise, then we get the multidimensional membership function $\mu_{premise}\left(e(t), \frac{d}{dt}e(t)\right)$ shown in Figure 2.11. Notice that if we pick values for $e(t)$ and $\frac{d}{dt}e(t)$, the value of the premise certainty $\mu_{premise}\left(e(t), \frac{d}{dt}e(t)\right)$ represents how certain we are that the rule

If error is zero **and** change-in-error is possmall **Then** force is negsmall

is applicable for specifying the force input to the plant. As $e(t)$ and $\frac{d}{dt}e(t)$ change, the value of $\mu_{premise}\left(e(t), \frac{d}{dt}e(t)\right)$ changes according to Figure 2.11, and we become less or more certain of the applicability of this rule.

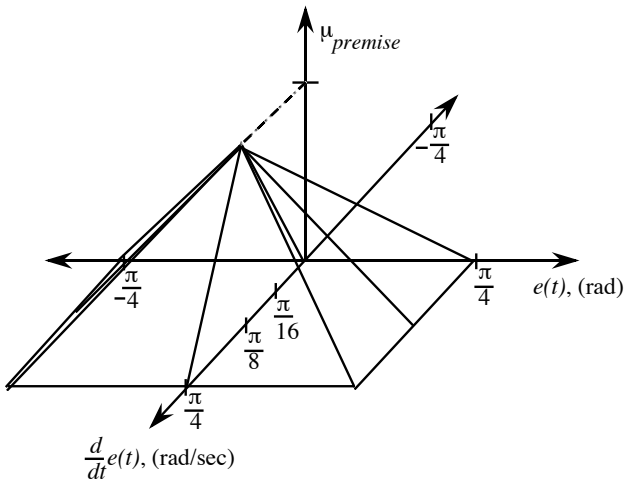


FIGURE 2.11 Membership function of the premise for a single rule.

In general we will have a different premise membership function for each of the rules in the rule-base, and each of these will be a function of $e(t)$ and $\frac{d}{dt}e(t)$ so that given specific values of $e(t)$ and $\frac{d}{dt}e(t)$ we obtain a quantification of the certainty

that each rule in the rule-base applies to the current situation. It is important you picture in your mind the situation where $e(t)$ and $\frac{d}{dt}e(t)$ change dynamically over time. When this occurs the values of $\mu_{premise}\left(e(t), \frac{d}{dt}e(t)\right)$ for each rule change, and hence the applicability of each rule in the rule-base for specifying the force input to the pendulum, changes with time.

Determining Which Rules Are On

Determining the applicability of each rule is called "matching." We say that a rule is "on at time t" if its premise membership function $\mu_{premise}(e(t), \frac{d}{dt}e(t)) > 0$. Hence, the inference mechanism seeks to determine which rules are on to find out which rules are relevant to the current situation. In the next step, the inference mechanism will seek to combine the recommendations of all the rules to come up with a single conclusion.

Consider, for the inverted pendulum example, how we compute the rules that are on. Suppose that

$$e(t) = 0$$

and

$$\frac{d}{dt}e(t) = \pi/8 - \pi/32 \ (= 0.294)$$

Figure 2.12 shows the membership functions for the inputs and indicates with thick black vertical lines the values above for $e(t)$ and $\frac{d}{dt}e(t)$. Notice that $\mu_{zero}(e(t)) = 1$ but that the other membership functions for the $e(t)$ input are all "off" (i.e., their values are zero). For the $\frac{d}{dt}e(t)$ input we see that $\mu_{zero}\left(\frac{d}{dt}e(t)\right) = 0.25$ and $\mu_{possmall}\left(\frac{d}{dt}e(t)\right) = 0.75$ and that all the other membership functions are off. This implies that rules that have the premise terms

"error is zero"
"change-in-error is zero"
"change-in-error is possmall"

are on (all other rules have $\mu_{premise}\left(e(t), \frac{d}{dt}e(t)\right) = 0$. So, which rules are these? Using Table 2.1 on page 32, we find that the rules that are on are the following:

1. **If** error is zero **and** change-in-error is zero **Then** force is zero
2. **If** error is zero **and** change-in-error is possmall **Then** force is negsmall

Note that since for the pendulum example we have at most two membership functions overlapping, we will never have more than four rules on at one time (this concept generalizes to many inputs and will be discussed in more detail in Sections 2.3 and 2.6). Actually, for this system we will either have one, two, or four rules on at any one time. To get only one rule on choose, for example, $e(t) = 0$ and $\frac{d}{dt}e(t) = \frac{\pi}{8}$ so that only rule 2 above is on. What values would you choose for

$e(t)$ and $\frac{d}{dt}e(t)$ to get four rules on? Why is it impossible, for this system, to have exactly three rules on?

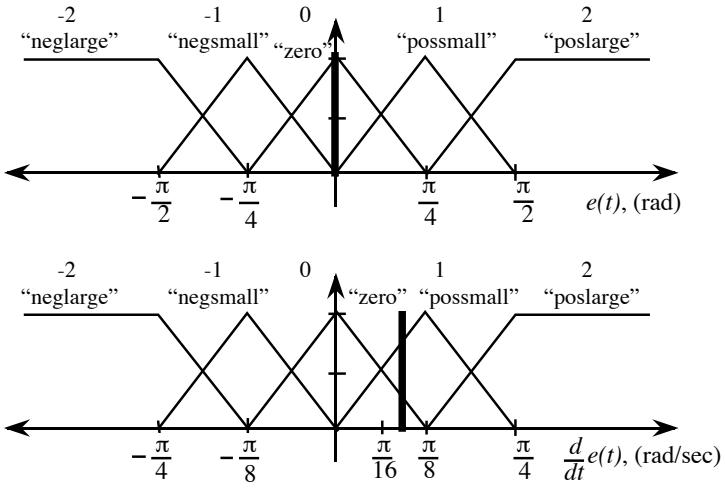


FIGURE 2.12 Input membership functions with input values.

It is useful to consider pictorially which rules are on. Consider Table 2.2, which is a copy of Table 2.1 on page 32 with boxes drawn around the consequents of the rules that are on (notice that these are the *same* two rules listed above). Notice that since $e(t) = 0$ ($e(t)$ is directly in the middle between the membership functions for "possmall" and "negsmall") both these membership functions are off. If we perturbed $e(t)$ slightly positive (negative), then we would have the two rules below (above) the two highlighted ones on also. With this, you should picture in your

TABLE 2.2 Rule Table for the Inverted Pendulum with Rules That Are "On" Highlighted.

"force" u		"change-in-error" $\dot{e}$				
		-2	-1	0	1	2
"error" e	-2	2	2	2	1	0
	-1	2	2	1	0	-1
	0	2	1	[0]	[-1]	-2
	1	1	0	-1	-2	-2
	2	0	-1	-2	-2	-2

mind how a region of rules that are on (that involves no more than four cells in the body of Table 2.2 due to how we define the input membership functions) will dynamically move around in the table as the values of $e(t)$ and $\frac{d}{dt}e(t)$ change. This completes our description of the "matching" phase of the inference mechanism.

2.2.5 Inference Step: Determining Conclusions

Next, we consider how to determine which conclusions should be reached when the rules that are on are applied to deciding what the force input to the cart carrying the inverted pendulum should be. To do this, we will first consider the recommendations of each rule independently. Then later we will combine all the recommendations from all the rules to determine the force input to the cart.

Recommendation from One Rule

Consider the conclusion reached by the rule

If error is zero **and** change-in-error is zero **Then** force is zero

which for convenience we will refer to as "rule (1)." Using the minimum to represent the premise, we have

$$\mu_{premise_{(1)}} = \min\{0.25, 1\} = 0.25$$

(the notation $\mu_{premise_{(1)}}$ represents $\mu_{premise}$ for rule (1)) so that we are 0.25 certain that this rule applies to the current situation. The rule indicates that if its premise is true then the action indicated by its consequent should be taken. For rule (1) the consequent is "force is zero" (this makes sense, for here the pendulum is balanced, so we should not apply any force since this would tend to move the pendulum away from the vertical). The membership function for this consequent is shown in Figure 2.13(a). The membership function for the conclusion reached by rule (1), which we denote by $\mu_{(1)}$, is shown in Figure 2.13(b) and is given by

$$\mu_{(1)}(u) = \min\{0.25, \mu_{zero}(u)\}$$

This membership function defines the "implied fuzzy set"[3] for rule (1) (i.e., it is the conclusion that is implied by rule (1)). The justification for the use of the minimum operator to represent the implication is that *we can be no more certain about our consequent than our premise*. You should convince yourself that we could use the product operation to represent the implication also (in Section 2.2.6 we will do an example where we use the product).

Notice that the membership function $\mu_{(1)}(u)$ is a *function* of u and that the minimum operation will generally "chop off the top" of the $\mu_{zero}(u)$ membership function to produce $\mu_{(1)}(u)$. For different values of $e(t)$ and $\frac{d}{dt}e(t)$ there will be different values of the premise certainty $\mu_{premise_{(1)}}\left(e(t), \frac{d}{dt}e(t)\right)$ for rule (1) and hence different *functions* $\mu_{(1)}(u)$ obtained (i.e., it will chop off the top at different points).

3. This term has been used in the literature for a long time; however, there is no standard terminology for this fuzzy set. Others have called it, for example, a "consequent fuzzy set" or an "output fuzzy set" (which can be confused with the fuzzy sets that quantify the consequents of the rules). We use "implied fuzzy set" so that there is no ambiguity and to help to distinguish it from the "overall implied fuzzy set" that is introduced in Section 2.3.

We see that $\mu_{(1)}(u)$ is in general a time-varying function that quantifies how certain rule (1) is that the force input u should take on certain values. It is most certain that the force input should lie in a region around zero (see Figure 2.13(b)), and it indicates that it is certain that the force input should not be too large in either the positive or negative direction—this makes sense if you consider the linguistic meaning of the rule. The membership function $\mu_{(1)}(u)$ quantifies the conclusion reached by only rule (1) and only for the current $e(t)$ and $\frac{d}{dt}e(t)$. It is important that the reader be able to picture how the shape of the implied fuzzy set changes as the rule's premise certainty changes over time.

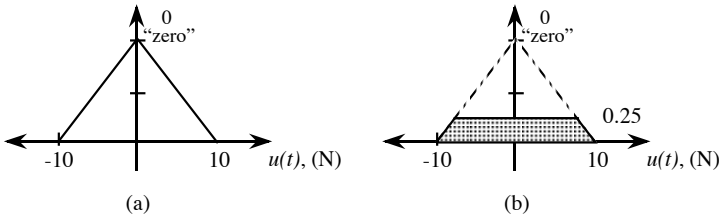


FIGURE 2.13 (a) Consequent membership function and (b) implied fuzzy set with membership function $\mu_{(1)}(u)$ for rule (1). Recall that the units for $u(t)$ are Newtons (N).

Recommendation from Another Rule

Next, consider the conclusion reached by the other rule that is on,

If error is zero **and** change-in-error is possmall **Then** force is negsmall

which for convenience we will refer to as "rule (2)." Using the minimum to represent the premise, we have

$$\mu_{premise_{(2)}} = \min\{0.75, 1\} = 0.75$$

so that we are 0.75 certain that this rule applies to the current situation. Notice that we are much more certain that rule (2) applies to the current situation than rule (1). For rule (2) the consequent is "force is negsmall" (this makes sense, for here the pendulum is perfectly balanced but is moving in the counterclockwise direction with a small velocity). The membership function for this consequent is shown in Figure 2.14(a). The membership function for the conclusion reached by rule (2), which we denote by $\mu_{(2)}$, is shown in Figure 2.14(b) (the shaded region) and is given by

$$\mu_{(2)}(u) = \min\{0.75, \mu_{negsmall}(u)\}$$

This membership function defines the implied fuzzy set for rule (2) (i.e., it is the conclusion that is reached by rule (2)). Once again, for different values of $e(t)$

and $\frac{d}{dt}e(t)$ there will be different values of $\mu_{premise_{(2)}}\left(e(t), \frac{d}{dt}e(t)\right)$ for rule (2) and hence different *functions* $\mu_{(2)}(u)$ obtained. The reader should carefully consider the meaning of the implied fuzzy set $\mu_{(2)}(u)$. Rule (2) is quite certain that the control output (process input) should be a small negative value. This makes sense since if the pendulum has some counterclockwise velocity then we would want to apply a negative force (i.e., one to the left). As rule (2) has a premise membership function that has higher certainty than for rule (1), we see that we are more certain of the conclusion reached by rule (2).

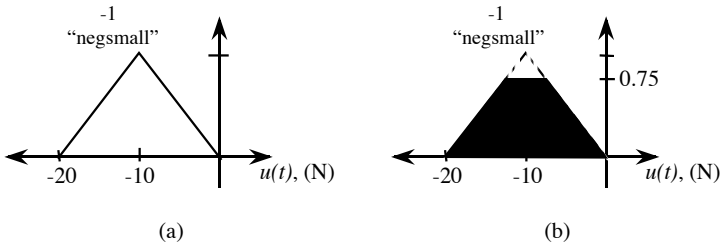


FIGURE 2.14 (a) Consequent membership function and (b) implied fuzzy set with membership function $\mu_{(2)}(u)$ for rule (2).

This completes the operations of the inference mechanism in Figure 2.1 on page 25. While the input to the inference process is the set of rules that are on, its output is the set of implied fuzzy sets that represent the conclusions reached by all the rules that are on. For our example, there are at most four conclusions reached since there are at most four rules on at any one time. (In fact, you could say that there are *always* four conclusions reached for our example, but that the implied fuzzy sets for some of the rules may have implied membership functions that are zero for all values.)

2.2.6 Converting Decisions into Actions

Next, we consider the defuzzification operation, which is the final component of the fuzzy controller shown in Figure 2.1 on page 25. Defuzzification operates on the implied fuzzy sets produced by the inference mechanism and combines their effects to provide the "most certain" controller output (plant input). Some think of defuzzification as "decoding" the fuzzy set information produced by the inference process (i.e., the implied fuzzy sets) into numeric fuzzy controller outputs.

To understand defuzzification, it is best to first draw all the implied fuzzy sets on one axis as shown in Figure 2.15. We want to find the one output, which we denote by "u^{crisp}," that best represents the conclusions of the fuzzy controller that are represented with the implied fuzzy sets. There are actually many approaches to defuzzification. We will consider two here and several others in Section 2.3.5 on page 65.

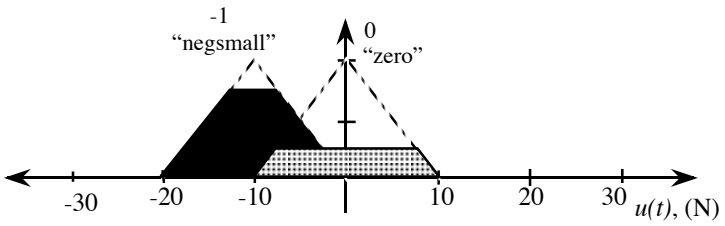


FIGURE 2.15 Implied fuzzy sets.

Combining Recommendations

Due to its popularity, we will first consider the "center of gravity" (COG) defuzzification method for combining the recommendations represented by the implied fuzzy sets from all the rules. Let b_i denote the center of the membership function (i.e., where it reaches its peak for our example) of the consequent of rule (i). For our example we have

$$b_1 = 0.0$$

and

$$b_2 = -10$$

as shown in Figure 2.15. Let

$$\int \mu_{(i)}$$

denote the area under the membership function $\mu_{(i)}$. The COG method computes u^{crisp} to be

$$u^{crisp} = \frac{\sum_i b_i \int \mu_{(i)}}{\sum_i \int \mu_{(i)}} \tag{2.1}$$

This is the classical formula for computing the center of gravity. In this case it is for computing the center of gravity of the implied fuzzy sets. Three items about Equation (2.1) are important to note:

1. Practically, we cannot have output membership functions that have infinite area since even though they may be "chopped off" in the minimum operation for the implication (or scaled for the product operation) they can still end up with infinite area. This is the reason we do not allow infinite area membership functions for the linguistic values for the controller output (e.g., we did not allow the saturated membership functions at the outermost edges as we had for the inputs shown in Figure 2.9 on page 36).

2. You must be careful to define the input and output membership functions so that the sum in the denominator of Equation (2.1) is not equal to zero no matter what the inputs to the fuzzy controller are. Essentially, this means that we must have some sort of conclusion for all possible control situations we may encounter.

3. While at first glance it may not appear so, $\int \mu_{(i)}$ is easy to compute for our example. For the case where we have symmetric triangular output membership functions that peak at one and have a base width of w, simple geometry can be used to show that the area under a triangle "chopped off" at a height of h (such as the ones in Figures 2.13 and 2.14) is equal to

$$w\left(h - \frac{h^2}{2}\right)$$

Given this, the computations needed to compute u^{crisp} are not too significant.

We see that the property of membership functions being symmetric for the output is important since in this case no matter whether the minimum or product is used to represent the implication, it will be the case that the center of the implied fuzzy set will be the same as the center of the consequent fuzzy set from which it is computed. If the output membership functions are not symmetric, then their centers, which are needed in the computation of the COG, will change depending on the membership value of the premise. This will result in the need to recompute the center at each time instant.

Using Equation (2.1) with Figure 2.15 we have

$$u^{crisp} = \frac{(0)(4.375) + (-10)(9.375)}{4.375 + 9.375} = -6.81$$

as the input to the pendulum for the given $e(t)$ and $\frac{d}{dt}e(t)$.

Does this value for a force input (i.e., 6.81 Newtons to the left) make sense? Consider Figure 2.16, where we have taken the implied fuzzy sets from Figure 2.15 and simply added an indication of what number COG defuzzification says is the best representation of the conclusions reached by the rules that are on. Notice that the value of u^{crisp} is roughly in the middle of where the implied fuzzy sets say they are most certain about the value for the force input. In fact, recall that we had

$$e(t) = 0$$

and

$$\frac{d}{dt}e(t) = \pi/8 - \pi/32 \; (= 0.294)$$

so the pendulum is in the inverted position but is moving counterclockwise with a small velocity; hence it makes sense to pull on the cart, and the fuzzy controller

does this.

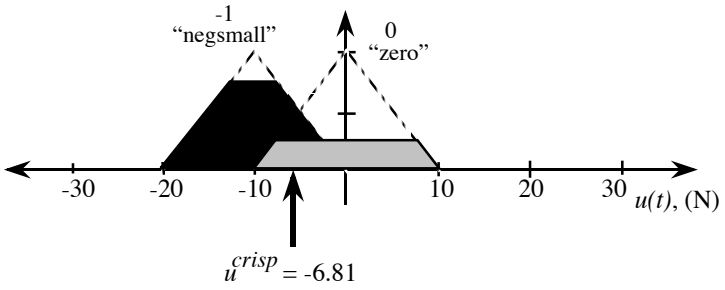


FIGURE 2.16 Implied fuzzy sets.

It is interesting to note that for our example it will be the case that

$$-20 \leq u^{crisp} \leq 20$$

To see this, consider Figure 2.17, where we have drawn the output membership functions. Notice that even though we have extended the membership functions at the outermost edges past -20 and $+20$ (see the shaded regions), the COG method will never compute a value outside this range.

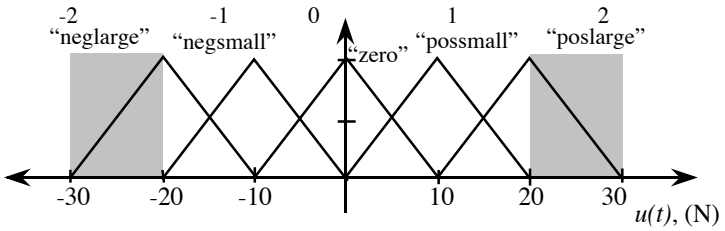


FIGURE 2.17 Output membership functions.

The reason for this comes directly from the definition of the COG method in Equation (2.1). The center of gravity for these shapes simply cannot extend beyond -20 and $+20$. Practically speaking, this ability to limit the range of inputs to the plant is useful; it may be the case that applying a force of greater than 20 Newtons is impossible for this plant. Thus we see that in defining the membership functions for the fuzzy controller, we must take into account what method is going to be used for defuzzification.

Other Ways to Compute and Combine Recommendations

As another example, it is interesting to consider how to compute, by hand, the operations that the fuzzy controller takes when we use the product to represent the implication or the "center-average" defuzzification method.

First, consider the use of the product. Consider Figure 2.18, where we have drawn the output membership functions for "negsmall" and "zero" as dotted lines. The implied fuzzy set from rule (1) is given by the membership function

$$\mu_{(1)}(u) = 0.25\mu_{zero}(u)$$

shown in Figure 2.18 as the shaded triangle; and the implied fuzzy set for rule (2) is given by the membership function

$$\mu_{(2)}(u) = 0.75\mu_{negsmall}(u)$$

shown in Figure 2.18 as the dark triangle. Notice that computation of the COG is easy since we can use $\frac{1}{2}wh$ as the area for a triangle with base width w and height h. When we use product to represent the implication, we obtain

$$u^{crisp} = \frac{(0)(2.5) + (-10)(7.5)}{2.5 + 7.5} = -7.5$$

which also makes sense.

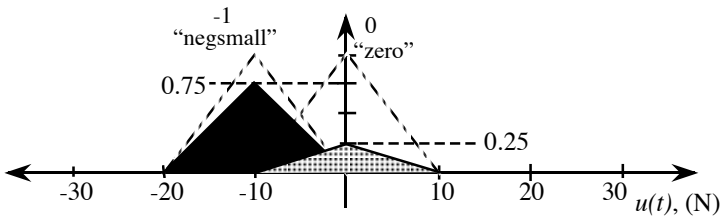


FIGURE 2.18 Implied fuzzy sets when the product is used to represent the implication.

Next, as another example of how to combine recommendations, we will introduce the "center-average" method for defuzzification. For this method we let

$$u^{crisp} = \frac{\sum_i b_i \mu_{premise_{(i)}}}{\sum_i \mu_{premise_{(i)}}} \tag{2.2}$$

where to compute $\mu_{premise_{(i)}}$ we use, for example, minimum. We call it the "center-average" method since Equation (2.2) is a weighted average of the center values of the output membership function centers. Basically, the center-average method replaces the areas of the implied fuzzy sets that are used in COG with the values of $\mu_{premise_{(i)}}$. This is a valid replacement since the area of the implied fuzzy set

is generally proportional to $\mu_{premise_{(i)}}$ since $\mu_{premise_{(i)}}$ is used to chop the top off (minimum) or scale (product) the triangular output membership function when COG is used for our example. For the above example, we have

$$u^{crisp} = \frac{(0)(0.25) + (-10)(0.75)}{0.25 + 0.75} = -7.5$$

which just happens to be the same value as above. Some like the center-average defuzzification method because the computations needed are simpler than for COG and because the output membership functions are easy to store since the only relevant information they provide is their center values (b_i) (i.e., their shape does not matter, just their center value).

Notice that while both values computed for the different inference and defuzzification methods provide reasonable command inputs to the plant, it is difficult to say which is best without further investigations (e.g., simulations or implementation). This ambiguity about how to define the fuzzy controller actually extends to the general case and also arises in the specification of all the other fuzzy controller components, as we discuss below. Some would call this "ambiguity" a design flexibility, but unfortunately there are not too many guidelines on how best to choose the inference strategy and defuzzification method, so such flexibility is of questionable value.

2.2.7 Graphical Depiction of Fuzzy Decision Making

For convenience, we summarize the procedure that the fuzzy controller uses to compute its outputs given its inputs in Figure 2.19. Here, we use the minimum operator to represent the "and" in the premise and the implication and COG defuzzification. The reader is advised to study each step in this diagram to gain a fuller understanding of the operation of the fuzzy controller. To do this, develop a similar diagram for the case where the product operator is used to represent the "and" in the premise and the implication, and choose values of $e(t)$ and $\frac{d}{dt}e(t)$ that will result in four rules being on. Then, repeat the process when center-average defuzzification is used with either minimum or product used for the premise. Also, learn how to picture in your mind how the parameters of this graphical representation of the fuzzy controller operations change as the fuzzy controller inputs change.

This completes the description of the operation of a simple fuzzy controller. You will find that while we will treat the fully general fuzzy controller in the next section, there will be little that is conceptually different from this simple example. We simply show how to handle the case where there are more inputs and outputs and show a fuller range of choices that you can make for the various components of the fuzzy controller.

As evidenced by the different values obtained by using the minimum, product, and defuzzification operations, there are many ways to choose the parameters of the fuzzy controller that make sense. This presents a problem since it is almost always difficult to know how to first design a fuzzy controller. Basically, the choice of all the components for the fuzzy controller is somewhat ad hoc. What are the

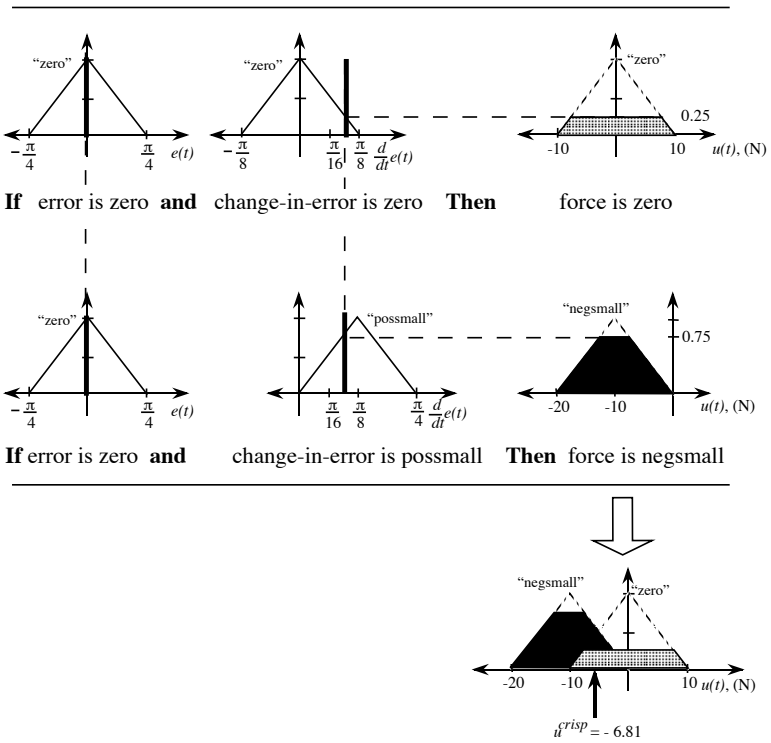


FIGURE 2.19 Graphical representation of fuzzy controller operations.

best membership functions? How many linguistic values and rules should there be? Should the minimum or product be used to represent the "and" in the premise—and which should be used to represent the implication? What defuzzification method should be chosen? These are all questions that must be addressed if you want to design a fuzzy controller.

We will show how to answer some of these questions by going through a design procedure for the inverted pendulum in Section 2.4 on page 77. After this, we will discuss how to write a computer program to simulate a fuzzy control system and how to do a real-time implementation of the fuzzy controller. Ultimately, however, the answers to the above questions are best found by studying how to design fuzzy controllers for a wide range of applications that present more challenging characteristics than the inverted pendulum. This is what we do in the case studies in Chapter 3.

2.2.8 Visualizing the Fuzzy Controller's Dynamical Operation

The figure on the cover of the book can serve as a nice visual depiction of how a fuzzy system operates dynamically over time. The figure represents a fuzzy system with two inputs, for example, e, and $\dot{e}$, and one output. There are triangular membership

functions on the two input universes of discourse, and minimum is used to represent the conjunction in the premise. The blue pyramids represent the premise certainties of the rules in a rule-base with 49 rules. Note that for simplicity of the graphic, the outermost membership functions do not saturate in this fuzzy controller; hence if e or $\dot{e}$ goes outside the range it appears that there will be no rules on, so the defuzzification will fail. Actually, the pyramids should be viewed as part of a rule-base with many more rules, and only the central ones for the rule-base are shown for simplicity.

The shading from blue, to red, to yellow, on the pyramids indicates progression in *time* of rules that were (are) on (i.e., the pyramids describing their premises had nonzero certainties) and the two in the middle that are fully shaded in yellow are the two rules that are on now. The pyramids with some blue on them, and some red, are ones that were on some time ago. The ones with red, and some yellow, were on more recently, while the ones that have a little less red shading and more yellow were on even more recently. The pyramids that are entirely blue, either were never turned on, or they were on a long time ago. Hence, the path of color (blue to red to yellow) could have traveled all over a large landscape of blue pyramids. At this time the path has come very near the $e = 0$, $\dot{e} = 0$ location in the rule-base and this is normally where you want it to be (for a tracking problem where $e = r - y$ where r is the reference input and y is the plant output we want $e = 0$ if y is to track r).

The colored vertical beam holds the four numbers that are the premise certainties for the four rules that are on now. Note that two of the rules that are on, are on with a certainty of zero, so really they are off and this is why they go to the output universe of discourse (top horizontal axis) at the zero level of certainty (see the top figure with the tan-colored output membership functions). The colored vertical beam contains only green and orange since these represent the values of the premise certainties from the two rules that are on. The beam does not have any purple or pink in it as these colors represent the zero values of the premises of the two rules that are off (we have constructed the rule-base so that there are at most four rules on at any time). The green and orange values chop the tops off two triangular output membership functions that then become the implied fuzzy sets (i.e., we use minimum to represent the implication). The defuzzified value is shown as the arrow at the top (it looks like a COG defuzzification).

2.3 General Fuzzy Systems

In the previous section we provided an intuitive overview of fuzzy control via a simple example. In this section we will take a step back and examine the more general fuzzy system to show the range of possibilities that can be used in defining a fuzzy system and to solidify your understanding of fuzzy systems.[4] In particular,

4. Note that we limit our range of definition of the general fuzzy system (controller) to those that have found some degree of use in practical control applications. The reader interested in studying the more general mathematics of fuzzy sets, fuzzy logic, and fuzzy systems should consult [95, 250].

we will consider the case where there are many fuzzy controller inputs and outputs and where there are more general membership functions, fuzzification procedures, inference strategies, and defuzzification methods. Moreover, we introduce a class of "functional fuzzy systems" that have been found to be useful in some applications and characterize the general capabilities of fuzzy systems via the "universal approximation property."

This section is written to build on the previous one in the sense that we rely on our intuitive explanations for many of the concepts and provide a more mathematical and complete exposition on the details of the operation of fuzzy systems. The astute reader will actually see intuitively how to extend the basic fuzzy controller to the case where there are more than two inputs. While an understanding of how to define other types of membership functions (Section 2.3.2) is important since they are often used in practical applications, the remainder of the material in Sections 2.3.2– 2.3.5 and 2.3.8 can simply be viewed as a precise mathematical characterization and generalization of what you have already learned in Section 2.2. Section 2.3.6, and hence much of this section, is needed if you want to understand Chapter 5. Section 2.3.7 on page 73 is important to cover if you wish to understand all of Section 4.3 in Chapter 4, Chapter 5 (except Section 5.6), all of Section 7.2.2 in Chapter 7, and other ideas in the literature. In fact, Section 2.3.7, particularly the "Takagi-Sugeno fuzzy system," is one of the most important new concepts in this section.

Hence, if you are only concerned with gaining a basic understanding of fuzzy control you can skim the part in Section 2.3.2 on membership functions, teach yourself Section 2.3.7, and skip the remainder of this section on a first reading and come back to it later to deepen your understanding of fuzzy systems and the wide variety of ways that their basic components can be defined.

2.3.1 Linguistic Variables, Values, and Rules

A fuzzy system is a static nonlinear mapping between its inputs and outputs (i.e., it is not a dynamic system).[5] It is assumed that the fuzzy system has inputs $u_i \in \mathcal{U}_i$ where $i = 1, 2, \ldots, n$ and outputs $y_i \in \mathcal{Y}_i$ where $i = 1, 2, \ldots, m$, as shown in Figure 2.20. The inputs and outputs are "crisp"—that is, they are real numbers, not fuzzy sets. The fuzzification block converts the crisp inputs to fuzzy sets, the inference mechanism uses the fuzzy rules in the rule-base to produce fuzzy conclusions (e.g., the implied fuzzy sets), and the defuzzification block converts these fuzzy conclusions into the crisp outputs.

Universes of Discourse

The ordinary ("crisp") sets $\mathcal{U}_i$ and $\mathcal{Y}_i$ are called the "universes of discourse" for u_i and y_i, respectively (in other words, they are their domains). In practical ap-

5. Some people include the preprocessing of the inputs to the fuzzy system (e.g., differentiators or integrators) in the definition of the fuzzy system and thereby obtain a "fuzzy system" that *is* dynamic. Here, we adopt the convention that such preprocessing is not part of the fuzzy system, and hence the fuzzy system will always be a memoryless nonlinear map.

plications, most often the universes of discourse are simply the set of real numbers or some interval or subset of real numbers. Note that sometimes for convenience we will refer to an "effective" universe of discourse $[\alpha, \beta]$ where α and β are the points at which the outermost membership functions saturate for input universes of discourse, or the points beyond which the outputs will not move for the output universe of discourse. For example, for the $e(t)$ universe of discourse in Figure 2.12 on page 41 we have $\alpha = -\frac{\pi}{2}$ and $\beta = \frac{\pi}{2}$; or for the $u(t)$ universe of discourse in Figure 2.17 on page 47, we have $\alpha = -20$ and $\beta = 20$. However, the actual universe of discourse for both the input and output membership functions for the inverted pendulum is the set of all real numbers. When we refer to effective universes of discourse, we will say that the "width" of the universe of discourse is $|\beta - \alpha|$.

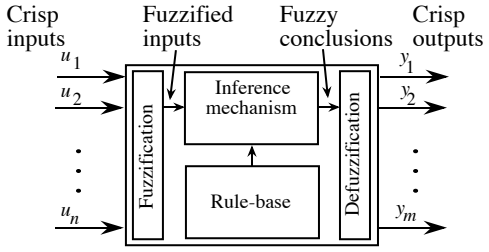


FIGURE 2.20 Fuzzy system (controller).

Linguistic Variables

To specify rules for the rule-base, the expert will use a "linguistic description"; hence, linguistic expressions are needed for the inputs and outputs and the characteristics of the inputs and outputs. We will use "linguistic variables" (constant symbolic descriptions of what are in general time-varying quantities) to describe fuzzy system inputs and outputs. For our fuzzy system, linguistic variables denoted by $\tilde{u}_i$ are used to describe the inputs u_i. Similarly, linguistic variables denoted by $\tilde{y}_i$ are used to describe outputs y_i. For instance, an input to the fuzzy system may be described as $\tilde{u}_1 =$"position error" or $\tilde{u}_2 =$"velocity error," and an output from the fuzzy system may be $\tilde{y}_1 =$"voltage in."

Linguistic Values

Just as u_i and y_i take on values over each universe of discourse $\mathcal{U}_i$ and $\mathcal{Y}_i$, respectively, linguistic variables $\tilde{u}_i$ and $\tilde{y}_i$ take on "linguistic values" that are used to describe characteristics of the variables. Let $\tilde{A}_i^j$ denote the j^{th} linguistic value of the linguistic variable $\tilde{u}_i$ defined over the universe of discourse $\mathcal{U}_i$. If we assume that there exist many linguistic values defined over $\mathcal{U}_i$, then the linguistic variable $\tilde{u}_i$ takes on the elements from the set of linguistic values denoted by

$$\tilde{A}_i = \{\tilde{A}_i^j : j = 1, 2, \ldots, N_i\}$$

(sometimes for convenience we will let the j indices take on negative integer values, as in the inverted pendulum example where we used the linguistic-numeric values). Similarly, let $\tilde{B}_i^j$ denote the j^{th} linguistic value of the linguistic variable $\tilde{y}_i$ defined over the universe of discourse $\mathcal{Y}_i$. The linguistic variable $\tilde{y}_i$ takes on elements from the set of linguistic values denoted by

$$\tilde{B}_i = \{\tilde{B}_i^p : p = 1, 2, \ldots, M_i\}$$

(sometimes for convenience we will let the p indices take on negative integer values). Linguistic values are generally descriptive terms such as "positive large," "zero," and "negative big" (i.e., adjectives). For example, if we assume that $\tilde{u}_1$ denotes the linguistic variable "speed," then we may assign $\tilde{A}_1^1$ = "slow," $\tilde{A}_1^2$ = "medium," and $\tilde{A}_1^3$ = "fast" so that $\tilde{u}_1$ has a value from $\tilde{A}_1 = \{\tilde{A}_1^1, \tilde{A}_1^2, \tilde{A}_1^3\}$.

Linguistic Rules

The mapping of the inputs to the outputs for a fuzzy system is in part characterized by a set of *condition* $\rightarrow$ *action* rules, or in *modus ponens* (If-Then) form,

$$\textbf{If} \text{ premise } \textbf{Then} \text{ consequent}. \tag{2.3}$$

Usually, the inputs of the fuzzy system are associated with the premise, and the outputs are associated with the consequent. These If-Then rules can be represented in many forms. Two standard forms, multi-input multi-output (MIMO) and multi-input single-output (MISO), are considered here. The MISO form of a linguistic rule is

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{A}_2^k \textbf{ and}, \ldots, \textbf{and } \tilde{u}_n \text{ is } \tilde{A}_n^l \textbf{ Then } \tilde{y}_q \text{ is } \tilde{B}_q^p \tag{2.4}$$

It is an entire set of linguistic rules of this form that the expert specifies on how to control the system. Note that if $\tilde{u}_1$ ="velocity error" and $\tilde{A}_1^j$ ="positive large," then "$\tilde{u}_1$ is $\tilde{A}_1^j$," a single term in the premise of the rule, means "velocity error is positive large." It can be easily shown that the MIMO form for a rule (i.e., one with consequents that have terms associated with each of the fuzzy controller outputs) can be decomposed into a number of MISO rules using simple rules from logic. For instance, the MIMO rule with n inputs and $m = 2$ outputs

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{A}_2^k \textbf{ and}, \ldots, \textbf{and } \tilde{u}_n \text{ is } \tilde{A}_n^l \textbf{ Then } \tilde{y}_1 \text{ is } \tilde{B}_1^r \textbf{ and } \tilde{y}_2 \text{ is } \tilde{B}_2^s$$

is linguistically (logically) equivalent to the two rules

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{A}_2^k \textbf{ and}, \ldots, \textbf{and } \tilde{u}_n \text{ is } \tilde{A}_n^l \textbf{ Then } \tilde{y}_1 \text{ is } \tilde{B}_1^r$$

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{A}_2^k \textbf{ and}, \ldots, \textbf{and } \tilde{u}_n \text{ is } \tilde{A}_n^l \textbf{ Then } \tilde{y}_2 \text{ is } \tilde{B}_2^s$$

This is the case since the logical "and" in the consequent of the MIMO rule is still represented in the two MISO rules since we still assert that both the first "and" second rule are valid. For implementation, we would specify two fuzzy systems, one with output y_1 and the other with output y_2. The logical "and" in the consequent of the MIMO rule is still represented in the MISO case since by implementing two fuzzy systems we are asserting that ones set of rules is true "and" another is true.

We assume that there are a total of R rules in the rule-base numbered $1, 2, \ldots, R$, and we naturally assume that the rules in the rule-base are distinct (i.e., there are no two rules with exactly the same premises and consequents); however, this does not in general need to be the case. For simplicity we will use tuples

$$(j, k, \ldots, l; p, q)_i$$

to denote the i^{th} MISO rule of the form given in Equation (2.4). Any of the terms associated with any of the inputs for any MISO rule can be included or omitted. For instance, suppose a fuzzy system has two inputs and one output with $\tilde{u}_1 =$ "position," $\tilde{u}_2 =$ "velocity," and $\tilde{y}_1 =$ "force." Moreover, suppose each input is characterized by two linguistic values $\tilde{A}_i^1 =$ "small" and $\tilde{A}_i^2 =$ "large" for $i = 1, 2$. Suppose further that the output is characterized by two linguistic values $\tilde{B}_1^1 =$ "negative" and $\tilde{B}_1^2 =$ "positive." A valid If-Then rule could be

If position is large **Then** force is positive

even though it does not follow the format of a MISO rule given above. In this case, one premise term (linguistic variable) has been omitted from the If-Then rule. We see that we allow for the case where the expert does not use all the linguistic terms (and hence the fuzzy sets that characterize them) to state some rules.[6]

Finally, we note that if all the premise terms are used in every rule and a rule is formed for each possible combination of premise elements, then there are

$$\prod_{i=1}^{n} N_i = N_1 \cdot N_2 \cdot \ldots \cdot N_n$$

rules in the rule-base. For example, if $n = 2$ inputs and we have $N_i = 11$ membership functions on each universe of discourse, then there are $11 \times 11 = 121$ possible rules. Clearly, in this case the number of rules increases exponentially with an increase in the number of fuzzy controller inputs or membership functions.

2.3.2 Fuzzy Sets, Fuzzy Logic, and the Rule-Base

Fuzzy sets and fuzzy logic are used to heuristically quantify the meaning of linguistic variables, linguistic values, and linguistic rules that are specified by the expert. The concept of a fuzzy set is introduced by first defining a "membership function."

6. Note, however, that we could require the rules to each have every premise term. Then we can choose a special membership function that is unity over the entire universe of discourse and associate it with any premise term that we want to omit. This achieves the same objective as simply ignoring a premise term. Why?

Membership Functions

Let $\mathcal{U}_i$ denote a universe of discourse and $\tilde{A}_i^j \in \tilde{A}_i$ denote a specific linguistic value for the linguistic variable $\tilde{u}_i$. The function $\mu(u_i)$ associated with $\tilde{A}_i^j$ that maps $\mathcal{U}_i$ to $[0, 1]$ is called a "membership function." This membership function describes the "certainty" that an element of $\mathcal{U}_i$, denoted u_i, with a linguistic description $\tilde{u}_i$, may be classified linguistically as $\tilde{A}_i^j$. Membership functions are subjectively specified in an ad hoc (heuristic) manner from experience or intuition.

For instance, if $\mathcal{U}_i = [-150, 150]$, $\tilde{u}_i$ ="velocity error," and $\tilde{A}_i^j$ ="positive large," then $\mu(u_i)$ may be a bell-shaped curve that peaks at one at $u_i = 75$ and is near zero when $u_i < 50$ or $u_i > 100$. Then if $u_i = 75$, $\mu(75) = 1$, so we are absolutely certain that u_i is "positive large." If $u_i = -25$ then $\mu(-25)$ is very near zero, which represents that we are very certain that u_i is not "positive large."

Clearly, many other choices for the shape of the membership function are possible (e.g., triangular and trapezoidal shapes), and these will each provide a different meaning for the linguistic values that they quantify. See Figure 2.21 for a graphical illustration of a variety of membership functions and Tables 2.3 and 2.4 for a mathematical characterization of the triangular and Gaussian membership functions (other membership functions can be characterized with mathematics using a similar approach).[7] For practice, you should sketch the membership functions that are described in Tables 2.3 and 2.4. Notice that for Table 2.3 c^L specifies the "saturation point" and w^L specifies the slope of the nonunity and nonzero part of μ^L. Similarly, for μ^R. For μ^C notice that c is the center of the triangle and w is the base-width. Analogous definitions are used for the parameters in Table 2.4. In Table 2.4, for the "centers" case note that this is the traditional definition for the Gaussian membership function. This definition is clearly different from a standard Gaussian probability density function, in both the meaning of c and σ, and in the scaling of the exponential function. Recall that it is possible that a Gaussian probability density function has a maximum value achieved at a value other than one; the standard Gaussian membership function always has its peak value at one.

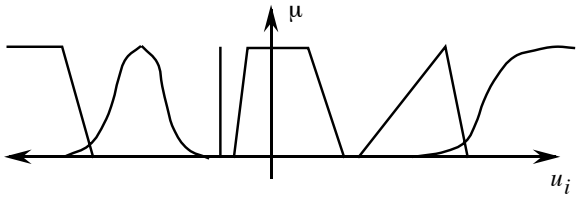


FIGURE 2.21 Some typical membership functions.

7. The reader should not fall into the trap of calling a membership function a "probability density function." There is nothing stochastic about the fuzzy system, and membership functions are not restricted to obey the laws of probability (consider, for example, the membership functions in Figure 2.21).

TABLE 2.3 Mathematical Characterization of Triangular Membership Functions

	Triangular membership functions
Left	$\mu^L(u) = \begin{cases} 1 & \text{if } u \le c^L \\ \max\left\{0, 1 + \frac{c^L - u}{0.5w^L}\right\} & \text{otherwise} \end{cases}$
Centers	$\mu^C(u) = \begin{cases} \max\left\{0, 1 + \frac{u-c}{0.5w}\right\} & \text{if } u \le c \\ \max\left\{0, 1 + \frac{c-u}{0.5w}\right\} & \text{otherwise} \end{cases}$
Right	$\mu^R(u) = \begin{cases} \max\left\{0, 1 + \frac{u-c^R}{0.5w^R}\right\} & \text{if } u \le c^R \\ 1 & \text{otherwise} \end{cases}$

TABLE 2.4 Mathematical Characterization of Gaussian Membership Functions

	Gaussian membership functions
Left	$\mu^L(u) = \begin{cases} 1 & \text{if } u \le c^L \\ \exp\left(-\frac{1}{2}\left(\frac{u-c^L}{\sigma^L}\right)^2\right) & \text{otherwise} \end{cases}$
Centers	$\mu(u) = \exp\left(-\frac{1}{2}\left(\frac{u-c}{\sigma}\right)^2\right)$
Right	$\mu^R(u) = \begin{cases} \exp\left(-\frac{1}{2}\left(\frac{u-c^R}{\sigma^R}\right)^2\right) & \text{if } u \le c^R \\ 1 & \text{otherwise} \end{cases}$

Fuzzy Sets

Given a linguistic variable $\tilde{u}_i$ with a linguistic value $\tilde{A}_i^j$ defined on the universe of discourse $\mathcal{U}_i$, and membership function $\mu_{A_i^j}(u_i)$ (membership function associated with the fuzzy set A_i^j) that maps $\mathcal{U}_i$ to $[0, 1]$, a "fuzzy set" denoted with A_i^j is defined as

$$A_i^j = \{(u_i, \mu_{A_i^j}(u_i)) : u_i \in \mathcal{U}_i\} \tag{2.5}$$

(notice that a fuzzy set is simply a crisp set of pairings of elements of the universe of discourse coupled with their associated membership values). For example, suppose we assign a linguistic variable $\tilde{u}_1$ = "temperature" and the linguistic value $\tilde{A}_1^1$ = "hot," then A_1^1 is a fuzzy set whose membership function describes the degree of certainty that the numeric value of the temperature, $u_1 \in \mathcal{U}_1$, possesses the property characterized by $\tilde{A}_1^1$ (see the pendulum example in the previous section for other examples of fuzzy sets).

Additional concepts related to membership functions and fuzzy sets are covered in Exercise 2.5 on page 104 and Exercise 2.6 on page 105. These include the following:

- *"Support of a fuzzy set":* The set of points on the universe of discourse where the membership function value is greater than zero.
- *"α-cut":* The set of points on the universe of discourse where the membership function value is greater than α.
- *"Height" of a fuzzy set or membership function:* The peak value reached by the membership function.
- *"Normal" fuzzy sets:* Ones with membership functions that reach one for at least one point on the universe of discourse.
- *"Convex fuzzy sets":* Ones that satisfy a certain type of convexity condition that is given in Equation (2.29) on page 104,
- *"Linguistic hedges":* Mathematical operations on membership functions of fuzzy sets that can be used to change the meaning of the underlying linguistics.
- *"Extension principle":* If you are given a function that maps some domain into some range and you have membership functions defined on the domain, the extension principle shows how to map the membership functions on the domain to the range.

Fuzzy Logic

Next, we specify some set-theoretic and logical operations on fuzzy sets. The reader should first understand the conventional counterparts to each of these; the fuzzy versions will then be easier to grasp as they are but extensions of the corresponding conventional notions. Also, we recommend that the reader sketch the fuzzy sets that result from the following operations.

Fuzzy Subset: Given fuzzy sets A_i^1 and A_i^2 associated with the universe of discourse $\mathcal{U}_i$ ($N_i = 2$), with membership functions denoted $\mu_{A_i^1}(u_i)$ and $\mu_{A_i^2}(u_i)$, respectively, A_i^1 is defined to be a "fuzzy subset" of A_i^2, denoted by $A_i^1 \subset A_i^2$, if $\mu_{A_i^1}(u_i) \leq \mu_{A_i^2}(u_i)$ for all $u_i \in \mathcal{U}_i$.

Fuzzy Complement: The complement ("not") of a fuzzy set A_i^1 with a membership function $\mu_{A_i^1}(u_i)$ has a membership function given by $1 - \mu_{A_i^1}(u_i)$.

Fuzzy Intersection (AND): The intersection of fuzzy sets A_i^1 and A_i^2, which are defined on the universe of discourse $\mathcal{U}_i$, is a fuzzy set denoted by $A_i^1 \cap A_i^2$, with a membership function defined by either of the following two methods:

1. *Minimum:* Here, we find the minimum of the membership values as in

$$\mu_{A_i^1 \cap A_i^2} = \min\{\mu_{A_i^1}(u_i), \mu_{A_i^2}(u_i) : u_i \in \mathcal{U}_i\} \tag{2.6}$$

2. *Algebraic Product:* Here, we find the product of the membership values as in

$$\mu_{A_i^1 \cap A_i^2} = \{\mu_{A_i^1}(u_i)\mu_{A_i^2}(u_i) : u_i \in \mathcal{U}_i\} \tag{2.7}$$

Other methods can be used to represent intersection (and) [95, 250], such as the ones given in Exercise 2.7 on page 105, but the two listed above are the most commonly used. Suppose that we use the notation $x * y = \min\{x, y\}$, or at other times we will use it to denote the product $x * y = xy$ ($*$ is sometimes called the "triangular norm"). Then $\mu_{A_i^1}(u_i) * \mu_{A_i^2}(u_i)$ is a general representation for the intersection of two fuzzy sets. In fuzzy logic, intersection is used to represent the "and" operation. For example, if we use minimum to represent the "and" operation, then the shaded membership function in Figure 2.22 is $\mu_{A_i^1 \cap A_i^2}$, which is formed from the two others ($\mu_{A_i^1}(u_i)$ and $\mu_{A_i^2}(u_i)$). This quantification of "and" provides the fundamental justification for our representation of the "and" in the premise of the rule.

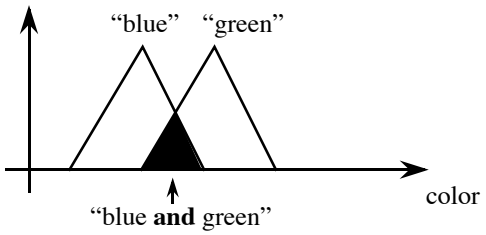


FIGURE 2.22 A membership function for the "and" of two membership functions.

Fuzzy Union (OR): The union of fuzzy sets A_i^1 and A_i^2, which are defined on the universe of discourse $\mathcal{U}_i$, is a fuzzy set denoted by $A_i^1 \cup A_i^2$, with a membership function defined by either one of the following methods:

1. *Maximum:* Here, we find the maximum of the membership values as in

$$\mu_{A_i^1 \cup A_i^2}(u_i) = \max\{\mu_{A_i^1}(u_i), \mu_{A_i^2}(u_i) : u_i \in \mathcal{U}_i\} \tag{2.8}$$

2. *Algebraic Sum:* Here, we find the algebraic sum of the membership values as in

$$\mu_{A_i^1 \cup A_i^2}(u_i) = \{\mu_{A_i^1}(u_i) + \mu_{A_i^2}(u_i) - \mu_{A_i^1}(u_i)\mu_{A_i^2}(u_i) : u_i \in \mathcal{U}_i\}. \tag{2.9}$$

Other methods can be used to represent union (or) [95, 250], such as the ones given in Exercise 2.7 on page 105, but the two listed above are the most commonly used. Suppose that we use the notation $x \oplus y = \max\{x, y\}$, or at other times we will use it to denote $x \oplus y = x + y - xy$ ($\oplus$ is sometimes called the "triangular co-norm").

Then $\mu_{A_i^1}(u_i) \oplus \mu_{A_i^2}(u_i)$ is a general representation for the union of two fuzzy sets. In fuzzy logic, union is used to represent the "or" operation. For example, if we use maximum to represent the "or" operation, then the shaded membership function in Figure 2.23, is $\mu_{A_i^1 \cup A_i^2}$, which is formed from the two others ($\mu_{A_i^1}(u_i)$ and $\mu_{A_i^2}(u_i)$). This quantification of "or" provides the fundamental justification for the "or" that inherently lies between the rules in the rule-base (note that we interpret the list of rules in the rule-base as "If premise-1 Then consequent-1" **or** "If premise-2 Then consequent-2," **or** so on). Note that in the case where we form the "overall implied fuzzy set" (to be defined more carefully below) this "or" between the rules is quantified directly with "$\oplus$" as it is described above. If we use only the implied fuzzy sets (as we did for the inverted pendulum problem in the last section), then the "or" between the rules is actually quantified with the way the defuzzification operation works (consider the way that the COG defuzzification method combines the effects of all the individual implied fuzzy sets).

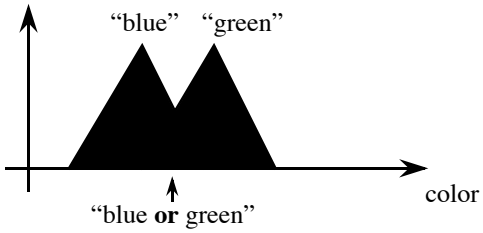


FIGURE 2.23 A membership function for the "or" of two membership functions.

Fuzzy Cartesian Product: The intersection and union above are both defined for fuzzy sets that lie on the same universe of discourse. The fuzzy Cartesian product is used to quantify operations on many universes of discourse. If $A_1^j, A_2^k, \ldots, A_n^l$ are fuzzy sets defined on the universes of discourse $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_n$, respectively, their Cartesian product is a fuzzy set (sometimes called a "fuzzy relation"), denoted by $A_1^j \times A_2^k \times \cdots \times A_n^l$, with a membership function defined by

$$\mu_{A_1^j \times A_2^k \times \cdots \times A_n^l}(u_1, u_2, \ldots, u_n) = \mu_{A_1^j}(u_1) * \mu_{A_2^k}(u_2) * \cdots * \mu_{A_n^l}(u_n)$$

The reader may wonder why the "$*$" operation is used here. Basically, it arises from our interpretation of a standard Cartesian product, which is formed by taking an element from the first element of the product "and" the second element of the product "and" so on. Clearly, in light of this interpretation, the use of "$*$" and hence "and" makes sense. Note that the "ands" used in the Cartesian product actually represent the "ands" used in the rule premises since normally each of the terms in a premise comes from a different universe of discourse.

Fuzzy Quantification of Rules: Fuzzy Implications

Next, we show how to quantify the linguistic elements in the premise and consequent of the linguistic If-Then rule with fuzzy sets. For example, suppose we are given the If-Then rule in MISO form in Equation (2.4). We can define the fuzzy sets as follows:

$$\begin{aligned} A_1^j &= \{(u_1, \mu_{A_1^j}(u_1)) : u_1 \in \mathcal{U}_1\} \\ A_2^k &= \{(u_2, \mu_{A_2^k}(u_2)) : u_2 \in \mathcal{U}_2\} \\ &\vdots \\ A_n^l &= \{(u_n, \mu_{A_n^l}(u_n)) : u_n \in \mathcal{U}_n\} \\ B_q^p &= \{(y_q, \mu_{B_q^p}(y_q)) : y_q \in \mathcal{Y}_q\} \end{aligned} \tag{2.10}$$

These fuzzy sets quantify the terms in the premise and the consequent of the given If-Then rule, to make a "fuzzy implication" (which is a fuzzy relation)

$$\textbf{If } A_1^j \textbf{ and } A_2^k \textbf{ and}, \ldots, \textbf{and } A_n^l \textbf{ Then } B_q^p \tag{2.11}$$

where the fuzzy sets $A_1^j, A_2^k, \ldots, A_n^l$, and B_q^p are defined in Equation (2.10). Therefore, the fuzzy set A_1^j is associated with, and quantifies the meaning of the linguistic statement "$\tilde{u}_1$ is $\tilde{A}_1^j$," and B_q^p quantifies the meaning of "$\tilde{y}_q$ is $\tilde{B}_q^p$." Each rule in the rule-base, which we denote by $(j, k, \ldots, l; p, q)_i$, $i = 1, 2, \ldots, R$, is represented with such a fuzzy implication (a fuzzy quantification of the linguistic rule).

There are two general properties of fuzzy logic rule-bases that are sometimes studied. These are "completeness" (i.e., whether there are conclusions for every possible fuzzy controller input) and "consistency" (i.e., whether the conclusions that rules make conflict with other rules' conclusions). These two properties are covered in Exercise 2.8 on page 106.

2.3.3 Fuzzification

Fuzzy sets are used to quantify the information in the rule-base, and the inference mechanism operates on fuzzy sets to produce fuzzy sets; hence, we must specify how the fuzzy system will convert its numeric inputs $u_i \in \mathcal{U}_i$ into fuzzy sets (a process called "fuzzification") so that they can be used by the fuzzy system.

Let $\mathcal{U}_i^*$ denote the set of all possible fuzzy sets that can be defined on $\mathcal{U}_i$. Given $u_i \in \mathcal{U}_i$, fuzzification transforms u_i to a fuzzy set denoted by[8] $\hat{A}_i^{\text{fuz}}$ defined on the universe of discourse $\mathcal{U}_i$. This transformation is produced by the fuzzification operator $\mathcal{F}$ defined by

$$\mathcal{F} : \mathcal{U}_i \to \mathcal{U}_i^*$$

8. In this section, as we introduce various fuzzy sets we will always use a hat over any fuzzy set whose membership function changes dynamically over time as the u_i change.

where

$$\mathcal{F}(u_i) = \hat{A}_i^{\text{fuz}}$$

Quite often "singleton fuzzification" is used, which produces a fuzzy set $\hat{A}_i^{\text{fuz}} \in \mathcal{U}_i^*$ with a membership function defined by

$$\mu_{\hat{A}_i^{\text{fuz}}}(x) = \begin{cases} 1 & x = u_i \\ 0 & \text{otherwise} \end{cases}$$

Any fuzzy set with this form for its membership function is called a "singleton." For a picture of a singleton membership function, see the single vertical line shown in Figure 2.21 on page 56. Note that the discrete impulse function can be used to represent the singleton membership function.

Basically, the reader should simply think of the singleton fuzzy set as a different representation for the number u_i. Singleton fuzzification is generally used in implementations since, without the presence of noise, we are absolutely certain that u_i takes on its measured value (and no other value), and since it provides certain savings in the computations needed to implement a fuzzy system (relative to, for example, "Gaussian fuzzification," which would involve forming bell-shaped membership functions about input points, or triangular fuzzification, which would use triangles).

Since most practical work in fuzzy control uses singleton fuzzification, we will also use it throughout the remainder of this book. The reasons other fuzzification methods have not been used very much are (1) they add computational complexity to the inference process and (2) the need for them has not been that well justified. This is partly due to the fact that very good functional capabilities can be achieved with the fuzzy system when only singleton fuzzification is used.

2.3.4 The Inference Mechanism

The inference mechanism has two basic tasks: (1) determining the extent to which each rule is relevant to the current situation as characterized by the inputs u_i, $i = 1, 2, \ldots, n$ (we call this task "matching"); and (2) drawing conclusions using the current inputs u_i and the information in the rule-base (we call this task an "inference step"). For matching note that $A_1^j \times A_2^k \times \cdots \times A_n^l$ is the fuzzy set representing the premise of the i^{th} rule $(j, k, \ldots, l; p, q)_i$ (there may be more than one such rule with this premise).

Matching

Suppose that at some time we get inputs u_i, $i = 1, 2, \ldots, n$, and fuzzification produces

$$\hat{A}_1^{\text{fuz}}, \hat{A}_2^{\text{fuz}}, \ldots, \hat{A}_n^{\text{fuz}}$$

the fuzzy sets representing the inputs. There are then two basic steps to matching.

Step 1: Combine Inputs with Rule Premises: The first step in matching involves finding fuzzy sets $\hat{A}_1^j, \hat{A}_2^k, \ldots, \hat{A}_n^l$ with membership functions

$$\begin{aligned}
\mu_{\hat{A}_1^j}(u_1) &= \mu_{A_1^j}(u_1) * \mu_{\hat{A}_1\text{fuz}}(u_1) \\
\mu_{\hat{A}_2^k}(u_2) &= \mu_{A_2^k}(u_2) * \mu_{\hat{A}_2\text{fuz}}(u_2) \\
&\vdots \\
\mu_{\hat{A}_n^l}(u_n) &= \mu_{A_n^l}(u_n) * \mu_{\hat{A}_n\text{fuz}}(u_n)
\end{aligned}$$

(for all $j, k, \ldots, l$) that combine the fuzzy sets from fuzzification with the fuzzy sets used in each of the terms in the premises of the rules. If singleton fuzzification is used, then each of these fuzzy sets is a singleton that is scaled by the premise membership function (e.g., $\mu_{\hat{A}_1^j}(\bar{u}_1) = \mu_{A_1^j}(\bar{u}_1)$ for $\bar{u}_1 = u_1$ and $\mu_{\hat{A}_1^j}(\bar{u}_1) = 0$ for $\bar{u}_1 \neq u_1$). That is, with singleton fuzzification we have $\mu_{\hat{A}_i\text{fuz}}(u_i) = 1$ for all $i = 1, 2, \ldots, n$ for the given u_i inputs so that

$$\begin{aligned}
\mu_{\hat{A}_1^j}(u_1) &= \mu_{A_1^j}(u_1) \\
\mu_{\hat{A}_2^k}(u_2) &= \mu_{A_2^k}(u_2) \\
&\vdots \\
\mu_{\hat{A}_n^l}(u_n) &= \mu_{A_n^l}(u_n)
\end{aligned}$$

We see that when singleton fuzzification is used, combining the fuzzy sets that were created by the fuzzification process to represent the inputs with the premise membership functions for the rules is particularly simple. It simply reduces to computing the membership values of the input fuzzy sets for the given inputs $u_1, u_2, \ldots, u_n$ (as we had indicated at the end of Section 2.2.3 for the inverted pendulum).

Step 2: Determine Which Rules Are On: In the second step, we form membership values $\mu_i(u_1, u_2, \ldots, u_n)$ for the i^{th} rule's premise (what we called $\mu_{premise}$ in the last section on the inverted pendulum) that represent the certainty that each rule premise holds for the given inputs. Define

$$\mu_i(u_1, u_2, \ldots, u_n) = \mu_{\hat{A}_1^j}(u_1) * \mu_{\hat{A}_2^k}(u_2) * \cdots * \mu_{\hat{A}_n^l}(u_n) \tag{2.12}$$

which is simply a function of the inputs u_i. When singleton fuzzification is used (as it is throughout this entire book), we have

$$\mu_i(u_1, u_2, \ldots, u_n) = \mu_{A_1^j}(u_1) * \mu_{A_2^k}(u_2) * \cdots * \mu_{A_n^l}(u_n) \tag{2.13}$$

We use $\mu_i(u_1, u_2, \ldots, u_n)$ to represent the certainty that the premise of rule i matches the input information when we use singleton fuzzification. This $\mu_i(u_1, u_2, \ldots, u_n)$ is simply a multidimensional certainty surface, a generalization of the surface shown

in Figure 2.11 on page 39 for the inverted pendulum example. It represents the certainty of a premise of a rule and thereby represents the degree to which a particular rule holds for a given set of inputs.

Finally, we would remark that sometimes an additional "rule certainty" is multiplied by μ_i. Such a certainty could represent our a priori confidence in each rule's applicability and would normally be a number between zero and one. If for rule i its certainty is 0.1, we are not very confident in the knowledge that it represents; while if for some rule j we let its certainty be 0.99, we are quite certain that the knowledge it represents is true. In this book we will not use such rule certainty factors.

This concludes the process of matching input information with the premises of the rules.

Inference Step

There are two standard alternatives to performing the inference step, one that involves the use of implied fuzzy sets (as we did for the pendulum earlier) and the other that uses the overall implied fuzzy set.

Alternative 1: Determine Implied Fuzzy Sets: Next, the inference step is taken by computing, for the i^{th} rule $(j, k, \ldots, l; p, q)_i$, the "implied fuzzy set" $\hat{B}_q^i$ with membership function

$$\mu_{\hat{B}_q^i}(y_q) = \mu_i(u_1, u_2, \ldots, u_n) * \mu_{B_q^p}(y_q) \tag{2.14}$$

The implied fuzzy set $\hat{B}_q^i$ specifies the certainty level that the output should be a specific crisp output y_q within the universe of discourse $\mathcal{Y}_q$, taking into consideration only rule i. Note that since $\mu_i(u_1, u_2, \ldots, u_n)$ will vary with time, so will the shape of the membership functions $\mu_{\hat{B}_q^i}(y_q)$ for each rule. An example of an implied fuzzy set can be seen in Figure 2.13(b) on page 43 for the inverted pendulum example.

Alternative 2: Determine the Overall Implied Fuzzy Set: Alternatively, the inference mechanism could, in addition, compute the "overall implied fuzzy set" $\hat{B}_q$ with membership function

$$\mu_{\hat{B}_q}(y_q) = \mu_{\hat{B}_q^1}(y_q) \oplus \mu_{\hat{B}_q^2}(y_q) \oplus \cdots \oplus \mu_{\hat{B}_q^R}(y_q) \tag{2.15}$$

that represents the conclusion reached considering all the rules in the rule-base at the same time (notice that determining $\hat{B}_q$ can, in general, require significant computational resources). Notice that we did not consider this possibility for the inverted pendulum example for reasons that will become clearer in the next subsection. Instead, our COG or center-average defuzzification method performed the aggregation of the conclusions of all the rules that are represented by the implied fuzzy sets.

Discussion: Compositional Rule of Inference Using the mathematical terminology of fuzzy sets, the computation of $\mu_{\hat{B}_q}(y_q)$ is said to be produced by a "sup-star compositional rule of inference." The "sup" in this terminology corresponds to the $\oplus$ operation, and the "star" corresponds to $*$. "Zadeh's compositional rule of inference" [245, 246, 95] is the special case of the sup-star compositional rule of inference when maximum is used for $\oplus$ and minimum is used for $*$. The overall justification for using the above operations to represent the inference step lies in the fact that *we can be no more certain about our conclusions than we are about our premises*. The operations performed in taking an inference step adhere to this principle. To see this, you should study Equation (2.14) and note that the scaling from $\mu_i(u_1, u_2, \ldots, u_n)$ that is produced by the premise matching process will always ensure that $\sup_{y_q}\{\mu_{\hat{B}_q^i}(y_q)\} \leq \mu_i(u_1, u_2, \ldots, u_n)$. The fact that we are no more certain of our consequents than our premises is shown graphically in Figure 2.19 on page 50 where the heights of the implied fuzzy sets are always less than the certainty values for all the premise terms.

Up to this point, we have used fuzzy logic to quantify the rules in the rule-base, fuzzification to produce fuzzy sets characterizing the inputs, and the inference mechanism to produce fuzzy sets representing the conclusions that it reaches after considering the current inputs and the information in the rule-base. Next, we look at how to convert this fuzzy set quantification of the conclusions to a numeric value that can be input to the plant.

2.3.5 Defuzzification

A number of defuzzification strategies exist, and it is not hard to invent more. Each provides a means to choose a single output (which we denote with y_q^{crisp}) based on either the implied fuzzy sets or the overall implied fuzzy set (depending on the type of inference strategy chosen, "Alternative 1 or 2," respectively, in the previous section).

Defuzzification: Implied Fuzzy Sets

As they are more common, we first specify typical defuzzification techniques for the implied fuzzy sets $\hat{B}_q^i$:

- *Center of gravity (COG):* A crisp output y_q^{crisp} is chosen using the center of area and area of each implied fuzzy set, and is given by

$$y_q^{\text{crisp}} = \frac{\sum_{i=1}^{R} b_i^q \int_{\mathcal{Y}_q} \mu_{\hat{B}_q^i}(y_q) dy_q}{\sum_{i=1}^{R} \int_{\mathcal{Y}_q} \mu_{\hat{B}_q^i}(y_q) dy_q}$$

  where R is the number of rules, b_i^q is the center of area of the membership function of B_q^p associated with the implied fuzzy set $\hat{B}_q^i$ for the i^{th} rule $(j, k, \ldots, l; p, q)_i$,

and

$$\int_{\mathcal{Y}_q} \mu_{\hat{B}_q^i}(y_q) dy_q$$

denotes the area under $\mu_{\hat{B}_q^i}(y_q)$. Notice that COG can be easy to compute since it is often easy to find closed-form expressions for $\int_{\mathcal{Y}_q} \mu_{\hat{B}_q^i}(y_q) dy_q$, which is the area under a membership function (see the pendulum example in Section 2.2.6 on page 44 where this amounts to finding the area of a triangle or a triangle with its top chopped off). Notice that the area under each implied fuzzy set must be computable, so the area under each of the output membership functions (that are used in the consequent of a rule) must be finite (this is why we cannot "saturate" the membership functions at the outermost edges of the output universe of discourse). Also, notice that the fuzzy system must be defined so that

$$\sum_{i=1}^{R} \int_{\mathcal{Y}_q} \mu_{\hat{B}_q^i}(y_q) dy_q \neq 0$$

for all u_i or y_q^{crisp} will not be properly defined. This value will be nonzero if there is a rule that is on for every possible combination of the fuzzy system inputs and the consequent fuzzy sets all have nonzero area.

- *Center-average:* A crisp output y_q^{crisp} is chosen using the centers of each of the output membership functions and the maximum certainty of each of the conclusions represented with the implied fuzzy sets, and is given by

$$y_q^{\text{crisp}} = \frac{\sum_{i=1}^{R} b_i^q \sup_{y_q}\{\mu_{\hat{B}_q^i}(y_q)\}}{\sum_{i=1}^{R} \sup_{y_q}\{\mu_{\hat{B}_q^i}(y_q)\}}$$

where "sup" denotes the "supremum" (i.e., the least upper bound which can often be thought of as the maximum value). Hence, $\sup_x\{\mu(x)\}$ can simply be thought of as the highest value of $\mu(x)$ (e.g., $\sup_u\{\mu_{(1)}(u)\} = 0.25$ for $\mu_{(1)}$ when product is used to represent the implication, as shown in Figure 2.18 on page 48). Also, b_i^q is the center of area of the membership function of B_q^p associated with the implied fuzzy set $\hat{B}_q^i$ for the i^{th} rule $(j, k, \ldots, l; p, q)_i$. Notice that the fuzzy system must be defined so that

$$\sum_{i=1}^{R} \sup_{y_q}\{\mu_{\hat{B}_q^i}(y_q)\} \neq 0$$

for all u_i. Also, note that $\sup_{y_q}\{\mu_{\hat{B}_q^i}(y_q)\}$ is often very easy to compute since if $\mu_{B_q^p}(y_q) = 1$ for at least one y_q (which is the normal way to define consequent

membership functions), then for many inference strategies, using Equation (2.14), we have

$$\sup_{y_q}\{\mu_{\hat{B}_q^i}(y_q)\} = \mu_i(u_1, u_2, \ldots, u_n)$$

which has already been computed in the matching process. Moreover, the formula for defuzzification is then given by

$$y_q^{\text{crisp}} = \frac{\sum_{i=1}^{R} b_i^q \mu_i(u_1, u_2, \ldots, u_n)}{\sum_{i=1}^{R} \mu_i(u_1, u_2, \ldots, u_n)} \tag{2.16}$$

where we must ensure that $\sum_{i=1}^{R} \mu_i(u_1, u_2, \ldots, u_n) \neq 0$ for all u_i. Also note that this implies that the shape of the membership functions for the output fuzzy sets does not matter; hence, you can simply use singletons centered at the appropriate positions. Convince yourself of this.

Defuzzification: The Overall Implied Fuzzy Set

Next, we present typical defuzzification techniques for the overall implied fuzzy set $\hat{B}_q$:

- *Max criterion:* A crisp output y_q^{crisp} is chosen as the point on the output universe of discourse $\mathcal{Y}_q$ for which the overall implied fuzzy set $\hat{B}_q$ achieves a maximum—that is,

$$y_q^{\text{crisp}} \in \left\{ \arg\sup_{\mathcal{Y}_q} \left\{ \mu_{\hat{B}_q}(y_q) \right\} \right\}$$

  Here, "$\arg\sup_x\{\mu(x)\}$" returns the value of x that results in the supremum of the function $\mu(x)$ being achieved. For example, suppose that $\mu_{overall}(u)$ denotes the membership function for the overall implied fuzzy set that is obtained by taking the maximum of the certainty values of $\mu_{(1)}$ and $\mu_{(2)}$ over all u in Figure 2.18 on page 48 (i.e., $\mu_{overall}(u) = \max_u\{\mu_{(1)}(u), \mu_{(2)}(u)\}$ per Equation (2.15)). In this case, $\arg\sup_u\{\mu_{overall}(u)\} = -10$, which is the defuzzified value via the max criterion.

  Sometimes the supremum can occur at more than one point in $\mathcal{Y}_q$ (e.g., consider the use of the max criterion for the case where minimum is used to represent the implication, and triangular membership functions are used on the output universe of discourse, such as in Figure 2.19 on page 50). In this case you also need to specify a strategy on how to pick only one point for y_q^{crisp} (e.g., choosing the smallest value). Often this defuzzification strategy is avoided due to this ambiguity; however, the next defuzzification method does offer a way around it.

- *Mean of maximum:* A crisp output y_q^{crisp} is chosen to represent the mean value of all elements whose membership in $\hat{B}_q$ is a maximum. We define $\hat{b}_q^{\max}$ as the supremum of the membership function of $\hat{B}_q$ over the universe of discourse $\mathcal{Y}_q$. Moreover, we define a fuzzy set $\hat{B}_q^* \in \mathcal{Y}_q$ with a membership function defined as

$$\mu_{\hat{B}_q^*}(y_q) = \begin{cases} 1 & \mu_{\hat{B}_q}(y_q) = \hat{b}_q^{\max} \\ 0 & \text{otherwise} \end{cases}$$

then a crisp output, using the mean of maximum method, is defined as

$$y_q^{\text{crisp}} = \frac{\int_{\mathcal{Y}_q} y_q \mu_{\hat{B}_q^*}(y_q) dy_q}{\int_{\mathcal{Y}_q} \mu_{\hat{B}_q^*}(y_q) dy_q} \tag{2.17}$$

where the fuzzy system must be defined so that $\int_{\mathcal{Y}_q} \mu_{\hat{B}_q^*}(y_q) dy_q \neq 0$ for all u_i. As an example, suppose that for Figure 2.19 on page 50 the two implied fuzzy sets are used to form an overall implied fuzzy set by taking the maximum of the two certainty values over all of u (i.e., $\mu_{overall}(u) = \max_u\{\mu_{(1)}(u), \mu_{(2)}(u)\}$ per Equation (2.15)). In this case there is an interval of u values around -10 where the overall implied fuzzy set is at its maximum value, and hence there is an ambiguity about which is the best defuzzified value. The mean of the maximum method would pick the value in the middle of the interval as the defuzzified value, so it would choose -10.

Note that the integrals in Equation (2.17) must be computed at each time instant since they depend on $\hat{B}_q$, which changes with time. This can require excessive computational resources for continuous universes of discourse. For some types of membership functions, simple ideas from geometry can be used to simplify the calculations; however, for some choices of membership functions, there may be many subintervals spread across the universe of discourse where the maximum is achieved. In these cases it can be quite difficult to compute the defuzzified value unless the membership functions are discretized. Complications such as these often cause designers to choose other defuzzification methods.

- *Center of area (COA):* A crisp output y_q^{crisp} is chosen as the center of area for the membership function of the overall implied fuzzy set $\hat{B}_q$. For a continuous output universe of discourse $\mathcal{Y}_q$, the center of area output is denoted by

$$y_q^{\text{crisp}} = \frac{\int_{\mathcal{Y}_q} y_q \mu_{\hat{B}_q}(y_q) dy_q}{\int_{\mathcal{Y}_q} \mu_{\hat{B}_q}(y_q) dy_q}$$

The fuzzy system must be defined so that $\int_{\mathcal{Y}_q} \mu_{\hat{B}_q}(y_q) dy_q \neq 0$ for all u_i. Note that, similar to the mean of the maximum method, this defuzzification approach can be computationally expensive. For instance, we leave it to the reader to compute the area of the overall implied fuzzy set $\mu_{overall}(u) = \max_u\{\mu_{(1)}(u), \mu_{(2)}(u)\}$ for

Figure 2.19 on page 50. Notice that in this case the computation is not as easy as just adding the areas of the two chopped-off triangles that represent the implied fuzzy sets. Computation of the area of the overall implied fuzzy set does not count the area that the implied fuzzy sets overlap twice; hence, the area of the overall implied fuzzy set can in general be much more difficult to compute in real time.

It is important to note that each of the above equations for defuzzification actually provides a mathematical quantification of the operation of the entire fuzzy system provided that each of the terms in the descriptions are fully defined. We discuss this in more detail in the next section.

Overall, we see that using the overall implied fuzzy set in defuzzification is often undesirable for two reasons: (1) the overall implied fuzzy set $\hat{B}_q$ is itself difficult to compute in general, and (2) the defuzzification techniques based on an inference mechanism that provides $\hat{B}_q$ are also difficult to compute. It is for this reason that most existing fuzzy controllers (including the ones in this book) use defuzzification techniques based on the implied fuzzy sets, such as center-average or COG.

2.3.6 Mathematical Representations of Fuzzy Systems

Notice that each formula for defuzzification in the previous section provides a mathematical description of a fuzzy system. There are many ways to represent the operations of a fuzzy system with mathematical formulas. Next, we clarify how to construct and interpret such mathematical formulas for the case where center-average defuzzification is used for MISO fuzzy systems. Similar ideas apply for some of the other defuzzification strategies, MIMO fuzzy systems, and the Takagi-Sugeno fuzzy systems that we discuss in the next section.

Assume that we use center-average defuzzification so that the formula describing how to compute the output is

$$y = \frac{\sum_{i=1}^{R} b_i \mu_i}{\sum_{i=1}^{R} \mu_i} \tag{2.18}$$

Notice that we removed the "crisp" superscript and "q" subscript from y (compare to Equation (2.16)). Also, we removed the "q" superscript from b_i. The q index is no longer needed in both cases since we are considering MISO systems, so that while there can be many inputs, there is only one output.

To be more explicit in Equation (2.18), we need to first define the premise membership functions μ_i in terms of the individual membership functions that describe each of the premise terms. Suppose that we use product to represent the conjunctions in the premise of each rule. Suppose that we use the triangular membership functions in Table 2.3 on page 57 where we suppose that $\mu_j^L(u_j)$ ($\mu_j^R(u_j)$) is the "left-" ("right-") most membership function on the j^{th} input universe of discourse. In addition, let $\mu_j^{C_i}(u_j)$ be the i^{th} "center" membership function for the j^{th} input universe of discourse. In this case, to define $\mu_j^L(u_j)$ we simply add a "j" subscript to the parameters of the "left" membership function from Table 2.3. In particular,

we use c_j^L and w_j^L to denote the j^{th} values of these parameters. We take a similar approach for the $\mu_j^R(u_j)$, $j = 1, 2, \ldots, n$. For $\mu_j^{C_i}(u_j)$ we use c_j^i (w_j^i) to denote the i^{th} triangle center (triangle base width) on the j^{th} input universe of discourse.

Suppose that we use all possible combinations of input membership functions to form the rules, and that each premise has a term associated with each and every input universe of discourse. A more detailed description of the fuzzy system in Equation (2.18) is given by

$$y = \frac{b_1 \prod_{j=1}^{n} \mu_j^L(u_j) + b_2 \mu_1^{C_1}(u_1) \prod_{j=2}^{n} \mu_j^L(u_j) + \cdots}{\prod_{j=1}^{n} \mu_j^L(u_j) + \mu_1^{C_1}(u_1) \prod_{j=2}^{n} \mu_j^L(u_j) + \cdots}$$

The first term in the numerator is $b_1\mu_1$ in Equation (2.18). Here, we have called the "first rule" the one that has premise terms all described by the membership functions $\mu_j^L(u_j)$, $j = 1, 2, \ldots, n$. The second term in the numerator is $b_2\mu_2$ and it uses $\mu_1^{C_1}(u_1)$ on the first universe of discourse and the leftmost ones on the other universes of discourse (i.e., $j = 2, 3, \ldots, n$). Continuing in a similar manner, the sum in the numerator (and denominator) extends to include all possible combinations of products of the input membership functions, and this fully defines the μ_i in Equation (2.18).

Overall, we see that because we need to define rules resulting from all possible combinations of *given* input membership functions, of which there are three kinds (left, center, right), the explicit mathematical representation of the fuzzy system is somewhat complicated. To avoid some of the complications, we first specify a single function that represents all three types of input membership functions. Suppose that on the j^{th} input universe of discourse we number the input membership functions from left to right as $1, 2, \ldots, N_j$, where N_j is the number of input membership functions on the j^{th} input universe of discourse. A single membership function that represents all three in Table 2.3 is

$$\mu_j^i(u_j) = \begin{cases} 1 & \text{if } u_j \le c_j^1 \text{ or } u_j \ge c_j^{N_j} \\ \max\left\{0, 1 + \frac{u_j - c_j^i}{0.5 w_j^i}\right\} & \text{if } u_j \le c_j^i \text{ and } (u_j > c_j^1 \text{ and } u_j < c_j^{N_j}) \\ \max\left\{0, 1 + \frac{c_j^i - u_j}{0.5 w_j^i}\right\} & \text{if } u_j > c_j^i \text{ and } (u_j > c_j^1 \text{ and } u_j < c_j^{N_j}) \end{cases}$$

A similar approach can be used for the Gaussian membership functions in Table 2.4.

Recall that we had used

$$(j, k, \ldots, l; p, q)_i$$

to denote the i^{th} rule. In this notation the indices in (the "tuple") $(j, k, \ldots, l)$ range over $1 \le j \le N_1$, $1 \le k \le N_2$, ..., $1 \le l \le N_n$, and specify which linguistic value is used on each input universe of discourse. Correspondingly, each index in the tuple $(j, k, \ldots, l)$ also specifies the linguistic-numeric value of the input membership function used on each input universe of discourse.

Let

$$b^{(j,k,\ldots,l;p,q)_i}$$

denote the output membership function (a singleton) center for the i^{th} rule (of course, $q = 1$ in our MISO case). Note that we use "i" in the notation $(j, k, \ldots, l; p, q)_i$ simply as a label for each rule (i.e., we number the rules in the rule-base, and i is this number). Hence, when we are given i, we know the values of j, k, ..., l, p, and q. Because of this, an explicit description of the fuzzy system in Equation (2.18) is given by

$$y = \frac{\sum_{i=1}^{R} b^{(j,k,\ldots,l;p,q)_i} \mu_1^j \mu_2^k \cdots \mu_n^l}{\sum_{i=1}^{R} \mu_1^j \mu_2^k \cdots \mu_n^l} \tag{2.19}$$

This formula clearly shows the use of the product to represent the premise. Notice that since we use all possible combinations of input membership functions to form the rules there are

$$R = \prod_{j=1}^{n} N_j$$

rules, and hence it takes

$$\sum_{j=1}^{n} 2N_j + \prod_{j=1}^{n} N_j \tag{2.20}$$

parameters to describe the fuzzy system since there are two parameters for each input membership function and R output membership function centers. For some applications, however, all the output membership functions are not distinct. For example, consider the pendulum example where five output membership function centers are defined, and there are $R = 25$ rules. To define the center positions $b^{(j,k,\ldots,l;p,q)_i}$ so that they take on only a fixed number of given values, that is less than R, one approach is to specify them as a function of the indices of the input membership functions. What is this function for the pendulum example?

A different approach to avoiding some of the complications encountered in specifying a fuzzy system mathematically is to use a different notation, and hence a different definition for the fuzzy system. For this alternative approach, for the sake of variety, we will use Gaussian input membership functions. In particular, for simplicity, suppose that for the input universes of discourse we only use membership functions of the "center" Gaussian form shown in Table 2.4. For the i^{th} rule, suppose that the input membership function is

$$\exp\left(-\frac{1}{2}\left(\frac{u_j - c_j^i}{\sigma_j^i}\right)^2\right)$$

for the j^{th} input universe of discourse. Hence, even though we use the same notation for the membership function, these centers c_j^i are different from those used above, both because we are using Gaussian membership functions here, and because the "i" in c_j^i is the index for the rules, not the membership function on the j^{th} input universe of discourse. Similar comments can be made about the σ_j^i, $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$. If we let b_i, $i = 1, 2, \ldots, R$, denote the center of the output membership function for the i^{th} rule, use center-average defuzzification, and product to represent the conjunctions in the premise, then

$$y = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{u_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{u_j - c_j^i}{\sigma_j^i}\right)^2\right)} \tag{2.21}$$

is an explicit representation of a fuzzy system. Note that we do not use the "left" and "right" versions of the Gaussian membership functions in Table 2.4 as this complicates the notation (how?). There are nR input membership function centers, nR input membership function spreads, and R output membership function centers. Hence, we need a total of

$$R(2n + 1)$$

parameters to describe this fuzzy system.

Now, while the fuzzy systems in Equations (2.19) and (2.21) are in general different, it is interesting to compare the number of parameters needed to describe a fuzzy system using each approach. In practical situations, we often have $N_j \geq 3$ for each $j = 1, 2, \ldots, n$, and sometimes the number of membership functions on each input universe of discourse can be quite large. From Equation (2.20) we can clearly see that large values of n will result in a fuzzy system with many parameters (there is an exponential increase in the number of rules). On the other hand, using the fuzzy system in Equation (2.21) the user specifies the number of rules and this, coupled with the number of inputs n, specifies the total number of parameters. There is not an exponential growth in the number of parameters in Equation (2.21) in the same way as there is in the fuzzy system in Equation (2.19) so you may be tempted to view the definition in Equation (2.21) as a better one. Such a conclusion, can, however be erroneous for several reasons.

First, the type of fuzzy system defined by Equation (2.19) is sometimes more natural in control design when you use triangular membership functions since you often need to make sure that there will be no point on any input universe of discourse where there is no membership function with a nonzero value (why?). Of course, if you are careful, you can avoid this problem with the fuzzy system represented by Equation (2.21) also. Second, suppose that the number of rules for Equation (2.21) is the same as that for Equation (2.19). In this case, the number of parameters

needed to describe the fuzzy system in Equation (2.21) is

$$\left(\prod_{j=1}^{n} N_j\right)(2n+1)$$

Now, comparing this to Equation (2.20) you see that for many values of N_j, $j = 1, 2, \ldots, n$, and number of inputs n, it is possible that the fuzzy system in Equation (2.21) will require many more parameters to specify it than the fuzzy system in Equation (2.19). Hence, the inefficiency in the representation in Equation (2.19) lies in having all possible combinations of output membership function centers, which results in exponential growth in the number of parameters needed to specify the fuzzy system. The inefficiency in the representation in Equation (2.21) lies in the fact that, in a sense, membership functions on the input universes of discourse are not re-used by each rule. There are new input membership functions for every rule.

Generally, it is difficult to know which is the best fuzzy system for a particular problem. In this book, we will sometimes (e.g., in Chapter 5) use the mathematical representation in Equation (2.21) because it is somewhat simpler, and possesses some properties that we will exploit. At other times we will be implicitly using the representation in Equation (2.19) because it will lend to the development of certain techniques (e.g., in Chapter 6). In every case, however, that we use Equation (2.21) (Equation (2.19)) you may want to consider how the concepts, approaches, and results change (or do not change) if the form of the fuzzy system in Equation (2.19) (Equation (2.21)) is used.

Finally, we would like to recommend that you practice creating mathematical representations of fuzzy systems. For instance, it is good practice to create a mathematical representation of the fuzzy controller for the inverted pendulum of the form of Equation (2.19), then also use Equation (2.21) to specify the same fuzzy system. Comparing these two approaches, and resolving the issues in specifying the output centers for the Equation (2.19) case, will help clarify the issues discussed in this section.

2.3.7 Takagi-Sugeno Fuzzy Systems

The fuzzy system defined in the previous sections will be referred to as a "standard fuzzy system." In this section we will define a "functional fuzzy system," of which the Takagi-Sugeno fuzzy system [207] is a special case.

For the functional fuzzy system, we use singleton fuzzification, and the i^{th} MISO rule has the form

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{A}_2^k \textbf{ and}, \ldots, \textbf{and } \tilde{u}_n \text{ is } \tilde{A}_n^l \textbf{ Then } b_i = g_i(\cdot)$$

where "$\cdot$" simply represents the argument of the function g_i and the b_i are *not* output membership function centers. The premise of this rule is defined the same as it is for the MISO rule for the standard fuzzy system in Equation (2.4) on page 54.

The consequents of the rules are different, however. Instead of a linguistic term with an associated membership function, in the consequent we use a *function* $b_i = g_i(\cdot)$ (hence the name "functional fuzzy system") that does not have an associated membership function. Notice that often the argument of g_i contains the terms $u_i, i = 1, 2, \ldots, n$, but other variables may also be used. The choice of the function depends on the application being considered. Below, we will discuss linear and affine functions but many others are possible. For instance, you may want to choose

$$b_i = g_i(\cdot) = a_{i,0} + a_{i,1}(u_1)^2 + \cdots + a_{i,n}(u_n)^2$$

or

$$b_i = g_i(\cdot) = \exp\left[a_{i,1}sin(u_1) + \cdots + a_{i,n}sin(u_n)\right]$$

Virtually any function can be used (e.g., a neural network mapping or another fuzzy system), which makes the functional fuzzy system very general.

For the functional fuzzy system we can use an appropriate operation for representing the premise (e.g., minimum or product), and defuzzification may be obtained using

$$y = \frac{\sum_{i=1}^{R} b_i \mu_i}{\sum_{i=1}^{R} \mu_i} \tag{2.22}$$

where μ_i is defined in Equation (2.13). It is assumed that the functional fuzzy system is defined so that no matter what its inputs are, we have $\sum_{i=1}^{R} \mu_i \neq 0$. One way to view the functional fuzzy system is as a nonlinear interpolator between the mappings that are defined by the functions in the consequents of the rules.

An Interpolator Between Linear Mappings

In the case where

$$b_i = g_i(\cdot) = a_{i,0} + a_{i,1}u_1 + \cdots + a_{i,n}u_n$$

(where the $a_{i,j}$ are real numbers) the functional fuzzy system is referred to as a "Takagi-Sugeno fuzzy system." If $a_{i,0} = 0$, then the $g_i(\cdot)$ mapping is a linear mapping and if $a_{i,0} \neq 0$, then the mapping is called "affine." Often, however, as is standard, we will refer to the affine mapping as a linear mapping for convenience. Overall, we see that the Takagi-Sugeno fuzzy system essentially performs a nonlinear interpolation between linear mappings.

As an example, suppose that $n = 1$, $R = 2$, and that we have rules

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^1 \textbf{ Then } b_1 = 2 + u_1$$

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^2 \textbf{ Then } b_2 = 1 + u_1$$

with the universe of discourse for u_1 given in Figure 2.24 so that μ_1 represents $\tilde{A}_1^1$ and μ_2 represents $\tilde{A}_1^2$. We have

$$y = \frac{b_1\mu_1 + b_2\mu_2}{\mu_1 + \mu_2} = b_1\mu_1 + b_2\mu_2$$

We see that for $u_1 > 1$, $\mu_1 = 0$, so $y = 1 + u_1$, which is a line. If $u_1 < -1$, $\mu_2 = 0$, so $y = 2 + u_1$, which is a different line. In between $-1 \leq u_1 \leq 1$, the output y is an interpolation between the two lines. Plot y versus u_1 to show how this interpolation is achieved.

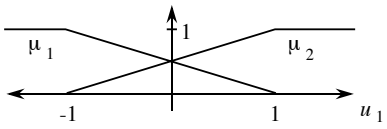


FIGURE 2.24 Membership functions for Takagi-Sugeno fuzzy system example.

Finally, it is interesting to note that if we pick

$$g_i = a_{i,0}$$

(i.e., $a_{i,j} = 0$ for $j > 0$), then the Takagi-Sugeno fuzzy system is equivalent to a standard fuzzy system that uses center-average defuzzification with singleton output membership functions at $a_{i,0}$. It is in this sense that the Takagi-Sugeno fuzzy system—or, more generally, the functional fuzzy system—is sometimes referred to as a "general fuzzy system."

An Interpolator Between Linear Systems

It is important to note that a Takagi-Sugeno fuzzy system may have any linear mapping (affine mapping) as its output function, which also contributes to its generality. One mapping that has proven to be particularly useful is to have a linear dynamic system as the output function so that the i^{th} rule has the form

If $\tilde{z}_1$ is $\tilde{A}_1^j$ **and** $\tilde{z}_2$ is $\tilde{A}_2^k$ **and**, $\ldots$, **and** $\tilde{z}_p$ is $\tilde{A}_p^l$ **Then** $\dot{x}^i(t) = A_i x(t) + B_i u(t)$

Here, $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^\top$ is the n-dimensional state (now n is not necessarily the number of inputs); $u(t) = [u_1(t), u_2(t), \ldots, u_m(t)]^\top$ is the m-dimensional model input; A_i and B_i, $i = 1, 2, \ldots, R$ are the state and input matrices of appropriate dimension; and $z(t) = [z_1(t), z_2(t), \ldots, z_p(t)]^\top$ is the p-dimensional input to the fuzzy system. This fuzzy system can be thought of as a nonlinear interpolator

between R linear systems. It takes the input $z(t)$ and has an output

$$\dot{x}(t) = \frac{\sum_{i=1}^{R}(A_i x(t) + B_i u(t))\mu_i(z(t))}{\sum_{i=1}^{R}\mu_i(z(t))}$$

or

$$\dot{x}(t) = \left(\sum_{i=1}^{R} A_i \xi_i(z(t))\right) x(t) + \left(\sum_{i=1}^{R} B_i \xi_i(z(t))\right) u(t) \tag{2.23}$$

where

$$\xi^\top = [\xi_1, \ldots, \xi_R] = \left[\frac{1}{\sum_{i=1}^{R}\mu_i}\right][\mu_1, \ldots, \mu_R]$$

If $R = 1$, we get a standard linear system. Generally, for $R > 1$ and a given value of $z(t)$, only certain rules will turn on and contribute to the output. Many choices are possible for $z(t)$. For instance, we often choose $z(t) = x(t)$, or sometimes $z(t) = [x^\top(t), u^\top(t)]^\top$.

As an example, suppose that $z(t) = x(t)$, $p = n = m = 1$, and $R = 2$ with rules

$$\textbf{If } \tilde{x}_1 \text{ is } \tilde{A}_1^1 \textbf{ Then } \dot{x}^1 = -x_1 + 2u_1$$

$$\textbf{If } \tilde{x}_1 \text{ is } \tilde{A}_1^2 \textbf{ Then } \dot{x}^2 = -2x_1 + u_1.$$

Suppose that we use μ_1 and μ_2 from Figure 2.24 as the membership functions for $\tilde{A}_1^1$ and $\tilde{A}_1^2$, respectively (i.e., we relabel the horizontal axis of Figure 2.24 with x_1). In this case Equation (2.23) becomes

$$\dot{x}_1(t) = (-\mu_1 - 2\mu_2)\, x_1(t) + (2\mu_1 + \mu_2)\, u_1(t)$$

If $x_1(t) > 1$, then $\mu_1 = 0$ and $\mu_2 = 1$, so the behavior of the nonlinear system is governed by

$$\dot{x}_1(t) = -2x_1(t) + u_1(t)$$

which is the linear system specified by the second rule above. However, if $x_1(t) < -1$, then $\mu_1 = 1$ and $\mu_2 = 0$, so the behavior of the nonlinear system is governed by

$$\dot{x}_1(t) = -x_1(t) + 2u_1(t)$$

which is the linear system specified by the first rule above. For $-1 \leq x_1(t) \leq 1$, the Takagi-Sugeno fuzzy system interpolates between the two linear systems. We see that for changing values of $x_1(t)$, the two linear systems that are in the consequents of the rules contribute different amounts.

We think of one linear system being valid on a region of the state space that is quantified via μ_1 and another on the region quantified by μ_2 (with a "fuzzy boundary" in between). For the higher-dimensional case, we have premise membership functions for each rule quantify whether the linear system in the consequent is valid for a specific region on the state space. As the state evolves, different rules turn on, indicating that other combinations of linear models should be used. Overall, we find that the Takagi-Sugeno fuzzy system provides a very intuitive representation of a nonlinear system as a nonlinear interpolation between R linear models.

2.3.8 Fuzzy Systems Are Universal Approximators

Fuzzy systems have very strong functional capabilities. That is, if properly constructed, they can perform very complex operations (e.g., much more complex than those performed by a linear mapping). Actually, many fuzzy systems are known to satisfy the "universal approximation property" [227].

For example, suppose that we use center-average defuzzification, product for the premise and implication, and Gaussian membership functions. Name this fuzzy system $f(u)$. Then, for any real continuous function $\psi(u)$ defined on a closed and bounded set and an arbitrary $\epsilon > 0$, there exists a fuzzy system $f(u)$ such that

$$\sup_u |f(u) - \psi(u)| < \epsilon \, .$$

Note, however, that all this "universal approximation property" does is guarantee that there exists a way to define the fuzzy system $f(u)$ (e.g., by picking the membership function parameters). It does *not* say how to find the fuzzy system, which can, in general, be very difficult. Furthermore, for arbitrary accuracy you may need an arbitrarily large number of rules.

The value of the universal approximation property for fuzzy systems is simply that it shows that if you work hard enough at tuning, you should be able to make the fuzzy system do what you are trying to get done. For control, practically speaking, it means that there is great flexibility in tuning the nonlinear function implemented by the fuzzy controller. Generally, however, there are no guarantees that you will be able to meet your stability and performance specifications by properly tuning a given fuzzy controller. You also have to choose the appropriate controller inputs and outputs, and there will be fundamental limitations imposed by the plant that may prohibit achieving certain control objectives no matter how you tune the fuzzy controller (e.g., a nonminimum phase system may provide certain limits on the quality of the performance that can be achieved).

2.4 Simple Design Example: The Inverted Pendulum

As there is no general systematic procedure for the design of fuzzy controllers that will definitely produce a high-performance fuzzy control system for a wide variety of applications, it is necessary to learn about fuzzy controller design via examples.

Here, we continue with the inverted pendulum example to provide an introduction to the typical procedures used in the design (and redesign) of a fuzzy controller. After reading the next section, on simulation of fuzzy control systems, the reader can follow this section more carefully by fully reproducing our design steps. For a first reading, however, we recommend that you not worry about how the simulations were produced; rather, focus on their general characteristics as they are related to design.

To simulate the fuzzy control system shown in Figure 2.4 on page 27 it is necessary to specify a mathematical model of the inverted pendulum. Note that we did not need the model for the initial design of the fuzzy controller in Section 2.2.1; but to accurately assess the quality of a design, we need either a model for mathematical analysis or simulation-based studies, or an experimental test bed in which to evaluate the design. Here, we will study simulation-based evaluations for design, while in Chapter 4 we will study the use of mathematical analysis to verify the quality of a design (and to assist in redesign). Throughout the book we will also show actual implementation results that are used to assess the performance of fuzzy controllers.

One model for the inverted pendulum shown in Figure 2.2 on page 25 is given by

$$\begin{aligned} \ddot{y} &= \frac{9.8\sin(y) + \cos(y)\left[\frac{-\bar{u}-0.25\dot{y}^2\sin(y)}{1.5}\right]}{0.5\left[\frac{4}{3} - \frac{1}{3}\cos^2(y)\right]} \\ \dot{\bar{u}} &= -100\bar{u} + 100u. \end{aligned} \tag{2.24}$$

The first order filter on u to produce $\bar{u}$ represents an actuator. Given this and the fuzzy controller developed in Section 2.2.1 (the one that uses the minimum operator to represent both the "and" in the premise and the implication and COG defuzzification), we can simulate the fuzzy control system shown in Figure 2.4 on page 27. We let the initial condition be $y(0) = 0.1$ radians ($= 5.73$ deg.), $\dot{y}(0) = 0$, and the initial condition for the actuator state is zero. The results are shown in Figure 2.25, where we see in the upper plot that the output appropriately moves toward the inverted position, and the force input in the lower plot that moves back and forth to achieve this.[9]

2.4.1 Tuning via Scaling Universes of Discourse

Suppose that the rate at which the pendulum balances in Figure 2.25 is considered to be unacceptably slow and that there is too much control action. To solve these problems, we use standard ideas from control engineering to conclude that we ought to try to tune the "derivative gain." To do this we introduce gains on the

9. If you attempt to reproduce these results, you should be cautioned that, as always, inaccurate results can be obtained if a small enough integration step size is not chosen for numerical simulation. For all the simulation results of this section, we use the fourth-order Runge-Kutta method and an integration step size of 0.001. The plots of this subsection were produced by Scott C. Brown.

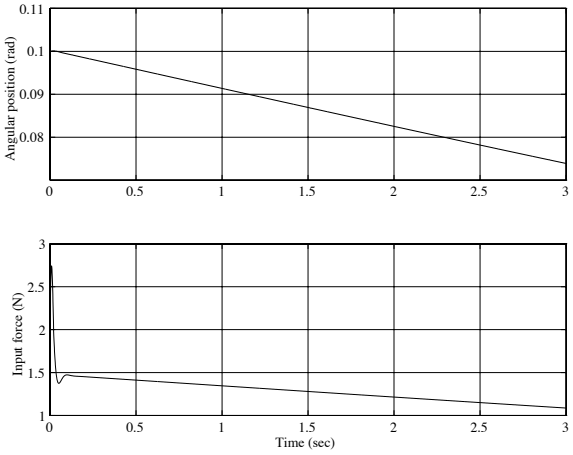


FIGURE 2.25 Fuzzy controller balancing an inverted pendulum, first design.

proportional and derivative terms, as shown in Figure 2.26, and at the same time we also put a gain h between the fuzzy controller and the inverted pendulum.

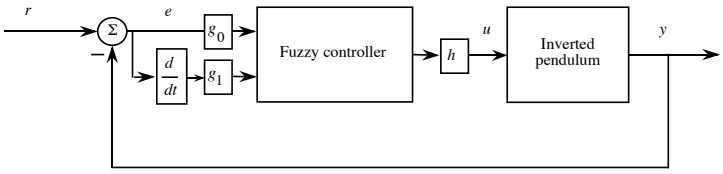


FIGURE 2.26 Fuzzy controller for inverted pendulum with scaling gains g_0, g_1, and h.

Choose $g_0 = 1$, $g_1 = 0.1$, and $h = 1$. To see the effect of this gain change, see Figure 2.27, where we see that the output angle reacts much faster and the control input is smoother.

If we still find the response of the pendulum rather slow, we may decide, using standard ideas from control engineering, that the proportional gain should be increased (often raising the "loop-gain" can speed up the system). Suppose next that we choose $g_0 = 2$, $g_1 = 0.1$, and $h = 1$—that is, we double the proportional gain. Figure 2.28 shows the resulting behavior of the fuzzy control system, where we see that the response is made significantly faster than in Figure 2.27. Actually, a similar effect to increasing the proportional gain can be achieved by increasing the output gain h. Choose $g_0 = 2$, $g_1 = 0.1$, and $h = 5$, and see Figure 2.29, where we see that the response is made even faster than in Figure 2.28. Indeed, as this is just a simulation study, we can increase h further and get even faster balancing provided that

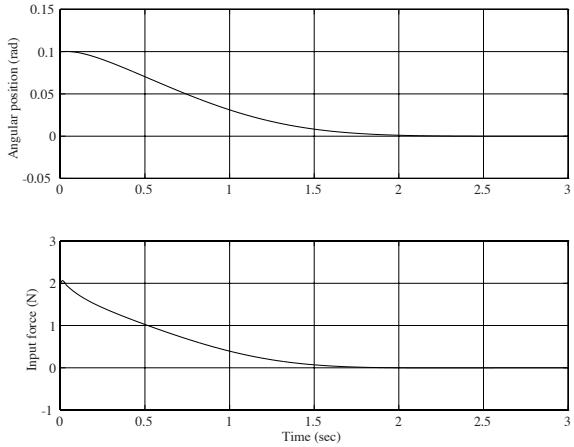


FIGURE 2.27 Fuzzy controller balancing an inverted pendulum with $g_0 = 1$, $g_1 = 0.1$, and $h = 1$.

we simulate the system properly by having a small enough integration step size. However, the reader must be cautioned that this may stretch the simulation model beyond its range of validity. For instance, further increases in h will generally result in faster balancing at the expense of a large control input, and for a big enough h the input may be larger than what is allowed in the physical system. At that point the simulation would not reflect reality since if the controller were actually implemented, the plant input would saturate and the proper balancing behavior may not be achieved.

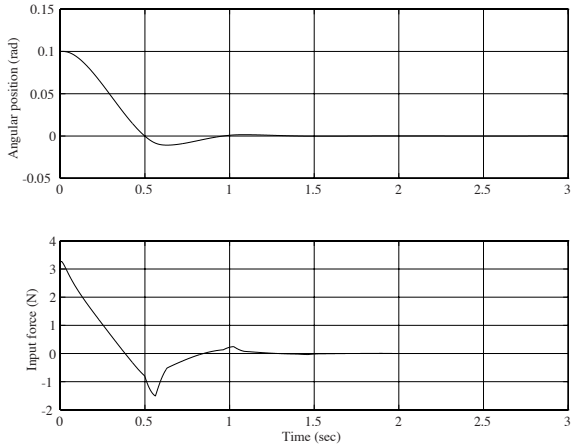


FIGURE 2.28 Fuzzy controller balancing an inverted pendulum with $g_0 = 2$, $g_1 = 0.1$, and $h = 1$.

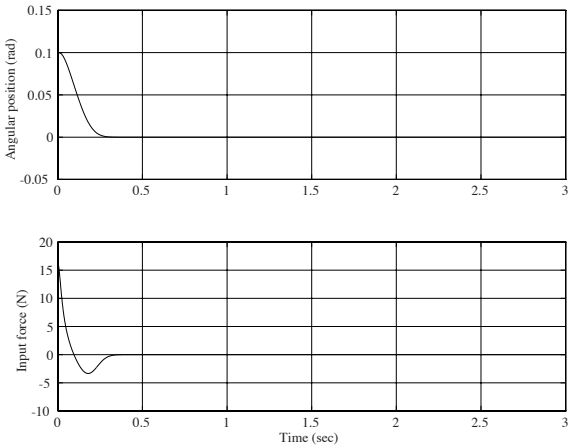


FIGURE 2.29 Fuzzy controller balancing an inverted pendulum with $g_0 = 2$, $g_1 = 0.1$, and $h = 5$.

We see that the change in the scaling gains at the input and output of the fuzzy controller can have a significant impact on the performance of the resulting fuzzy control system, and hence they are often a convenient parameter for tuning. Because they are frequently used for tuning fuzzy controllers, it is important to study exactly what happens when these scaling gains are tuned.

Input Scaling Gains

First, consider the effect of the input scaling gains g_0 and g_1. Notice that we can actually achieve the same effect as scaling via g_1 by simply changing the labeling of the $\frac{d}{dt}e(t)$ axis for the membership functions of that input. The case where $g_0 = g_1 = h = 1.0$ corresponds to our original choice for the membership functions in Figure 2.9 on page 36. The choice of $g_1 = 0.1$ as a scaling gain for the fuzzy controller with these membership functions is equivalent to having the membership functions shown in Figure 2.30 with a scaling gain of $g_1 = 1$.

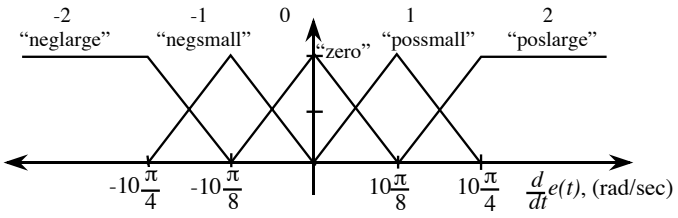


FIGURE 2.30 Scaled membership functions for $\frac{d}{dt}e(t)$.

We see that the choice of a scaling gain g_1 results in scaling the horizontal axis

of the membership functions by $\frac{1}{g_1}$. Generally, the scaling gain g_1 has the following effects:

- If $g_1 = 1$, there is no effect on the membership functions.
- If $g_1 < 1$, the membership functions are uniformly "spread out" by a factor of $\frac{1}{g_1}$ (notice that multiplication of each number on the horizontal axis of Figure 2.9 on page 36 by 10 produces Figure 2.30).
- If $g_1 > 1$, the membership functions are uniformly "contracted" (to see this, choose $g_1 = 10$ and notice that the numbers on the horizontal axis of the new membership functions that we would obtain by collapsing the gain into the choice of the membership functions, would be scaled by 0.1).

The expansion and contraction of the horizontal axes by the input scaling gains is sometimes described as similar to how an accordion operates, especially for triangular membership functions. Notice that the membership functions for the other input to the fuzzy controller will be affected in a similar way by the gain g_0.

Now that we see how we can either use input scaling gains or simply redefine the horizontal axis of the membership functions, it is interesting to consider how the scaling gains actually affect the meaning of the linguistics that form the basis for the definition of the fuzzy controller. Notice that

- If $g_1 = 1$, there is no effect on the meaning of the linguistic values.
- If $g_1 < 1$, since the membership functions are uniformly "spread out," this changes the meaning of the linguistics so that, for example, "poslarge" is now characterized by a membership function that represents larger numbers.
- If $g_1 > 1$, since the membership functions are uniformly "contracted," this changes the meaning of the linguistics so that, for example, "poslarge" is now characterized by a membership function that represents smaller numbers.

Similar statements can be made about all the other membership functions and their associated linguistic values. Overall, we see that the input scaling factors have an inverse relationship in terms of their ultimate effect on scaling (larger g_1 that is greater than 1 corresponds to changing the meaning of the linguistics so that they quantify smaller numbers). While such an inverse relationship exists for the input scaling gains, just the opposite effect is seen for the output scaling gains, as we shall see next.

Output Scaling Gain

Similar to what you can do to the input gains, you can collapse the output scaling gain into the definition of the membership functions on the output. In particular,

- If $h = 1$, there is no effect on the output membership functions.

- If $h < 1$, there is the effect of contracting the output membership functions and hence making the meaning of their associated linguistics quantify smaller numbers.
- If $h > 1$, there is the effect of spreading out the output membership functions and hence making the meaning of their associated linguistics quantify larger numbers.

There is a proportional effect between the scaling gain h and the output membership functions. As an example, for the inverted pendulum the output membership functions are scaled by h as shown in Figure 2.31. The reader should verify the effect of h by considering how the membership functions shown in Figure 2.31 will move for varying values of h.

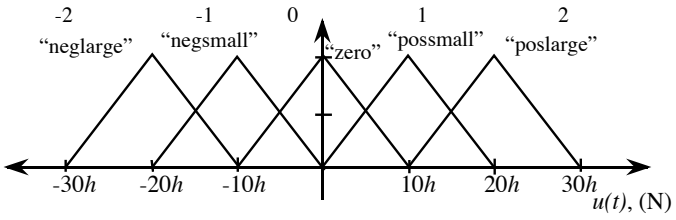


FIGURE 2.31 The effect of scaling gain h on the spacing of the output membership functions.

Overall, the tuning of scaling gains for fuzzy systems is often referred to as "scaling a fuzzy system." Notice that if for the pendulum example the effective universes of discourse for all inputs and outputs are $[-1, +1]$ (i.e., the input (output) left-most membership function saturates (peaks) at -1 and the right-most input (output) membership function saturates (peaks) at $+1$), then we say that the fuzzy controller is "normalized." Clearly, scaling gains can be used to normalize the given fuzzy controllers for the pendulum. What gains g_0, g_1, and h will do this?

2.4.2 Tuning Membership Functions

It is important to realize that the scaling gains are not the only parameters that can be tuned to improve the performance of the fuzzy control system. Indeed, sometimes it is the case that for a given rule-base and membership functions you cannot achieve the desired performance by tuning only the scaling gains. Often, what is needed is a more careful consideration of how to specify additional rules or better membership functions.

The problem with this is that there are often too many parameters to tune (e.g., membership function shapes, positioning, and number and type of rules) and often there is not a clear connection between the design objectives (e.g., better rise-time) and a rationale and method that should be used to tune these parameters. There are, however, certain methods to overcome this problem, and here we will examine

one of these that has been found to be very useful for real implementations of fuzzy control systems for challenging applications.

Output Membership Function Tuning

In this method we will tune the positioning of the output membership functions (assume that they are all symmetric and equal to one only at one point) by characterizing their centers by a function. Suppose that we use c^i, $i = -2, -1, 0, 1, 2$, to denote the centers of the output membership functions for the fuzzy controller for the inverted pendulum, where the indices i for the c^i are the linguistic-numeric values used for the output membership functions (see Figure 2.9 on page 36). (This is a different notation from that used for the centers in our discussion on defuzzification in Section 2.3.5 since there the index referred to the rule.) If $h = 1$ then

$$c^i = 10i$$

describes the positioning of the centers of the output membership functions shown in Figure 2.9 on page 36 and if we scale by h then

$$c^i = 10hi$$

describes the position centers as shown in Figure 2.31. We see that a linear relationship in the c^i equation produces a linear (uniform) spacing of the membership functions. Suppose that we instead choose

$$c^i = 5hsign(i)i^2 \tag{2.25}$$

($sign(x)$ returns the sign of the number x and $sign(0) = 1$), then this will have the effect of making the output membership function centers near the origin be more closely spaced than the membership functions farther out on the horizontal axis. The effect of this is to make the "gain" of the fuzzy controller smaller when the signals are small and larger as the signals grow larger (up to the point where there is a saturation, as usual). Hence, the use of Equation (2.25) for the centers indicates that if the error and change-in-error for the pendulum are near where they should be, then do not make the force input to the plant too big, but if the error and change-in-error are large, then the force input should be much bigger so that it quickly returns the pendulum to near the balanced position (note that a cubic function $c^i = 5hi^3$ will provide a similar effect as the $sign(i)i^2$ term in Equation (2.25)).

Effect on Performance

At this point the reader should wonder why we would even bother with more complex tuning of the fuzzy controller for the inverted pendulum since the performance seen in our last design iteration, in Figure 2.29 on page 81, was quite successful.

Consider, however, the effect of a disturbance such that during the previous simulation in Figure 2.29 on page 81 we let

$$u' = u + 600$$

for t such that $0.99 < t < 1.01$ sec where u' is now the force input to the pendulum and u is as before the output of the fuzzy controller (for $t \leq 0.99$ and $t \geq 1.01$, we let $u' = u$). This corresponds to a 600 Newton pulse on the input to the pendulum, and simulates the effect of someone bumping the cart so that we can study the ability of the controller to then rebalance the pendulum. The performance of our best controller up till now, shown in Figure 2.29 on page 81, is shown in Figure 2.32, where we see that the fuzzy controller fails to rebalance the pendulum when the cart is bumped.

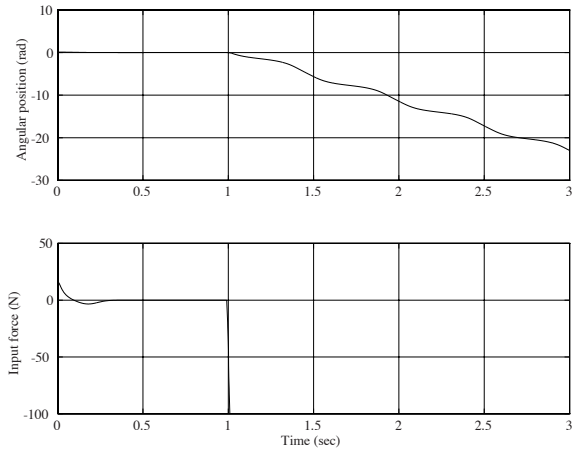


FIGURE 2.32 Effect of a disturbance (a bump to the cart of the pendulum) on the balancing capabilities of the fuzzy controller.

Suppose that to overcome this problem we decide that while the design in Figure 2.29 on page 81 was good for small-angle perturbations, something needs to be done for larger perturbations. In particular, let us attempt to use the fact that if there is a large variation from the inverted position there had better be a large enough input to get the pendulum closer to its inverted position so that it will not fall. To do this, we will use the above approach and choose

$$c^i = 5hsign(i)i^2$$

where $h = 10.0$ (we keep $g_0 = 2.0$ and $g_1 = 0.1$). If you were to simulate the resulting fuzzy control system for the case where there is no disturbance, you would find a performance that is virtually identical to that of the design that resulted in

Figure 2.29 on page 81. The reason for this can be explained as follows: Notice that for Figure 2.29 on page 81 the gains were $g_0 = 2.0$ and $g_1 = 0.1$ and that we have the output membership function centers given by

$$c^i = 5hi$$

where $h = 10$. Notice that for both controllers, if $i = 0$ or $i = 1$ we get the same positions of the output membership functions. Hence, if the signals are small, we will get nearly the same effect from both fuzzy controllers. However, if, for example, $i = 2$ then the center resulting from the controller with $c^i = 5hsign(i)i^2$ will have a membership function that is much farther out, which says that the input to the plant should be larger. The effect of this will be to have the fuzzy controller provide very large force inputs when the pendulum is not near its inverted position. To see this, consider Figure 2.33, where we see that the newly redesigned fuzzy controller can in fact rebalance the pendulum in the presence of the disturbance (and it performs similarly to the best previous one, shown in Figure 2.29 on page 81, in response to smaller perturbations from the inverted position, as is illustrated by how it recovers from the initial condition). Notice, however, that it used a large input force to counteract the bump to the pendulum.

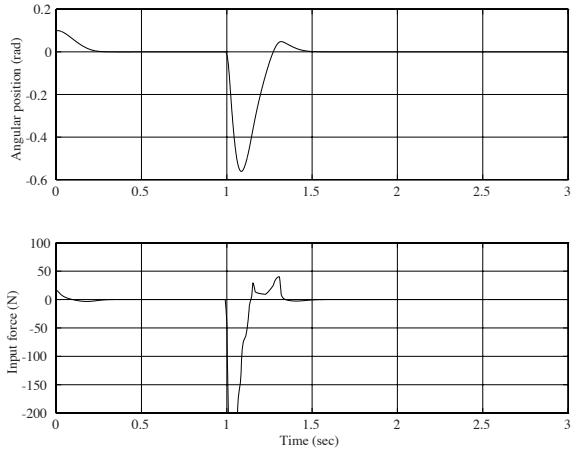


FIGURE 2.33 Effect of a disturbance (a bump to the cart of the pendulum) on the balancing capabilities of the fuzzy controller.

You may wonder why we did not just increase the gain on the fuzzy controller depicted in Figure 2.29 on page 81 to the point where it would be able to recover similarly to this new control system. However, if we did this, we would also raise the gain of the controller when its input signals are small, which can have adverse effects of amplifying noise in a real implementation. Besides, our redesign above

was used simply to illustrate the design approach. In the applications studied in Chapter 3, we will use a similar design approach where the need for the nonlinear spacing of the output membership functions is better motivated due to the fact that a more challenging application dictates this.

2.4.3 The Nonlinear Surface for the Fuzzy Controller

Ultimately, the goal of tuning is to shape the nonlinearity that is implemented by the fuzzy controller. This nonlinearity, sometimes called the "control surface," is affected by all the main fuzzy controller parameters. Consider, for example, the control surface for the fuzzy controller that resulted in the response shown in Figure 2.29 on page 81 (i.e., $g_0 = 2.0$, $g_1 = 0.1$, and $h = 5$), which is shown in Figure 2.34, where the output of the fuzzy controller is now plotted against its two inputs. Notice that the surface represents in a compact way all the information in the fuzzy controller (but of course this representation is limited in that if there are more than two inputs it becomes difficult to visualize the surface). To convince yourself of this, you should pick a value for e and $\frac{d}{dt}e(t)$, read the corresponding fuzzy controller output value off the surface, and determine if the rule-base would indicate that the controller should behave in this way. Figure 2.34 simply represents the range of possible defuzzified values for all possible inputs e and $\frac{d}{dt}e(t)$.

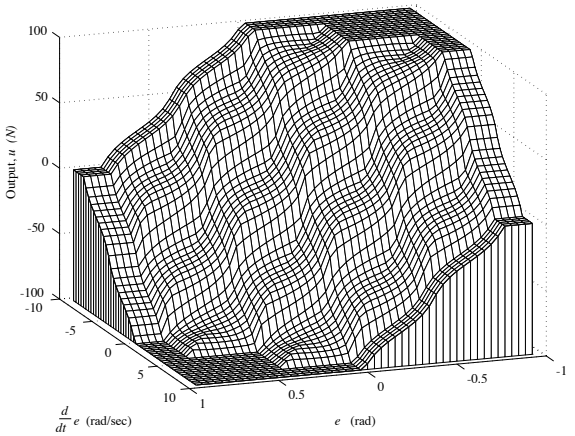


FIGURE 2.34 Control surface of the fuzzy controller for $g_0 = 2.0$, $g_1 = 0.1$, and $h = 5$.

Note that the control surface for a simple proportional-derivative (PD) controller is a plane in three dimensions. With the proper choice of the PD gains, the linear PD controller can easily be made to have the same shape as the fuzzy controller near the origin. Hence, in this case the fuzzy controller will behave similarly to the PD controller provided its inputs are small. However, notice that there is no way that the linear PD controller can achieve a nonlinear control surface of

the shape shown in Figure 2.34 (this is not surprising considering the complexity difference of the two controllers).

Next, notice changing the gains g_0 and g_1 will rescale the axis, which will change the slope of the surface. Increasing g_0 is analogous to increasing the proportional gain in a PD controller (i.e., it will often make the system respond faster). Increasing the gain g_1 is analogous to increasing the derivative gain in a PD controller. Notice, also, that changing h will scale the vertical axis of the controller surface plot. Hence, for instance, increasing h will make the entire surface have a higher slope and make the output saturate at higher values.

It is useful to notice that there is a type of interpolation that is performed by the fuzzy controller that is nicely illustrated in Figure 2.34. If you study the plot carefully, you will notice that the rippled surface is created by the rules and membership functions. For instance, if we kept a similar uniform distribution of membership functions for the input and outputs of the fuzzy system, but increased the number of membership functions, the ripples would correspondingly increase in number and the amplitude of the ripple would decrease (indeed, in the limit, as more and more membership functions are added in this way, the controller can be made to approximate a plane in a larger and larger region—but this may not occur for other membership function distributions and rule-base choices). What is happening is that there is an interpolation between the rules. The output is an interpolation of the effects of the four rules that are on for the inverted pendulum fuzzy controller. For more general fuzzy controllers, it is important to keep in mind that this sort of interpolation is often occurring (but not always—it depends on your choice of the membership functions).

When we tune the fuzzy controller, it changes the shape of the control surface, which in turn affects the behavior of the closed-loop control system. Changing the scaling gains changes the slope of the surface and hence the "gain" of the fuzzy controller as we discussed above and as we will discuss in Chapter 4 in more detail. The output membership function centers will also affect the shape of the surface. For instance, the control surface for the fuzzy controller that had

$$c^i = 5hsign(i)i^2$$

where $h = 10.0$, $g_0 = 2.0$, and $g_1 = 0.1$ is shown in Figure 2.35. You must carefully compare this surface to the one in Figure 2.34 to assess the effects of using the nonlinear spacing of the output membership function centers. Notice that near the center of the plot (i.e., where the inputs are zero) the shape of the two plots is nearly the same (i.e., as explained above, the two controllers will behave similarly for small input signals). Notice, however, that the slope of the surface is greater for larger signals in Figure 2.35 than in Figure 2.34. This further illustrates the effect of the choice of the nonlinear spacing for the output membership function centers.

This concludes the design process for the fuzzy controller for the pendulum. Certainly, if you were concerned with the design of a fuzzy controller for an industrial control problem, many other issues besides the ones addressed above would have to be considered. Here, we simply use the inverted pendulum as a convenient

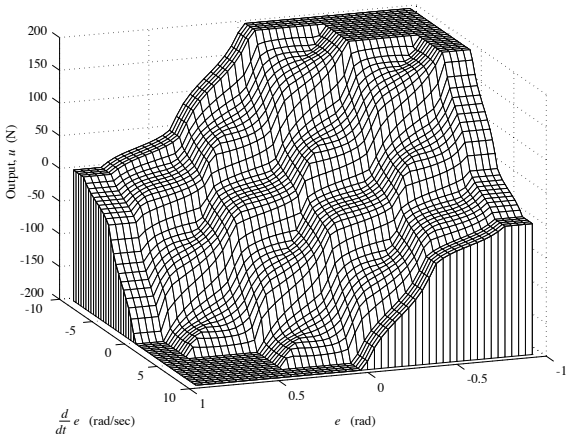


FIGURE 2.35 Control surface of the fuzzy controller for $c^i = 5hsign(i)i^2$, $h = 10.0$, $g_0 = 2.0$, and $g_1 = 0.1$.

example to illustrate the design procedures that are often used for fuzzy control systems. In Chapter 3 we will study several more fuzzy control design problems, several of which are much more challenging (and interesting) than the inverted pendulum studied here.

2.4.4 Summary: Basic Design Guidelines

This section summarizes the main features of the design process from the previous subsection. The goal is to try to provide some basic design guidelines that are generic to all fuzzy controllers. In this spirit, we list some basic design guidelines for (nonadaptive) fuzzy controllers:

1. Begin by trying a simple conventional PID controller. If this is successful, do not even try a fuzzy controller. The PID is computationally simpler and very easy to understand.

2. Perhaps you should also try some other conventional control approaches (e.g., a lead-lag compensator or state feedback) if it seems that these may offer a good solution.

3. For a variety of reasons, you may choose to try a fuzzy controller (for a discussion of these reasons, see Chapter 1). Be careful to choose the proper inputs to the fuzzy controller. Carefully assess whether you need proportional, integral, and derivative inputs (using standard control engineering ideas). Consider processing plant data into a form that you believe would be most useful for you to control the system if you were actually a "human-in-the-loop." Specify your best guess at as simple a fuzzy controller as possible (do not add inputs, rules, or membership functions until you know you need them).

4. Try tuning the fuzzy controller using the scaling gains, as we discussed in the previous section.

5. Try adding or modifying rules and membership functions so that you more accurately characterize the best way to control the plant (this can sometimes require significant insight into the physics of the plant).

6. Try to incorporate higher-level ideas about how best to control the plant. For instance, try to shape the nonlinear control surface using a nonlinear function of the linguistic-numeric values, as explained in the previous section.

7. If there is unsmooth or chattering behavior, you may have a gain set too high on an input to the fuzzy controller (or perhaps the output gain is too high). Setting the input gain too high makes it so that the membership functions saturate for very low values, which can result in oscillations (i.e., limit cycles).

8. Sometimes the addition of more membership functions and rules can help. These can provide for a "finer" (or "higher-granularity") control, which can sometimes reduce chattering or oscillations.

9. Sometimes it is best to first design a linear controller, then choose the scaling gains, membership functions, and rule-base so that near the origin (i.e., for small controller inputs) the slope of the control surface will match the slope of the linear controller. In this way we can incorporate all of the good ideas that have gone into the design of the linear controller (about an operating point) into the design of the fuzzy controller. After this, the designer should seek to shape the nonlinearity for the case where the input signals are not near the origin using insights about the plant. This design approach will be illustrated in Chapter 3 when we investigate case studies in fuzzy control system design.

Generally, you do *not* tune the fuzzy controller by evaluating all possibilities for representing the "and" in the premise or for the implication (e.g., minimum or product operations) or by studying different defuzzification strategies. While there are some methods for tuning fuzzy controllers this way, these methods most often do not provide insights into how these parameters ultimately affect the performance that we are trying to achieve (hence it is difficult to know how to tune them to get the desired performance). We must emphasize that the above guidelines do not constitute a systematic design procedure. As with conventional control design, a process of trial and error is generally needed.

Generally, we have found that if you are having trouble coming up with a good fuzzy controller, you probably need to gain a better understanding of the physics of the process you are trying to control, and you then need to get the knowledge of how to properly affect the plant dynamics into the fuzzy controller.

2.5 Simulation of Fuzzy Control Systems

Often, before you implement a fuzzy controller, there is a need to perform a simulation-based evaluation of its performance. As we saw in Section 2.4, where we studied the inverted pendulum, these simulation-based investigations can help to provide insight into how to improve the design of the fuzzy controller and verify that it will operate properly when it is implemented. To perform a simulation, we will need a model of the plant and a computer program that will simulate the fuzzy control system (i.e., a program to simulate a nonlinear dynamic system).

2.5.1 Simulation of Nonlinear Systems

In the next subsection we will explain how to write a subroutine that will simulate a fuzzy controller. First, however, we will briefly explain how to simulate a nonlinear system since every fuzzy control system is a nonlinear system (even if the plant is linear, the fuzzy controller and hence fuzzy control system is nonlinear). Suppose that we denote the fuzzy controller in Figure 2.4 on page 27 by $f(e, \dot{e})$. Suppose that the fuzzy control system in Figure 2.4 can be represented by the ordinary differential equation

$$\begin{aligned} \dot{x}(t) &= F(x(t), r(t)) \\ y &= G(x(t), r(t)) \end{aligned} \tag{2.26}$$

where $x = [x_1, x_2, \ldots, x_n]^\top$ is a state vector, $F = [F_1, F_2, \ldots, F_n]^\top$ is a vector of nonlinear functions, G is a nonlinear function that maps the states and reference input to the output of the system, and $x(0)$ is the initial state. To simulate a nonlinear system, we will assume that the nonlinear ordinary differential equations are put into the form in Equation (2.26).

To see how to put a given ordinary differential equation into the form given in Equation (2.26), consider the inverted pendulum example. For our pendulum example, define the state

$$x = [x_1, x_2, x_3]^\top = [y, \dot{y}, \bar{u}]^\top$$

Then, using Equation (2.25) on page 78 we have

$$\begin{aligned} \dot{x}_1 &= x_2 = F_1(x, r) \\ \dot{x}_2 &= \frac{9.8 \sin(x_1) + \cos(x_1) \left[\frac{-x_3 - 0.25 x_2^2 \sin(x_1)}{1.5} \right]}{0.5 \left[\frac{4}{3} - \frac{1}{3} \cos^2(x_1) \right]} = F_2(x, r) \\ \dot{x}_3 &= -100 x_3 + 100 f(-x_1, -x_2) = F_3(x, r) \end{aligned}$$

since $u = f(e, \dot{e})$, $e = r - y$, $r = 0$, and $\dot{e} = -\dot{y}$. Also, we have $y = G(x, r) = x_1$. This puts the fuzzy control system for the nonlinear inverted pendulum in the proper form for simulation.

Now, to simulate Equation (2.26), we could simply use Euler's method to approximate the derivative $\dot{x}$ in Equation (2.26) as

$$\frac{x(kh+h)-x(kh)}{h} = F(x(kh), r(kh), kh) \qquad (2.27)$$
$$y = G(x(kh), r(kh), kh)$$

Here, h is a parameter that is referred to as the "integration step size" (not to be confused with the scaling gain h used earlier). Notice that any element of the vector

$$\frac{x(kh+h)-x(kh)}{h}$$

is simply an approximation of the slope of the corresponding element in the time varying vector $x(t)$ at $t = kh$ (i.e., an approximation of the derivative). For small values of h, the approximation will be accurate provided that all the functions and variables are continuous. Equation (2.27) can be rewritten as

$$x(kh+h) = x(kh) + hF(x(kh), r(kh), kh)$$
$$y = G(x(kh), r(kh), kh)$$

for $k = 0, 1, 2, \ldots$. The value of the vector $x(0)$ is the initial condition and is assumed to be given. Simulation of the nonlinear system proceeds recursively by computing $x(h)$, $x(2h)$, $x(3h)$, and so on, to generate the response of the system for the input $r(kh)$. For practice the reader should place the pendulum differential equations developed above into the form for simulation via the Euler method given in Equation (2.27). Using this, and provided that you pick your integration step size h small enough, the Euler method can be used to reproduce all the simulation results of the previous section.

Note that by choosing h small, we are trying to simulate the *continuous-time* nonlinear system. If we want to simulate the way that a digital control system would be implemented on a computer in the laboratory, we can simulate a controller that only samples its inputs every T seconds (T is not the same as h; it is the "sampling interval" for the computer in the laboratory) and only updates its control outputs every T seconds (and it would hold them constant in between). Normally, you would choose $T = \alpha h$ where $\alpha > 0$ is some positive integer. In this way we simulate the plant as a continuous-time system that interacts with a controller that is implemented on a digital computer.

While Euler's method is easy to understand and implement in code, sometimes to get good accuracy the value of h must be chosen to be very small. Most often, to get good simulation accuracy, more sophisticated methods are used, such as the Runge-Kutta method with adaptive step size or predictor-corrector methods. In the fourth-order Runge-Kutta method, we begin with Equation (2.26) and a given

$x(0)$ and let

$$x(kh+h) = x(kh) + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{2.28}$$

where the four vectors

$$\begin{aligned}
k_1 &= hF(x(kh), r(kh), kh) \\
k_2 &= hF\left(x(kh) + \frac{k_1}{2}, r\left(kh + \frac{h}{2}\right), kh + \frac{h}{2}\right) \\
k_3 &= hF\left(x(kh) + \frac{k_2}{2}, r\left(kh + \frac{h}{2}\right), kh + \frac{h}{2}\right) \\
k_4 &= hF\left(x(kh) + k_3, r(kh+h), kh + h\right)
\end{aligned}$$

These extra calculations are used to achieve a better accuracy than the Euler method. We see that the Runge-Kutta method is very easy to use; it simply involves computing the four vectors k_1 to k_4, and plugging them into Equation (2.28). Suppose that you write a computer subroutine to compute the output of a fuzzy controller given its inputs (in some cases these inputs could include a state of the closed-loop system). In this case, to calculate the four vectors, k_1 to k_4, you will need to use the subroutine four times, once for the calculation of each of the vectors, and this can increase the computational complexity of the simulation. To simplify the complexity of the simulation you can simulate the fuzzy controller as if it were implemented on a digital computer in the laboratory with a sampling interval of $T = h$ (i.e., $\alpha = 1$ in our discussion above). Now, you may not be concerned with implementation of the fuzzy controller on a digital computer in the laboratory, or your choice of h may not actually correspond to a reasonable choice of a sampling period in the laboratory; however, using this approach you typically can simplify computations. The savings come from assuming that over the length of time corresponding to an integration step size, you hold the value of the fuzzy controller output constant. Hence, this approach to simplifying computations is really simply based on making an approximation to the fuzzy controller output over the amount of time corresponding to an integration step size.

Generally, if the Runge-Kutta method has a small enough value of h, it is sufficiently accurate for the simulation of most fuzzy control systems (and if an adaptive step size method [59, 215] is used, then even more accuracy can be obtained if it is needed). We invite the reader to code the Runge-Kutta method on the problems at the end of the chapter.[10]

Clearly, the above approaches are complete only if we can compute the fuzzy controller outputs given its inputs. That is, we need a subroutine to compute $u = f(e, \dot{e})$. This is what we study next.

10. The reader can, however, download the code for a Runge-Kutta algorithm for simulating an n^{th} order nonlinear ordinary differential equation from the web site or ftp site listed in the Preface.

2.5.2 Fuzzy Controller Arrays and Subroutines

The fuzzy controller can be programmed in C, Fortran, Matlab, or virtually any other programming language. There may be some advantage to programming it in C since it is then sometimes easier to transfer the code directly to an experimental setting for use in real-time control. At other times it may be advantageous to program it in Matlab since plotting capabilities and other control computations may be easier to perform there. Here, rather than discussing the syntax and characteristics of the multitude of languages that we could use to simulate the fuzzy controller, we will develop a computer program "pseudocode" that will be useful in developing the computer program in virtually any language. For readers who are not interested in learning how to write a program to simulate the fuzzy controller, this section will provide a nice overview of the steps used by the fuzzy controller to compute its outputs given some inputs.

We will use the inverted pendulum example of the last section to illustrate the basic concepts on how to program the fuzzy controller, and for that example we will use the minimum operation to represent both the "and" in the premise and the implication (it will be obvious how to switch to using, for example, the product). At first we will make no attempt to code the fuzzy controller so that it will minimize execution time or minimize the use of memory. However, after introducing the pseudocode, we will address these issues.

First, suppose that for convenience we use a different set of linguistic-numeric descriptions for the input and output membership functions than we used up till now. Rather than numbering them

$$-2, -1, 0, 1, 2$$

we will renumber them as

$$0, 1, 2, 3, 4$$

so that we can use these as indices for arrays in the program (if your language does not allow for the use of "0" as an index, simply renumber them as $1, 2, 3, 4, 5$). Suppose that we let the computer variable `x1` denote (notice that a different typeface is used for all computer variables) $e(t)$, which we will call the first input, and `x2` denote $\frac{d}{dt}e(t)$, which we will call the second input. Next, we define the following arrays and functions:

- Let `mf1[i]` (`mf2[j]`) denote the value of the membership function associated with input 1 (2) and linguistic-numeric value `i` (`j`). In the computer program `mf1[i]` could be a subroutine that computes the membership value for the i^{th} membership function given a numeric value for the first input `x1` (note that in the subroutine we can use simple equations for lines to represent triangular membership functions). Similarly for `mf2[j]`.
- Let `rule[i,j]` denote the index of the consequent of the rule that has linguistic-numeric value "`i`" as the first term in its premise and "`j`" as the second term in

its premise. Hence `rule[i,j]` is essentially a matrix that holds the body of the rule-base table shown in Table 2.1 with the appropriate changes to the linguistic-numeric values (i.e., switching from the use of $-2, -1, 0, 1, 2$ to $0, 1, 2, 3, 4$). In particular, for the inverted pendulum we have

$$\texttt{rule[i, j]} = \begin{bmatrix} 4 & 4 & 4 & 3 & 2 \\ 4 & 4 & 3 & 2 & 1 \\ 4 & 3 & 2 & 1 & 0 \\ 3 & 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \end{bmatrix}$$

- Let `prem[i,j]` denote the certainty of the premise of the rule that has linguistic-numeric value "`i`" as the first term in its premise and "`j`" as the second term in its premise given the inputs `x1` and `x2`.
- Let `center[k]` denote the center of the k^{th} output membership function. For the inverted pendulum $k = 0, 1, 2, 3, 4$ and the centers are at the points where the triangles reach their peak.
- Let `areaimp[k,h]` denote the area under the k^{th} output membership function (where for the inverted pendulum $k = 0, 1, 2, 3, 4$) that has been chopped off at a height of `h` by the minimum operator. Hence, we can think of `areaimp[k,h]` as a subroutine that is used to compute areas under the membership functions for the implied fuzzy sets.
- Let `imps[i,j]` denote the areas under the membership functions for the implied fuzzy sets for the rule that has linguistic-numeric value "`i`" as the first term in its premise, and "`j`" as the second term in its premise given the inputs `x1` and `x2`.

2.5.3 Fuzzy Controller Pseudocode

Using these definitions, consider the pseudocode for a simple fuzzy controller that is used to compute the fuzzy controller output given its two inputs:

```
1. Obtain x1 and x2 values
   (Get inputs to fuzzy controller)

2. Compute mf1[i] and mf2[j] for all i, j
   (Find the values of all membership functions given the values for
   x1 and x2)

3. Compute prem[i,j]=min[mf1[i],mf2[j]] for all i, j
   (Find the values for the premise membership functions for a given
   x1 and x2 using the minimum operation)
```

```
4. Compute imps[i,j]=areaimp[rule[i,j],prem[i,j]] for all i, j
   (Find the areas under the membership functions for all
   possible implied fuzzy sets)

5. Let num=0, den=0
   (Initialize the COG numerator and denominator values)

6. For i=0 to 4, For j=0 to 4,
   (Cycle through all areas to determine COG)

    num=num+imps[i,j]*center[rule[i,j]]
       (Compute numerator for COG)

    den=den+imps[i,j]
       (Compute denominator for COG)

7. Next i, Next j

8. Output ucrisp=num/den
   (Output the value computed by the fuzzy controller)

9. Go to Step 1.
```

To learn how this code operates, the reader should define each of the functions and arrays for the inverted pendulum example and show how to compute the fuzzy controller output for the same (and some different) inputs used in Section 2.4. Following this, the reader should develop the computer code to simulate the fuzzy controller for the inverted pendulum and verify that the computations made by the computer match the ones made by hand.[11]

We do not normally recommend that initially you use only the computer-aided design (CAD) packages for fuzzy systems since these tend to remove you from understanding the real details behind the operation of the fuzzy controller. However, after you have developed your own code and fully understand the details of fuzzy control, we do advise that you use (or develop) the tools you believe are necessary to automate the process of constructing fuzzy controllers.

Aside from the effort that you must put into writing the code for the fuzzy controller, there are the additional efforts that you must take to initially type in the rule-base and membership functions and possibly modify them later (which might be necessary if you need to perform redesigns of the fuzzy controller). For large rule-bases, this effort could be considerable, especially for initially typing the rule-base into the computer. While some CAD packages may help solve this problem, it is not hard to write a computer program to generate the rule-base, because there are often certain regular patterns in the rule-base. For example, a very common pattern found in rule-bases is the "diagonal" one shown in Table 2.1 on page 32. Here, the linguistic-numeric indices in the row at the top and the column on the

11. One way to start with the coding of the fuzzy controller is to start with the code that is available for downloading at the web site or ftp site described in the Preface.

left are simply added, and the sum is multiplied by minus one and saturated so that it does not grow beyond the available indices for the consequent membership functions (i.e., below -2 or above 2).

Also notice that since there is a proportional correspondence between the input linguistic-numeric values and the values of the inputs, you will often find it easy to express the input membership functions as a nonlinear function of their linguistic-numeric values. Another trick that is used to make the adjustment of rule-bases easier is to make the centers of the output membership functions a function of their linguistic-numeric indices, as we discussed in Section 2.4.2.

2.6 Real-Time Implementation Issues

When it comes to implementing a fuzzy controller, you often want to try to minimize the amount of memory used and the time that it takes to compute the fuzzy controller outputs given some inputs. The pseudocode in the last section was not written to exploit certain characteristics of the fuzzy controller that we had developed for the inverted pendulum; hence, if we were to actually implement this fuzzy controller and we had severe implementation constraints, we could try to optimize the code with respect to memory and computation time.

2.6.1 Computation Time

First, we will focus on reducing the amount of time it takes to compute the outputs for some given inputs. Notice the following about the pseudocode:

- We compute `prem[i,j]` for all values of `i` and `j` (25 values) when for our fuzzy controller for the inverted pendulum, since there are never more than two membership functions overlapping, there will be at most four values of `prem[i,j]` needed (the rest will have zero values and hence will have no impact on the ultimate computation of the output).
- In a similar manner, while we compute `imps[i,j]` for all `i` and `j`, we only need four of these values.
- If we compute only four values for `imps[i,j]`, we will have at most four values to sum up in the numerator and denominator of the COG computation (and not 25 for each).

At this point, from the view of computational complexity, the reader may wonder why we even bothered with the pseudocode of the last section since it appears to be so inefficient. However, the code is only inefficient for the chosen form for the fuzzy controller. If we had chosen Gaussian-shaped (i.e., bell-shaped) membership functions for the input membership functions, then no matter what the input was to the fuzzy controller, all the rules would be on so all the computations shown in the pseudocode were necessary and not too much could be done to improve on the computation time needed. Hence, we see that if you are concerned with real-time

implementation of your fuzzy controller, you may want to put constraints on the type of fuzzy controller (e.g., membership functions) you construct.

It is important to note that the problems with the efficiency of the pseudocode highlighted above become particularly acute when there are many inputs to the fuzzy controller and many membership functions for each input, since the number of rules increases exponentially with an increase in the number of inputs (assuming all possible rules are used, which is often the case). For example, if you have a two-input fuzzy controller with 11 membership functions for each input, you will have $11^2 = 121$ rules, and you can see that if you increase the number of inputs, this number will quickly increase.

How do we overcome this problem? Assume that you have defined your fuzzy controller so that at most two input membership functions overlap at any one point, as we had for the inverted pendulum example. The trick is to modify your code so that it will compute only four values for the premise membership functions, only four values for areas of implied fuzzy sets, and hence have only four additions in the numerator and denominator of the COG computation. There are many ways to do this. For instance, you can have the program scan `mf1[i]` beginning at position zero until a nonzero membership value is obtained. Call the index of the first nonzero membership value "`istar`." Repeat this process for `mf2[j]` to find a corresponding "`jstar`." The rules that are on are the following:

```
rule[istar,jstar]
rule[istar,jstar+1]
rule[istar+1,jstar]
rule[istar+1,jstar+1]
```

provided that the indicated indices are not out of range. If only the rules identified by the indices of the premises of these rules are used in the computations, then we will reduce the number of required computations significantly, because we will not be computing values that will be zero anyway (notice that for the inverted pendulum example, there will be one, two, or four rules on at any one time, so there could still be a few wasted computations). Notice that even in the case where there are many inputs to the fuzzy controller *the problem of how to code efficiently reduces to a problem of how to determine the set of indices for the rules that are on.* So that you may fully understand the issues in coding the controller in an efficient manner, we challenge you to develop the code for an n-input fuzzy controller that will exploit the fact that only a hypercubical block of 2^n rules will be on at any one time (provided that at most two input membership functions overlap at any one point).

2.6.2 Memory Requirements

Next, we consider methods for reducing memory requirements. Basically, this can be done by recognizing that it may be possible to compute the rule-base at each time instant rather than using a stored one. Notice that there is a regular pattern to the rule-base for the inverted pendulum; since there are at most four rules on

at any one time, it would not be hard to write the code so that it would actually generate the rules while it computes the controller outputs. It may also be possible to use a memory-saving scheme for the output membership functions. Rather than storing their positions, there may be a way to specify their spacing with a function so that it can be computed in real-time. For large rule-bases, these approaches can bring a huge savings in memory (however, if you are working with adaptive fuzzy systems where you automatically tune membership functions, then it may not be possible to use this memory-saving scheme). We are, however, gaining this savings in memory at the expense of possibly increasing computation time.

Finally, note that while we focus here on the real-time implementation issues by discussing the optimization of *software*, you could consider redesigning the *hardware* to make real-time implementation possible. Implementation prospects could improve by using a better microprocessor or signal processing chip. An alternative would be to investigate the advantages and disadvantages of using a "fuzzy processor" (i.e., a processor designed specifically for implementing fuzzy controllers). Of course, many additional issues must be taken into consideration when trying to decide if a switch in computing technology is needed. Not the least among these are cost, durability, and reliability.

2.7 Summary

In this chapter we have provided a "tutorial introduction" to direct fuzzy control. In our tutorial introduction we provided a step-by-step overview of the operations of the fuzzy controller. We provided an inverted pendulum problem for which we discussed several basic issues in the design of fuzzy controllers. Moreover, we discussed simulation and implementation via the use of a pseudocode for a fuzzy controller. Our introduction is designed to provide the reader with an intuitive understanding of the mechanics of the operation of the fuzzy controller.

Our mathematical characterization served to show how the fuzzy controller can handle more inputs and outputs, the range of possibilities for the definition of universes of discourse, the membership functions, the rules, the inference mechanism, and defuzzification methods. The reader who studies the mathematical characterization of fuzzy systems will gain a deeper understanding of fuzzy systems.

The design example for the inverted pendulum problem is meant to be an introduction to basic design methods for fuzzy controllers. The section on coding is meant to help the reader bridge the gap between theory and application so that you can quickly get a fuzzy controller implemented.

Upon completing this chapter, the reader should understand the following topics:

- Issues in the choice of the inputs and outputs of the fuzzy controller.
- Linguistic variables.
- Linguistic values (and linguistic-numeric values).

- Linguistic rules (MISO, MIMO, and ones that do not use all their premise or consequent terms).
- Membership functions (in terms of how they quantify linguistics and their mathematical definition).
- Fuzzy sets (mathematical definition and relation to crisp sets).
- Operations on fuzzy sets (subset, complement, union, intersection, and relations to representation of the logical "and" and "or").
- Fuzzy Cartesian product and its use in representation of the premise.
- The multidimensional premise membership function that represents the conjunction of terms in the premise.
- Fuzzification (singleton and more general forms).
- Inference mechanism (three stages: matching, selection of rules that are on, and taking the actions specified by the applicable rules).
- Implied fuzzy sets.
- Overall implied fuzzy sets (and the differences from the implied fuzzy sets).
- Sup-star and Zadeh's compositional rule of inference.
- Defuzzification methods (including those for the implied fuzzy sets and overall implied fuzzy set).
- The method of providing a graphical explanation of the inference process that was given at the end of Section 2.2.1.
- Mathematical representations of fuzzy systems, including issues related to the number of parameters needed to represent a fuzzy system.
- Functional fuzzy systems (and Takagi-Sugeno fuzzy systems).
- The universal approximation property and its implications.
- Basic approaches to the design of the fuzzy controller, including the use of proportional, integral, and derivative terms.
- The value of getting the best knowledge about how to achieve good control into the rule-base and methods for doing this (e.g., the use of functions mapping the linguistic-numeric indices to the centers of the output membership functions).
- The manner in which a fuzzy controller implements a nonlinearity and connections between the choice of controller parameters (e.g., scaling gains) and the shape of this nonlinearity.

- How to simulate a nonlinear system.
- How to simulate a fuzzy system and a fuzzy control system.
- Methods to optimize the code that implements a fuzzy controller (with respect to both time and memory).

Essentially, this is a checklist for the major topics of this chapter. The reader should be sure to understand each of the above concepts or approaches before moving on to later chapters.

2.8 For Further Study

There are many conferences and journals that cover issues in fuzzy systems and control. Some journals to consider include the following: (1) *IEEE Transactions on Fuzzy Systems*, (2) *IEEE Control Systems Magazine*, (3) *IEEE Transactions on Systems, Man, and Cybernetics*, and (4) *Fuzzy Sets and Systems*. The field of fuzzy sets and logic was first introduced by Lotfi Zadeh [245, 246], and fuzzy control was first introduced by E. Mamdani [135, 134]. There are many books on the mathematics of fuzzy sets, fuzzy logic, and fuzzy systems; a few that the reader may want to study include [95, 94, 250, 48, 87] or the article [138]. There are also several books that provide introductions to the area of fuzzy control [47, 230, 238, 229, 167]. Other sources for introductory material on fuzzy control are in [165, 115].

An early version of the mathematical introduction to fuzzy control given in this chapter is given in [107, 110] and a more developed one, that was the precursor to Section 2.3 is in [165]. While in most applications singleton fuzzification is used, there have been some successful uses of nonsingleton fuzzification [146]. For more details on how to simulate nonlinear systems, see [59, 215]. The ball-suspension system problem at the end of the chapter was taken from [103]. The automated highway system problem was taken from [200].

2.9 Exercises

Exercise 2.1 (Defining Membership Functions: Single Universe of Discourse): In this problem you will study how to represent various concepts and quantify various relations with membership functions. For each part below, there is more than one correct answer. Provide one of these and justify your choice in each case.

(a) Draw a membership function (and hence define a fuzzy set) that quantifies the set of all people of medium height.

(b) Draw a membership function that quantifies the set of all short people.

(c) Draw a membership function that quantifies the set of all tall people.

(d) Draw a membership function that quantifies the statement "the number x is near 10."

(e) Draw a membership function that quantifies the statement "the number x is less than 10."

(f) Draw a membership function that quantifies the statement "the number x is greater than 10."

(g) Repeat (d)–(f) for -5 rather than 10.

Exercise 2.2 (Defining Membership Functions: Multiple Universes of Discourse): In this problem you will study how to represent various concepts and quantify various relations with membership functions when there is more than one universe of discourse. Use minimum to quantify the "and." For each part below, there is more than one correct answer. Provide one of these and justify your choice in each case. Also, in each case draw the three-dimensional plot of the membership function.

(a) Draw a membership function (and hence define a fuzzy set) that quantifies the set of all people of medium height who are "tan" in color (i.e., tan *and* medium-height people). Think of peoples' colors being on a spectrum from white to black.

(b) Draw a membership function (and hence define a fuzzy set) that quantifies the set of all short people who are "white" in color (i.e., short *and* white people).

(c) Draw a membership function (and hence define a fuzzy set) that quantifies the set of all tall people who are "black" in color (i.e., tall *and* black people).

(d) Draw a membership function that quantifies the statement "the number x is near 10 and the number y is near 2."

(e) Draw a membership function that quantifies the statement "the number x is less than 10 and the number y is near 2."

(f) Draw a membership function that quantifies the statement "the number x is greater than 10 and the number y is near 2."

(g) Repeat (d)–(f) for -5 rather than 10 and -1 rather than 2.

(h) Repeat (d)–(f) using product rather than minimum to represent the "and."

Exercise 2.3 (Inverted Pendulum: Gaussian Membership Functions): Suppose that for the inverted pendulum example, we use Gaussian membership functions as defined in Table 2.4 on page 57 rather than the triangular membership functions. To do this, use the same center values as we had for the triangular membership functions, use the "left" and "right" membership functions shown in Table 2.4 for the outer edges of the input universes of discourse, and choose the widths of all the membership functions to get a uniform distribution of the membership functions and to get adjacent membership functions to cross over with their neighboring membership functions at a certainty of 0.5.

(a) Draw the membership functions for the input and output universes of discourse. Be sure to label all the axes and include both the linguistic values and the linguistic-numeric values. Explain why this choice of membership functions also properly represents the linguistic values.

(b) Assuming that we use the same rules as earlier, use a computer program to plot the membership function for the premise of a rule when you use the minimum operation to represent the "and" between the two elements in the premise. For this plot you will have e and $\frac{d}{dt}e$ on the x and y axes and the value of the premise membership function on the z axis. Use the rule

If error is zero **and** change-in-error is possmall **Then** force is negsmall

as was done when we used triangular membership functions (see its premise membership function in Figure 2.11 on page 39).

(c) Repeat (b) for the case where the product operation is used. Compare the results of (b) and (c).

(d) Suppose that $e(t) = 0$ and $\frac{d}{dt}e(t) = \pi/8 - \pi/32$ $(= 0.294)$. Which rules are on? Assume that minimum is used to represent the premise and implication. Provide a plot of the implied fuzzy sets for the two rules that result in the highest peak on their implied fuzzy sets (i.e., the two rules that are "on" the most).

(e) Repeat (d) for the case where $e(t) = \pi/4$ and $\frac{d}{dt}e(t) = \pi/8$. Assume that the product is used to represent the implication and minimum is used for the premise. However, plot only the one implied fuzzy set that reaches the highest value.

(f) For (d) use COG defuzzification and find the output of the fuzzy controller. First, compute the output assuming that only the two rules found in (d) are on. Next, use the implied fuzzy sets from all the rules that are on (note that more than two rules are on). Note that for computation of the area under a Gaussian curve, you will need to write a simple numerical integration routine (e.g., based on a trapezoidal approximation) since there is no closed-form solution for the area under a Gaussian curve.

(g) Repeat (f) for the case in (e).

(h) Assume that the minimum operation is used to represent the premise and implication. Plot the control surface for the fuzzy controller.

(i) Repeat (h) for the case where the product operation is used for the premise and implication. Compare (h) and (i).

Exercise 2.4 (Inverted Pendulum: Rule-Base Modifications): In this problem we will study the effects of adding rules to the rule-base. Suppose that we use seven triangular membership functions on each universe of discourse and make them uniformly distributed in the same manner as how we did in Exercise 2.3. In particular, make the points at which the outermost input membership functions

for e saturate at $\pm\frac{\pi}{2}$ and for $\dot{e}$ at $\pm\frac{\pi}{4}$. For u make the outermost ones have their peaks at ± 20.

(a) Define a rule-base (i.e., membership functions and rules) that uses all possible rules, and provide a rule-base table to list all of the rules (make an appropriate choice of the linguistic-numeric values for the premise terms and consequents). There should be 49 rules.

(b) Use triangular membership functions and repeat Exercise 2.3 (a), (b), (c), (d), (e) (but provide the implied fuzzy sets for the four rules that are on), (f), (g) (but use all four implied fuzzy sets in the COG computation), (h), and (i).

Exercise 2.5 (Fuzzy Sets): There are many concepts that are used in fuzzy sets that sometimes become useful when studying fuzzy control. The following problems introduce some of the more popular fuzzy set concepts that were not treated earlier in the chapter.

(a) The "support" of a fuzzy set with membership function $\mu(x)$ is the (crisp) set of all points x on the universe of discourse such that $\mu(x) > 0$ and the "α-cut" is the (crisp) set of all points on the universe of discourse such that $\mu(x) > \alpha$. What is the support and 0.5-cut for the fuzzy set shown in Figure 2.6 on page 33?

(b) The "height" of a fuzzy set with membership function $\mu(x)$ is the highest value that $\mu(x)$ reaches on the universe of discourse on which it is defined. A fuzzy set is said to be "normal" if its height is equal to one. What is the height of the fuzzy set shown in Figure 2.6 on page 33? Is it normal? Give an example of a fuzzy set that is not normal.

(c) A fuzzy set with membership function $\mu(x)$ where the universe of discourse is the set of real numbers is said to be "convex" if and only if

$$\mu(\lambda x_1 + (1-\lambda)x_2) \geq \min\{\mu(x_1), \mu(x_2)\} \tag{2.29}$$

for all x_1 and x_2 and all $\lambda \in [0,1]$. Note that just because a fuzzy set is said to be convex does not mean that its membership function is a convex function in the usual sense. Prove that the fuzzy set shown in Figure 2.6 on page 33 is convex. Prove that the Gaussian membership function is not convex. Give an example (besides the fuzzy set with a Gaussian membership function) of a fuzzy set that is not convex.

(d) A linguistic "hedge" is a modifier to a linguistic value such as "very" or "more or less." When we use linguistic hedges for linguistic values that already have membership functions, we can simply modify these membership functions so that they represent the modified linguistic values. Consider the membership function in Figure 2.6 on page 33. Suppose that we obtain the membership function for "error is very possmall" from the one for "possmall" by squaring the membership values (i.e., $\mu_{verypossmall} = (\mu_{possmall})^2$).

Sketch the membership function for "error is very possmall." For "error is more or less possmall" we could use $\mu_{moreorlesspossmall} = \sqrt{\mu_{possmall}}$. Sketch the membership function for "error is more or less possmall."

Exercise 2.6 (The Extension Principle): A method for fuzzifying crisp functions is called an "extension principle." If X is a universe of discourse, let X^* denote the "fuzzy power set" of X, which is the set of all fuzzy sets that can be defined on X (since there are many ways to define membership functions, X^* is normally a large set—e.g., if X is the set of real numbers, then there is a continuum number of elements in X^*). Suppose that X and Y are two sets. The "extension principle" states that any function

$$f : X \to Y$$

induces two functions,

$$f : X^* \to Y^*$$

and

$$f^{-1} : Y^* \to X^*$$

which are defined by

$$[f(A)](y) = \sup_{\{x : y = f(x)\}} \mu_A(x)$$

for all fuzzy sets A defined on X^* that have membership functions denoted by $\mu_A(x)$ (we use $[f(A)](y)$ to denote the membership function produced by the mapping f and defined on the range of f) and

$$[f^{-1}(B)](x) = \mu_B(f(x))$$

for all fuzzy sets B defined on Y^* that have membership functions denoted by $\mu_B(x)$ (we use $[f^{-1}(B)](x)$ to denote the membership function produced by the mapping f^{-1} and defined on the domain of f).

(a) Suppose that $X = [0, \infty)$, $Y = [0, \infty)$ and $y = f(x) = x^3$. Find $[f(A)](y)$.

(b) Repeat (a) for $y = f(x) = x^2$.

Exercise 2.7 (Fuzzy Logic): There are many concepts that are used in fuzzy logic that sometimes become useful when studying fuzzy control. The following problems introduce some of the more popular fuzzy logic concepts that were not treated earlier in the chapter or were treated only briefly.

(a) The complement ("not") of a fuzzy set with a membership function μ has a membership function given by $\bar{\mu}(x) = 1 - \mu(x)$. Sketch the complement of the fuzzy set shown in Figure 2.6 on page 33.

(b) There are other ways to define the "triangular norm" for representing the intersection operation ("and") on fuzzy sets, different from the ones introduced in the chapter. Two more are given by defining "$*$" as a "bounded difference" (i.e., $x * y = \max\{0, x + y - 1\}$) and "drastic intersection" (where $x * y$ is x when $y = 1$, y when $x = 1$, and zero otherwise). Consider the membership functions shown in Figure 2.9 on page 36. Sketch the membership function for the premise "error is zero **and** change-in-error is possmall" when the bounded difference is used to represent this conjunction (premise). Do the same for the case when we use the drastic intersection. Compare these to the case where the minimum operation and the product were used (i.e., plot these also and compare all four).

(c) There are other ways to define the "triangular co-norm" for representing the union operation ("or") on fuzzy sets, different from the ones introduced in the chapter. Two more are given by defining "$\oplus$" as a "bounded sum" (i.e., $x \oplus y = \min\{1, x + y\}$) and "drastic union" (where $x \oplus y$ is x when $y = 0$, y when $x = 0$, and one otherwise). Consider the membership functions shown in Figure 2.9 on page 36. Sketch the membership function for "error is zero **or** change-in-error is possmall" when the bounded sum is used. Do the same for the case when we use the drastic union. Compare these to the case where the maximum operation and the algebraic sum were used (i.e., plot these also and compare all four).

Exercise 2.8 (Rule-Base Completeness and Consistency): A system of logic is "complete" if everything that is true that can be derived can in fact be derived. It is "consistent" if only true things can be derived according to the system of logic. We consider a rule-base to be "complete" if for every possible combination of inputs to the fuzzy system, the fuzzy system can infer a response and generate an output. We consider it to be consistent if there are no rules that have the same premise and different consequents.

(a) Is the rule-base for the inverted pendulum example shown in Table 2.1 on page 32 with membership functions shown in Figure 2.9 on page 36 complete? Consistent?

(b) Suppose that any one rule is removed from the rule-base shown in Table 2.1 on page 32. Is it still complete and consistent? If it *is* complete and consistent, explain why. If it is *not*, explain this also. In particular, if it is not complete, provide the values of the fuzzy controller inputs that will result in the fuzzy controller failing to provide an output for the rule that you choose to omit. Also, provide the rule that you choose to omit.

(c) Suppose that you replace the triangular membership functions in the inverted pendulum problem with Gaussian ones, as explained in Exercise 2.3. Repeat parts (a) and (b).

(d) Suppose that for the inverted pendulum problem (with triangular membership functions) we remove the membership functions associated with "zero" and "possmall" on the e universe of discourse, which are shown in Figure 2.9 on page 36, and all rules that use these two membership functions in their premises. Show the resulting rule-base table. Is the resulting rule-base complete and consistent? Explain why.

(e) Suppose you designed a slightly different pattern of consequent linguistic-numeric values than those shown in Table 2.1 on page 32 (but with the same triangular membership functions and the same number of rules). Furthermore, suppose that we used your rules *and* the rules shown in Table 2.1 in the new fuzzy controller (i.e., a rule-base that has twice as many rules, with many of the rules you created inconsistent with the ones in Table 2.1). Essentially, this scheme will provide an interpolation between your fuzzy controller design and the one in Table 2.1. Why? Will the fuzzy system still provide a plant input for every possible combination of fuzzy controller inputs?

Exercise 2.9 (Normalized Fuzzy Systems): Sometimes when we use the scaling gains for the inputs and outputs of the fuzzy controller, we refer to the resulting fuzzy system, with the gains, as a "scaled fuzzy system." When a fuzzy system is scaled so that the left-most membership function saturates (peaks) at -1 and the right-most one at $+1$ for both the input and output universes of discourse, we call this a "normalized fuzzy system." Often in computer implementations you will work with a subroutine for a fuzzy system that makes its computations for a normalized fuzzy system, and scaling factors are then used outside the subroutine to obtain appropriately scaled universes of discourse (in this way a single subroutine can be used for many choices of the scaling gains).

(a) For the inverted pendulum problem, what are the scaling factors for the input and output universes of discourse that will achieve normalization of the fuzzy controller? (Use the fuzzy controller that is defined via Table 2.1 on page 32 with membership functions in Figure 2.9 on page 36.)

(b) Given that the fuzzy controller for the inverted pendulum was normalized, what are the scaling gains that should be used to get the universes of discourse shown in Figure 2.9 on page 36?

(c) Suppose that you are given the fuzzy controller that is defined via Table 2.1 on page 32 with membership functions in Figure 2.9 on page 36, but that you would like the universes of discourse to be on a different scale. In particular, you would like the effective universes of discourse to be $[-10, 10]$ for e, $[-5, 5]$ for $\dot{e}$, and $[-2, 2]$ for u. What are the scaling gains that will achieve this?

Exercise 2.10 (Defuzzification): Suppose that for the inverted pendulum we

have

$$e(t) = \frac{\pi}{8}$$

and

$$\frac{d}{dt}e(t) = -\frac{\pi}{8} + \frac{\pi}{32}$$

at some time t. Assume that we use the rule-base shown in Table 2.1 on page 32 and minimum to represent both the premise and implication.

(a) Draw all the implied fuzzy sets on the output universe of discourse.

(b) Draw the overall implied fuzzy set assuming that maximum is used.

(c) Find the output of the fuzzy controller using center-average defuzzification.

(d) Find the output of the fuzzy controller using COG defuzzification.

(e) For the overall implied fuzzy set, find the output of the fuzzy controller using the maximum criterion, the mean of the maximum, and the COA defuzzification techniques.

(f) Assume that we use the product to represent both the premise and implication. Repeat (a)–(e).

(g) Assume that we use the product to represent the premise and minimum to represent the implication. Repeat (a)–(e).

(h) Assume that we use the minimum to represent the premise and product to represent the implication. Repeat (a)–(e).

(i) Suppose that rather than using the membership functions shown in Figure 2.9 on page 36, we make a small change to one membership function on the output universe of discourse. In particular, we take the right-most membership function (i.e., the one for "poslarge") on the output universe of discourse and make it the same *shape* as the right-most one on the e universe of discourse (i.e., to saturate at 20 and remain at a value of one for all values greater than 20). Suppose that the inputs to the fuzzy controller are

$$e(t) = \frac{d}{dt}e(t) = -\frac{\pi}{2}$$

at some time t. Repeat (a)–(e) (use minimum to represent both the premise and implication). Explain any problems that you encounter.

Exercise 2.11 (Graphical Depiction of Fuzzy Decision Making): Develop a graphical depiction of the operation of the fuzzy controller for the inverted pendulum similar to the one given in Figure 2.19 on page 50. For this, choose $e(t) = \frac{3\pi}{8}$ and $\frac{d}{dt}e(t) = \frac{\pi}{16}$, which will result in four rules being on. Be sure to show all parts of the graphical depiction, including an indication of your choices for $e(t)$ and $\frac{d}{dt}e(t)$, the implied fuzzy sets, and the final defuzzified value.

(a) Use minimum for the premise and implication and COG defuzzification.

(b) Use product for the premise and implication and center-average defuzzification.

Exercise 2.12 (Fuzzy Controllers as Interpolators): Fuzzy controllers act as interpolators in the sense that they interpolate between the conclusions that each individual rule of the rule-base reaches. It is possible to derive formulas that show exactly how this interpolation takes place; this is the focus of this problem.

Suppose that you are given a single-input, single-output fuzzy system with input x and output y. Suppose that the input universe of discourse has only two membership functions. The first one is zero from minus infinity to $x = -1$. Then it increases linearly to reach a value of unity when $x = 1$. From $x = 1$ out to plus infinity, the value of the membership function is one. Hence, at $x = 0$ the membership function's value is 0.5. The second membership function is a mirror image of this one about the vertical axis. That is, at minus infinity it starts at one and stays there up till $x = -1$. Then it starts decreasing linearly so that it has a value of zero by $x = 1$. There are only two output membership functions, each of which is a singleton, with one of these centered at $y = -1$ and the other centered at $y = 1$. There are two rules, one that has as a premise the first input membership function and a consequent of the singleton that is centered at $y = -1$, and the other that has as a premise the other input membership function and as a consequent the output membership function centered at $y = 1$. Notice that this fuzzy system is so simple that the input membership functions are the same as the premise membership functions. Use center-average defuzzification.

(a) Sketch the membership functions. Are the computations used to compute the output y for an input x any different if we use symmetric triangular output membership functions centered at ± 1? Why?

(b) Show that for $x \in [-1, 1]$, $y = x$. Show that for $x \in (-\infty, -1]$, $y = -1$. Show that for $x \in [1, +\infty)$, $y = 1$.

This demonstrates that for this case center-average defuzzification performs a linear interpolation between the output centers. Other types of fuzzy systems, such as ones with Gaussian membership functions or COG defuzzification, achieve different types of interpolations that result in different-shaped functions (e.g., see the nonlinear control surface in Figure 2.35 on page 89).

Exercise 2.13 (Takagi-Sugeno Fuzzy Systems): In this problem you will study the way that a Takagi-Sugeno fuzzy system interpolates between linear mappings. Consider in particular the example from Section 2.3.7 where $n = 1$, $R = 2$, and that we had rules

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^1 \textbf{ Then } b_1 = 2 + u_1$$

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^2 \textbf{ Then } b_2 = 1 + u_1$$

with the universe of discourse for u_1 given in Figure 2.24 on page 75 so that μ_1 represents $\tilde{A}_1^1$ and μ_2 represents $\tilde{A}_1^2$. We have $y = b_1\mu_1 + b_2\mu_2$.

(a) Show that the nonlinear mapping induced by this Takagi-Sugeno fuzzy system is given by

$$y = \begin{cases} 1 + u_1 & \text{if } u_1 > 1 \\ 0.5u_1 + 1.5 & \text{if } -1 \leq u_1 \leq 1 \\ 2 + u_1 & u_1 < -1 \end{cases}$$

(Hint: The Takagi-Sugeno fuzzy system represents three lines, two in the consequents of the rules and one that interpolates between these two.)

(b) Plot y versus u_1 over a sufficient range of u_1 to illustrate the nonlinear mapping implemented by the Takagi-Sugeno fuzzy system.

Exercise 2.14 (Fuzzy Controller Simulation): In this problem you will develop a computer program that can simulate a fuzzy controller. You may use the code available at the web site or ftp site listed in the Preface but you must recode it (and add comments to the code) to be able to meet the specifications given in part (a).

(a) Using the approach developed in this chapter, develop a subroutine that will simulate a two-input, one-output fuzzy controller that uses triangular membership functions (except at the outermost edges), either the minimum or the product to represent the "and" in the premise or the implication, and COG or center-average defuzzification.

(b) Use the rule-base from Table 2.1 on page 32 for the inverted pendulum, let $e(t) = \frac{3\pi}{8}$ and $\frac{d}{dt}e(t) = \frac{\pi}{16}$, and find the output of the fuzzy controller.

2.10 Design Problems

Design Problem 2.1 (Inverted Pendulum: Design and Simulation): In this problem you will study the simulation of the fuzzy control system for the inverted pendulum studied in the tutorial introduction to fuzzy control. Use the model defined in Equation (2.25) on page 78 for the model for the pendulum. Be sure to use an appropriate numerical simulation technique for the nonlinear system and a small enough integration step size.

(a) Verify all the simulation results of Section 2.4.1 (i.e., use all the same parameters as used there and reproduce all the simulation results shown).

(b) Repeat (a) for the case where we use Gaussian membership functions, as in Exercise 2.3. Use product to represent the premise and implication and COG defuzzification. This problem demonstrates that changing membership function shapes and the inference strategy can have a significant impact on performance. Once you have completed (a) for all its parts, tune the scaling

gains g_0, g_1, and h to achieve a performance that is at least as good as that shown in Figure 2.25 on page 79.

(c) Repeat (a) for the case where we use 49 rules, as in Exercise 2.4(b) (use triangular membership functions).

(d) Compare the performance obtained in each case. Does switching to the use of Gaussian membership functions and the product improve performance? Why? Does the addition of more rules improve performance? Why?

Design Problem 2.2 (Fuzzy Cruise Control): In this problem you will develop a fuzzy controller that regulates a vehicle's speed $v(t)$ to a driver-specified value $v_d(t)$. The dynamics of the automobile are given by

$$\dot{v}(t) = \frac{1}{m}(-A_\rho v^2(t) - d + f(t))$$
$$\dot{f}(t) = \frac{1}{\tau}(-f(t) + u(t))$$

where u is the control input ($u > 0$ represents a throttle input and $u < 0$ represents a brake input), $m = 1300$ kg is the mass of the vehicle, $A_\rho = 0.3$ Ns^2/m^2 is its aerodynamic drag, $d = 100$ N is a constant frictional force, f is the driving/braking force, and $\tau = 0.2$ sec is the engine/brake time constant. Assume that the input $u \in [-1000, 1000]$ (i.e., that u is saturated at ± 1000 N).

(a) Suppose that we wish to be able to track a step or ramp change in the driver-specified speed value $v_d(t)$ very accurately. Suppose that you choose to use a "PI fuzzy controller" as shown in Figure 2.36. Why does this choice make sense for this problem? In Figure 2.36 the fuzzy controller is denoted by Φ; g_0, g_1, and g_2 are scaling gains; and $b(t)$ is the output of the integrator.

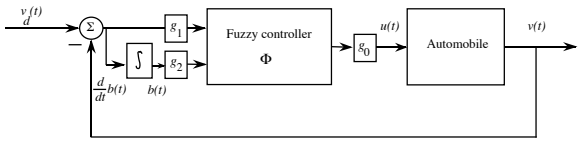


FIGURE 2.36 PI fuzzy cruise controller.

Find the differential equation that describes the closed-loop system. Let the state be $x = [x_1, x_2, x_3]^\top = [v, f, b]^\top$ and find a system of three first-order ordinary differential equations that can be used by the Runge-Kutta method in the simulation of the closed-loop system (i.e., find $F_i(x, v_d)$ for $i = 1, 2, 3$, in Equation (2.26)). Use Φ to represent the fuzzy controller in the differential equations.

For the reference input we will use three different test signals:

1. Test input 1 makes $v_d(t) = 18$ m/sec (40.3 mph) for $0 \leq t \leq 10$ and $v_d(t) = 22$ m/sec (49.2 mph) for $10 \leq t \leq 30$.
2. Test input 2 makes $v_d(t) = 18$ m/sec (40.3 mph) for $0 \leq t \leq 10$ and $v_d(t)$ increases linearly (a ramp) from 18 to 22 m/sec by $t = 25$ sec, and then $v_d(t) = 22$ for $25 \leq t \leq 30$.
3. Test input 3 makes $v_d(t) = 22$ for $t \geq 0$ and we use $x(0) = 0$ as the initial condition (this represents starting the vehicle at rest and suddenly commanding a large increase in speed).

Use $x(0) = [18, 197.2, 20]^\top$ for test inputs 1 and 2. Why is $x(0) = [18, 197.2, 20]^\top$ a reasonable choice for the initial conditions?

Design the fuzzy controller Φ to get less than 2% overshoot, a rise-time between 5 and 7 sec, and a settling time of less than 8 sec (i.e., reach to within 2% of the final value within 8 sec) for the jump from 18 to 22 m/sec in "test input 1" that is defined above. Also, for the ramp input ("test input 2" above) it must have less than 1 mph (0.447 m/sec) steady-state error (i.e., at the end of the ramp part of the input have less than 1 mph error). Fully specify your controller (e.g., the membership functions, rule-base defuzzification, etc.) and simulate the closed-loop system to demonstrate that it performs properly. Provide plots of $v(t)$ and $v_d(t)$ on the same axis and $u(t)$ on a different plot. For test input 3 find the rise-time, overshoot, 2% settling time, and steady-state error for the closed-loop system for the controller that you designed to meet the specifications for test input 1 and 2. In your simulations use the Runge-Kutta method and an integration step size of 0.01.

(b) Next, suppose that you are concerned with tracking a step change in $v_d(t)$ accurately and that you use the PD fuzzy controller shown in Figure 2.37. To represent the derivative, simply use a backward difference

$$c(t) = \frac{e(t) - e(t-h)}{h}$$

where h is the integration step size in your simulation (or it could be your sampling period in an implementation).

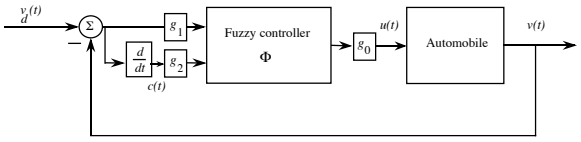


FIGURE 2.37 PD fuzzy cruise controller.

Design a PD fuzzy controller to get less than 2% overshoot, a rise-time between 7 and 10 sec, and a settling time of less than 10 sec for test input 1

defined in (a). Also, for the ramp input (test input 2 in (a)) it must have less than 1 mph steady-state error to the ramp (i.e., at the end of the ramp part of the input, have less than $1mph$ error). Fully specify your controller and simulate the closed-loop system to demonstrate that it performs properly. Provide plots of $v(t)$ and $v_d(t)$ on the same axis and $u(t)$ on a different plot. In your simulations use the Runge-Kutta method and an integration step size of 0.01.

Assume that $x(0) = [18, 197.2]^\top$ for test inputs 1 and 2 (hence we ignore the derivative input in coming up with the state equations for the closed-loop system and simply use the approximation for $c(t)$ that is shown above so that we have a two-state system). As a final test let $x(0) = 0$ and use test input 3 defined in (a). For this, what is the rise-time, overshoot, 2% settling time, and steady state error for your controller?

(c) Explain the effect of the aerodynamic drag term and how you would redesign a rule-base to take this effect into account if you used vehicle velocity directly as an input to the fuzzy controller.

An expanded version of this problem is given in Design Problem 2.4. There, PD controllers are used, and we show how to turn the cruise control problem into an automated highway system control problem where the speeds of many vehicles are regulated so that they can move together as a "platoon."

Design Problem 2.3 (Fuzzy Control for a Thermal Process): This problem is used to show how you can get into trouble in fuzzy control design if you do not understand basic ideas from conventional control or if you do not tune the controller properly. Suppose that you are given the thermal process shown in Figure 4.8 on page 209 described in Chapter 4 except that you use the plant

$$\frac{\tau(s)}{q(s)} = \frac{1}{s+1}$$

(this is a thermal process with slower dynamics than the one in Chapter 4). Note that $q(t) > 0$ corresponds to adding heat while $q(t) < 0$ corresponds to cooling. Suppose that we wish to track a unit-step input of desired temperature difference τ_d with zero steady-state tracking error. Using ideas from conventional control for linear systems, you would normally first choose to put a pole of the compensator at zero since this would give you zero steady-state tracking error to a step input (why?). Next, for a linear control system design you might proceed with the design of a cascaded lead controller (why?).

Now, rather than designing a linear controller, suppose that you decide to try a fuzzy controller that has as an output $q(t)$ and inputs $g_0 e(t)$ and $g_1 \dot{e}(t)$ where $e(t) = \tau_d(t) - \tau(t)$ and g_0 and g_1 are scaling gains (i.e., a PD fuzzy controller). That is, you are ignoring that you may need an integrator in the loop to effectively eliminate steady-state tracking error. For the PD fuzzy controller, use the same membership functions as we did in Figure 2.9 on page 36 for the inverted

pendulum. Here, however, make the effective universes of discourse for $e(t)$ and $\dot{e}(t)$, $[-1, 1]$ and $[-0.5, 0.5]$, respectively, and the effective universe of discourse for $q(t)$, $[-20, 20]$ (i.e., the exact same output membership functions as for the inverted pendulum in Figure 2.9 on page 36). Use minimum for the premise and implication and COG defuzzification. For the rule-base we simply modify the one used in Table 2.1 on page 32 for the inverted pendulum: Specifically, simply multiply each element of the body of Table 2.1 by -1 and use the resulting rule-base table as a rule-base for the PD fuzzy controller (this shows one case where you can reuse rule-bases in a convenient manner). Why is this a reasonable choice for a rule-base? To explain this, compare it to the pendulum's rule-base and explain the meaning of a few of the new rules for the thermal process.

(a) Design a linear controller that will result in zero steady-state tracking error for the step input, minimize the rise time, achieve less than 5% overshoot, and try to minimize the settling time (treat the tracking error and rise-time specifications as your primary objectives, and the overshoot and settling time as your secondary objectives). Simulate the control system you design, and provide plots of τ versus t to verify that you meet the desired objectives.

(b) Simulate the fuzzy control system using the PD fuzzy controller described above. Plot $q(t)$ and $\tau(t)$ and discuss the results. Use the Runge-Kutta method for simulation with an integration step size of 0.0005 and zero initial conditions.

(c) Even though it may be more appropriate to use a PI fuzzy controller, you can tune the PD fuzzy controller to try to meet the above specifications. Tune the PD fuzzy controller by changing the scaling gains g_0 and g_1 to meet the same objectives as stated in (a). Compare the results from (a) and (b).

(d) Is it fair to compare the linear and fuzzy controllers? Which uses more computations? Is nonlinear control (fuzzy control) really needed for this linear plant?

Design Problem 2.4 (Fuzzy Control for an Automated Highway System)*:[12]
Due to increasing traffic congestion, there has been a renewed interest in the development of an automated highway system (AHS) in which high traffic flow rates may be safely achieved. Since many of today's automobile accidents are caused by human error, automating the driving process may actually increase safety on the highway. Vehicles will be driven automatically with on-board lateral and longitudinal controllers. The lateral controllers will be used to steer the vehicles around corners, make lane changes, and perform additional steering tasks. The longitudinal controllers will be used to maintain a steady velocity if a vehicle is traveling alone (conventional cruise control), follow a lead vehicle at a safe

12. Reminder: Exercises or design problems that are particularly challenging (considering how far along you are in the text) or that require you to help define part of the problem are designated with a star ("$\star$").

distance, or perform other speed/tracking tasks. For more details on intelligent vehicle highway systems see [53] and [185, 186].

The dynamics of the car-following system for the i^{th} vehicle may be described by the state vector $X_i = [\delta_i, v_i, f_i]^\top$, where $\delta_i = x_i - x_{i-1}$ is the intervehicle spacing between the i^{th} and $i-1^{st}$ vehicles, v_i is the i^{th} vehicle's velocity, and f_i is the driving/braking force applied to the longitudinal dynamics of the i^{th} vehicle. The i^{th} vehicle follows vehicle $i-1$. The longitudinal dynamics may be expressed as

$$\dot{\delta}_i = v_i - v_{i-1} \tag{2.30}$$

$$\dot{v}_i = \frac{1}{m_i}\left(-A_\rho v_i^2 - d_i + f_i\right) \tag{2.31}$$

$$\dot{f}_i = \frac{1}{\tau_i}(-f_i + u_i) \tag{2.32}$$

where u_i is the control input (if $u_i > 0$, it represents a throttle input, while if $u_i < 0$, it represents a brake input), and $m_i = 1300$ kg is the mass of all the vehicles, $A_\rho = 0.3$ Ns^2/m^2 is the aerodynamic drag for all the vehicles, $d_i = 100$ N is a constant frictional force for all the vehicles, and $\tau_i = 0.2$ sec is the engine/brake time constant for all the vehicles.

The reference input is $r(t) = 0$. The plant output is $y_i = \delta_i + \lambda_i v_i$, and we want $y_i \to 0$ for all i. This is a "velocity-dependent headway policy." As the velocity of the i^{th} vehicle increases, the distance between the i^{th} and $i-1^{st}$ vehicles should increase. A standard good driving rule for humans is to allow an intervehicle spacing of one vehicle length per 10 mph of velocity (this roughly corresponds to $\lambda_i = 0.9$ for all i).

Suppose that we wish to design a controller for each vehicle that is to be put in the AHS that will achieve good tracking with no steady state error. In fact, our goal is to make the system react as a first-order system with a pole at -1 would to a unit-step input. Suppose that the lead vehicle is commanded to have a speed of 18 m/sec for 20 sec, then switch to 22 m/sec for 20 sec, then back to 18 m/sec and repeat the alternation between 18 and 22 m/sec for a total of 300 sec.

(a) Assume that there are only two vehicles in the AHS and that you implement a controller on the following vehicle that will regulate the intervehicle spacing. Design a PD controller that will achieve the indicated specifications. For your PD controller use

$$e_i(t) = r(t) - y_i(t)$$

and

$$u_i(t) = K_{p_i} e_i(t) + K_{d_i} \frac{d}{dt} e_i(t)$$

(b) Repeat (a) except use a fuzzy controller.

(c) Repeat (a) except use a sliding-mode controller.

(d) Compare the performance of the controllers and make recommendations on which one should be used. Be careful to tune each of the controllers as well as you can so that you will feel confident about your recommendation of which approach to use.

(e) Repeat (a)–(d) for five vehicles all with different masses, aerodynamic drags, and engine/brake time constants.

Design Problem 2.5 (Fuzzy Control for a Magnetic Ball Suspension System)*: See the model of the magnetic ball suspension system shown in Figure 6.19 on page 366 in Chapter 6.

(a) Use the linear model given in Chapter 6 to design a linear controller that achieves zero steady-state tracking error and a fast rise-time with as little overshoot as possible. Demonstrate that the controller works properly for the linear plant model. Next, investigate how it performs for the nonlinear plant model (you may need to pick a reference input that is small in magnitude when you test your system in simulation with the nonlinear plant model).

(b) Repeat (a) but design a conventional nonlinear controller for the nonlinear model of the system.

(c) Repeat (b) except use a fuzzy controller.

(d) Compare the performance of the fuzzy and conventional linear and nonlinear controllers. Be careful to tune each of the controllers as well as you can so that you will feel confident about your recommendation of which approach to use.

Design Problem 2.6 (Fuzzy System Design for Basic Math Operations)*: In a PD controller, the plant input is generated by scaling the error and derivative of the error and summing these two values. A fuzzy controller that uses the error and derivative of the error as inputs can be designed to perform a similar scaling and summing operation (a linear operation), at least locally. For example, in the inverted pendulum problem we actually achieve such a scaling and summing operation with the fuzzy controllers that we designed (provided that the fuzzy controller input signals are small). The scaling is actually achieved by the scaling gains, and the summing operation is achieved by the rule-base (recall that the pattern of the consequent linguistic-numeric values in Table 2.1 on page 32 is achieved by *adding* the linguistic-numeric values associated with each of the inputs, taking the negative of the result, and saturating their values at $+2$ or -2). We see that fuzzy systems are capable of performing basic mathematical operations, at least on a region of their input space.

(a) Suppose that there are two inputs to the fuzzy system, x and y, and one output, z. Define a fuzzy system that can add two numbers that lie within the regions $x \in [-2, 2]$ and $y \in [-1, 1]$. Plot the three-dimensional nonlinear surface induced by the fuzzy system.

(b) Repeat (a) for subtraction.

(c) Repeat (a) for multiplication.

(d) Repeat (a) for division.

(e) Repeat (a) for taking the maximum of two numbers.

(f) Repeat (a) for taking the minimum of two numbers.

CHAPTER 3

Case Studies in Design and Implementation

Example is the school of mankind.

–Edmund Burke

3.1 Overview

As indicated in Chapters 1 and 2, there is no generally applicable systematic methodology for the construction of fuzzy controllers for challenging control applications that is *guaranteed* to result in a high-performance closed-loop control system. Hence, the best way to learn the basics of how to design fuzzy controllers is to do so yourself—and for a variety of applications. In this chapter we show how to design fuzzy controllers for a variety of applications in a series of case studies. We then include at the end of the chapter a variety of design problems that the reader can use to gain experience in fuzzy control system design.

Despite the lack of a general systematic design procedure, by reading this chapter you will become convinced that the fuzzy control design methodology does provide a way to design controllers for a wide variety of applications. Once the methodology is understood, it tends to provide a "way to get started," a "way to at least get a solution," and often a "way to quickly get a solution" for many types of control problems. Indeed, we have found that if you focus on *one* application, a (somewhat) systematic design methodology for that application seems to emerge from the fuzzy control approach. While the procedure is typically closely linked to application-specific concepts and parameters and is therefore not generally applicable to other plants, it does often provide a very nice framework in which the designer can think about how to achieve high-performance control.

You must keep in mind that the fuzzy controller has significant functional capabilities (recall the universal approximation property described in Section 2.3.8 on page 77) and therefore *with enough work* the designer should be able to achieve just about anything *that is possible* in terms of performance (up to the computational limits of the computer on which the controller is implemented). The problem is that just because the controller can be tuned does *not* mean that it is easy to tune, or that the current framework in which you are tuning will work (e.g., you may not be using the proper preprocessing of the fuzzy controller inputs or enough rules). We have found that while for some applications fuzzy control makes it easy to "do what makes sense" in terms of control, in others high performance is achieved only after a significant amount of work on the part of the control designer, who must get the best knowledge on how to control the system into the rule-base, which often can only occur by understanding the physics of the process very well.

Ultimately, the reader should always remember that the fuzzy control design process is nothing more than a heuristic technique for the synthesis of nonlinear controllers (there is nothing mystical about a fuzzy controller). For each of the case studies and design problems, the reader should keep in mind that an underlying nonlinearity is being shaped in the design of a fuzzy controller (recall that we showed the nonlinear surface that results from a fuzzy controller in Figure 2.35 on page 89). The shape of this nonlinearity is what determines the behavior of the closed-loop system, and it is the task of the designer to get the proper control knowledge into the rule-base so that this nonlinearity is properly shaped.

Conventional control provides a different approach to the construction of nonlinear controllers (e.g., via feedback-linearization or sliding-mode control). When you have a reasonably good model of the plant, which satisfies the necessary assumptions—and even sometimes when it does not (e.g., for some PID controllers that we design with no model or a very poor one)—then conventional control can offer quite a viable solution to a control problem. Indeed, conventional control is more widely used than fuzzy control (it is said that more than 90% of all controllers in operation are PID controllers), and for a variety of reasons may be a more viable approach (see Chapters 1 and 8 for more discussion on the relative merits of fuzzy versus conventional control). Due to the success of conventional control, we place a particular emphasis in this book on comparative analysis of fuzzy versus conventional control; the reader will see this emphasis winding its way through the case studies and design problems in this chapter. We believe that it is unwise to ignore past successes in control in the excitement over trying fuzzy control.

In this chapter we begin, in Section 3.2, by providing an overview of a general methodology for fuzzy controller design (including issues in computer-aided design) and then show how to design fuzzy controllers for a variety of challenging applications: a two-link flexible robot, a rotational inverted pendulum, a machine scheduling problem, and fuzzy decision-making systems. In each case study we have a specific objective in mind:

1. *Vibration damping for a flexible-link robot* (Section 3.3): Here, we illustrate the basic strength of the fuzzy control methodology, which is to use heuristic

information about how to achieve high-performance control. We explain in a series of steps how to quantify control knowledge in a fuzzy controller and show how performance can be subsequently improved. Moreover, we provide experimental results in each case and especially highlight the importance of understanding the physics of the underlying control problem so that appropriate control rules can be designed. In Chapters 6 and 7 we will study adaptive and supervisory fuzzy control techniques for this problem, and achieve even better performance than in this chapter, even for the case where a mass is added to the second link's endpoint.

2. *Rotational inverted pendulum* (Section 3.4): In this case study we first design a conventional linear controller for balancing the pendulum. Then we introduce a general procedure for incorporating these conventional control laws into a fuzzy controller. In this way, for small signals the fuzzy controller will act like a well-designed linear controller, and for larger signals the fuzzy controller nonlinearity can be shaped appropriately. Experimental results are provided to compare the conventional and fuzzy control approaches. In Chapter 6 we show how an adaptive fuzzy controller can be used to achieve very good balancing performance even if a sealed bottle half-filled with water is attached to the pendulum endpoint to create a disturbance.

3. *Machine Scheduling* (Section 3.5): Here, we show how a fuzzy controller can be used to schedule part processing in a simple manufacturing system. This case study is included to show how a fuzzy system has wide applicability since it can be used as a very general decision maker. Comparisons are made to conventional scheduling methods to try to uncover the advantages of fuzzy control. In Chapter 6 we extend the basic approach to provide an adaptive scheduler that can reconfigure itself to maintain throughput performance even if there are unpredictable changes in the machine.

4. *Fuzzy decision-making systems* (Section 3.6): In this case study we explain the various roles that fuzzy systems can serve in the implementation of general decision-making systems. Then we show how to construct fuzzy decision-making systems for providing warnings of the spread of an infectious disease and failure warnings for an aircraft. This case study is used to show that fuzzy systems have broad applicability outside the area of traditional feedback control.

When you complete this chapter, you will have solidified your understanding of the general fuzzy control system design methodology over that which was presented in Chapter 2 for the academic inverted pendulum design problem. Also, you will have gained an understanding of how to design fuzzy controllers for three specific applications and fuzzy decision-making systems for several applications.

As indicated above, the case studies in this chapter will actually be used throughout the remainder of the book. In particular, they will be used in Chapter 6 on adaptive fuzzy control and Chapter 7 on fuzzy supervisory control. Moreover, they will be used as design problems in this and these later chapters. Hence, you

will want to at least skim the case studies if you are concerned with understanding the corresponding later case studies where we will use adaptive and supervisory control for the same plants. However, the reader who wants to learn techniques alone and is not as concerned with applications and implementations can skip this chapter.

3.2 Design Methodology

In Chapter 2 we provided an introduction to how to design fuzzy controllers, and several basic guidelines for their design were provided in Section 2.4.4 on page 89. Here, we provide an overview of the design procedure that we have in mind when we construct the fuzzy controllers for the first two case studies in this chapter. Our methodology is as follows:

1. Try to understand the behavior of the plant, how it reacts to inputs, what are the effects of disturbances, and what fundamental limitations it presents (e.g., nonminimum phase or unstable behavior). A clear understanding comes from studying the physics of the process, developing mathematical models, using system identification methods, doing analysis, performing simulations, and using heuristic knowledge about the plant dynamics. The analysis could involve studying stability, controllability, or observability of the plant; how fast the plant can react to various inputs; or how noise propagates in the dynamics of the process (e.g., via stochastic analysis). The heuristic knowledge may come from, for example, a human operator of the process or a control engineer. Sometimes, knowledge of the plant's behavior comes from actually trying out a controller on the system (e.g., a PID, lead-lag, state-feedback, or fuzzy controller).

2. Gain a clear understanding of the closed-loop specifications (i.e., the performance objectives). These may be stated in terms of specifications on stability, rise-time, overshoot, settling time, steady-state error, disturbance rejection, robustness, and so on. You should make sure that the performance objectives are reasonable and achievable, and that they properly characterize exactly what is desired in terms of closed-loop behavior.

3. Establish the basic structure of the control system (here we assume that a "direct" (nonadaptive) controller is used). This will establish what the plant and controller inputs and outputs should be.

4. Perform an initial control design. This may be with a simple PID controller, some other linear technique (e.g., lead-lag compensation or state feedback), or a simple fuzzy controller (often you should first try a fuzzy PD, PI, or PID controller). For some basic ideas on how to design fuzzy controllers, see Chapter 2, Section 2.4.4 on page 89. The basic approaches include (a) inclusion of good control knowledge, (b) tuning the scaling gains, (c) tuning the membership functions, and (d) adding more rules or membership functions. Work hard

to tune the chosen method. Evaluate if the performance objectives are met via simulations or mathematical analysis (such as that found in Chapter 4) if you have a model.

5. If your simple initial approach to control is successful, begin working on an implementation. If it is not successful, first make sure that you are using solid control engineering ideas to pick the "nonfuzzy" part of the controller (e.g., the preprocessing of fuzzy controller inputs by choosing to use an integrator to try to remove steady-state error). If this does not work, consider the following options:

   - A more sophisticated conventional control method (e.g., feedback-linearization or sliding-mode control).
   - A more sophisticated fuzzy controller. You may need more inputs to the fuzzy controller or more rules in the rule-base. You should carefully consider if you have loaded the best knowledge about how to control the process into the rule-base (often, the problem with tuning a fuzzy controller boils down to a basic lack of understanding of how best to control the plant and the corresponding lack of knowledge in the rule-base).
   - Try designing the fuzzy controller by using a well-designed linear control technique to specify the general shape of the control surface (especially around zero) and then tune the surface starting from there (this approach is illustrated in this chapter for the rotational inverted pendulum).
   - Conventional or fuzzy adaptive or supervisory control approaches (see Chapters 6 and 7).

   Work hard to tune the chosen method. Evaluate if the performance objectives are met.

6. Repeat the above process as often as necessary, evaluating the designs in simulation and, if possible, implementation. When you have met the performance objectives for the implementation, you will likely have additional work including "burn-in" tests, marketing analyses, cost analyses, and other issues (of course, several of these will have to be considered much earlier in the design process).

Computer-aided design (CAD) packages are designed to try to help automate the above process. While we recommend that you strongly consider their use, we must reemphasize that it is best to first know how to program the fuzzy controller in a high-level language before moving on to the use of CAD packages where the user can be removed from understanding the low-level details of the operation of fuzzy systems. Once fuzzy systems are well understood, you can use one of the existing packages (e.g., the one currently in Matlab) or design a package on your own. However, you should not dismiss the importance of knowing how to code a fuzzy controller on your own. Often this is necessary for implementation anyway (e.g., in C or assembly language).

3.3 Vibration Damping for a Flexible Robot

For nearly a decade, control engineers and roboticists alike have been investigating the problem of controlling robotic mechanisms that have very flexible links. Such mechanisms are important in space structure applications, where large, lightweight robots are to be utilized in a variety of tasks, including deployment, spacecraft servicing, space-station maintenance, and so on. Flexibility is not designed into the mechanism; it is usually an undesirable characteristic that results from trading off mass and length requirements in optimizing effectiveness and "deployability" of the robot. These requirements and limitations of mass and rigidity give rise to many interesting issues from a control perspective.

In this section we present a design case study that makes use of previous experience in the modeling and control of a two-link planar flexible robot. First, though, we provide some motivation for why you would want to consider using fuzzy control for the robot.

The modeling complexity of multilink flexible robots is well documented, and numerous researchers have investigated a variety of techniques for representing flexible and rigid dynamics of such mechanisms. Equally numerous are the works addressing the control problem in simulation studies based on mathematical models, under assumptions of perfect modeling. Even in simulation, however, a challenging control problem exists; it is well known that vibration suppression in slewing mechanical structures whose parameters depend on the configuration (i.e., are time varying) can be extremely difficult to achieve. Compounding the problem, numerous experimental studies have shown that when implementation issues are taken into consideration, modeling uncertainties either render the simulation-based control designs useless, or demand extensive tuning of controller parameters (often in an ad hoc manner).

Hence, even if a relatively accurate model of the flexible robot can be developed, it is often too complex to use in controller development, especially for many control design procedures that require restrictive assumptions for the plant (e.g., linearity). It is for this reason that conventional controllers for flexible robots are developed either (1) via simple crude models of the plant behavior that satisfy the necessary assumptions (e.g., the model we develop below), or (2) via the ad hoc tuning of linear or nonlinear controllers. Regardless, heuristics enter the design process when the conventional control design process is used.

It is important to emphasize, however, that such conventional control-engineering approaches that use appropriate heuristics to tune the design have been relatively successful. For a process such as a flexible robot, you are left with the following question: How much of the success can be attributed to the use of the mathematical model and conventional control design approach, and how much should be attributed to the clever heuristic tuning that the control engineer uses upon implementation? While control engineers have a relatively good understanding of the capabilities of conventional mathematical approaches to control, much less is understood about whether or not control techniques that are designed to exploit the use of heuristic information (such as fuzzy control approaches) can perform better

than conventional techniques.

In this section we show that fuzzy control can, in fact, perform quite well for a particular two-link flexible robot. In Chapters 6 and 7 we will show how to use adaptive and supervisory fuzzy control for this same mechanism. These methods build on the direct fuzzy control methods studied in this chapter and provide the best controllers developed for this experiment to date (including many conventional methods).

3.3.1 The Two-Link Flexible Robot

In this section we describe the laboratory test bed, the control objectives, and how the robot reacts to open-loop control.

Laboratory Test Bed

The two-link flexible robot shown in Figure 3.1 consists of three main parts: (1) the robot with its sensors, (2) the computer and the interface to the robot, and (3) the camera with its computer and interface. The robot is made up of two very flexible links constrained to operate in the horizontal plane. The "shoulder link" is a counterbalanced aluminum strip that is driven by a DC direct-drive motor with an input voltage v_1. The "elbow link," which is mounted on the shoulder link endpoint, is a smaller aluminum strip. The actuator for the elbow link is a geared DC motor with an input voltage v_2. The sensors on the robot are two optical encoders for the motor shaft positions Θ_1 and Θ_2, and two accelerometers mounted on the link endpoints to measure the accelerations a_1 and a_2.

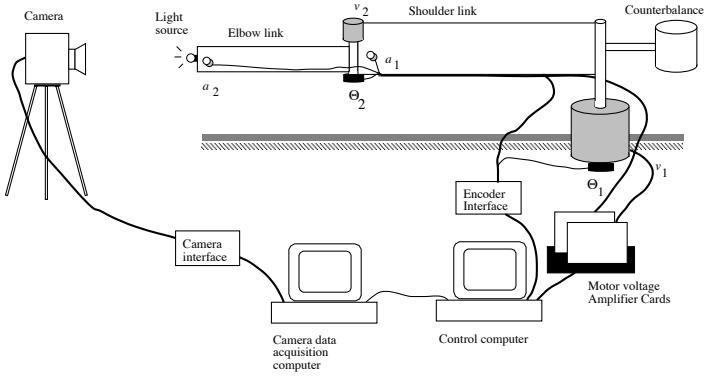


FIGURE 3.1 Two-link flexible robot setup (figure taken from [145], © IEEE).

A line scan camera is used to monitor the endpoint position of the robot for plotting; this data is not used for feedback. The sampling period used for all sensors and control updates is 15 milliseconds (ms). For comparative purposes, we use the

camera data for robot movements that begin in some position and end in a fully extended position, to approximate equal movements in each joint. When responses are plotted, the final endpoint position is nominally indicated (on the plot) to reflect (approximately) the total movement, in degrees, of the shoulder joint.

Objectives and Open-Loop Control

The primary objective of this case study is to develop a controller that makes the robot move to its desired position as quickly as possible, with little or no endpoint oscillation. To appreciate the improvement in the plant behavior due to the application of the various control strategies, we will first study how the robot operates under the "no control" situation; that is, when no external digital control algorithm is applied for vibration compensation. To implement the no control case, we simply apply $v_1 = v_2 = 0.3615$ volts at $t = 0$ seconds and return v_1 and v_2 to zero voltages as soon as the links reach their set-points. Note that for this experiment we monitor the movement of the links but do not use this information as feedback for control.

The results of the "no control" experiment are plotted in Figure 3.2, where the endpoint position shows a significant amount of endpoint oscillation. As is typical in mechanisms of this sort, inherent modal damping is present. It is well known that the effect of mass-loading a slewing flexible beam is to reduce the modal frequencies and this is indeed the case for this experiment. Indeed, when a 30-gram payload is attached to the robot endpoint, the first modal frequency of the second link (endpoint) reduces significantly. This effect causes performance degradation in fixed, linear controllers. In Figure 3.2, as in all plots to follow, endpoint position refers to the position of the elbow link endpoint. Note that the inset shown in Figure 3.2 depicts the robot slew employed. The two dashed lines describe the initial position of the links. The arrows show the direction of movement, and the solid line shows the final position of the links. Hence, for this open-loop experiment, we wanted 90 degrees of movement in each link. In the ideal case the shaft should stop moving the instant the voltage signal to the motor amplifier is cut off. But the arm had been moving at a constant velocity before the signal was cut off, and thus had a momentum that dragged the shaft past the angle at which it was to stop. This movement depends on the speed at which the arm was moving, which in turn depends on the voltage signal applied. Clearly, there is a significant need for vibration damping in endpoint positioning.

Quantitatively speaking, in terms of step-type responses (for motions through large angles in each joint), the control objectives are as follows: system settling (elimination of residual vibrations in endpoint position) within 2 seconds of motion initiation, and overshoot minimized to less than 5% deviation from the final desired position. In addition, we wish to achieve certain qualitative aspects such as eliminating jerky movements and having smooth transitions between commanded motions.

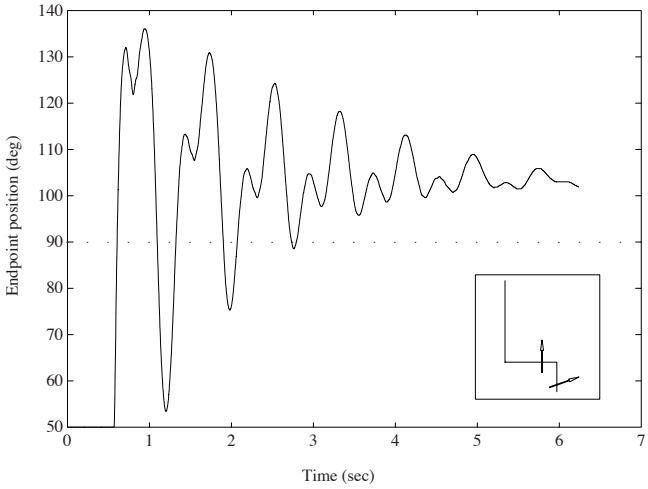


FIGURE 3.2 Endpoint position: "No control" response (figure taken from [145], © IEEE).

Model

While it is difficult to produce an accurate model of the two-link robot using modeling based on first principles, it is possible to perform system identification studies for this system to produce approximate linear models. Working along these lines, the authors in [243] developed linear models for the two-link flexible robot. In particular, random inputs were injected via the voltage inputs, data was gathered at the outputs, and a least squares method was used to compute the parameters of linear models. Several experiments had to be performed since there are two inputs and four outputs. To identify transfer functions from the inputs to the shaft velocity and endpoint acceleration for the shoulder link, the elbow link was initially fixed at a 180-degree angle (directly in line) with the shoulder link. While voltage was applied to one link it was set to zero for the other link so that it would not be commanded to move from its initial position. The sampling period for these system identification experiments is 20 ms (note that this is different from the 15-ms sampling period used in our control implementation studies to follow). Note that the joint angles Θ_1 and Θ_2 must lie in a ± 250-degree range, and v_1 and v_2 must lie in a range of ± 5 volts (the values are saturated beyond this point). The saturation constraints should be considered part of the model (so that the resulting model is nonlinear).

Let ω_1 and ω_2 denote the shaft velocity of the shoulder and elbow joints, respectively. The models produced by the system identification experiments in [243]

are given by

$$\frac{\omega_1}{v_1} = \frac{-0.0166(z - 0.6427 \pm j1.2174)(z - 1.4092)}{(z - 0.7385 \pm j0.6288)(z - 0.8165 \pm j0.2839)} \tag{3.1}$$

$$\frac{\omega_2}{v_2} = \frac{-0.1(z - 1.8062 \pm j1.7386)(z + 0.9825)}{(z - 0.7158 \pm j0.615)(z - 0.8377 \pm j0.2553)}$$

These equations provide models for relating voltages to velocities, but we actually need the models for relating voltages to positions. To get these, you can simply use a discrete approximation to an integrator (using a sampling period of 20 ms) concatenated with the models for velocities to obtain the positions Θ_1 and Θ_2.

The transfer functions that describe how the motor voltages affect the endpoint accelerations a_1 and a_2 were determined in a similar way in [243] and are given by

$$\frac{a_1}{v_1} = \frac{0.1425(z - 0.9589 \pm j0.9083)(z - 1.7945)}{(z - 0.7521 \pm j0.573)(z - 0.9365 \pm j0.139)}$$

$$\frac{a_2}{v_2} = \frac{-0.228(z - 1.5751)(z - 1.2402)}{(z - 0.9126)(z - 0.8387 \pm j0.4752)}$$

Experiments showed that lower-order models resulted in less accurate models and higher-order ones did not seem to make any of the above models more accurate. Simple inspection of the root locations in the z-plane shows that parts of the dynamics are especially lightly damped, which characterizes the vibration damping challenge for this problem.

Notice that we are ignoring certain cross-coupling effects in the model (e.g., how v_1 combined with v_2 will affect a_2); the effect of the movement of the modes, and hence plant parameters, due to mass-loading (these models are for a robot that is not mass-loaded); the effects of the position of one link on the model used for the other link; dead-zone nonlinearities due to the gearbox on the elbow motor; and many other characteristics. It is for these reasons that this model cannot be expected to be a perfectly accurate representation of the two-link robot. It is correct only under the experimental conditions outlined above. We present the model here mainly to give the reader an idea of the type of dynamics involved in this experiment and to use these models in a design problem at the end of the chapter.

We would like to emphasize that models that accurately characterize the coupling effects between the two links are particularly difficult to develop. This has significant effects on what is possible to illustrate in simulation, relative to what can be illustrated in implementation. For instance, in the two following subsections we will show that while an "uncoupled controller" (i.e., one where there are separate controllers for the shoulder and elbow links) performs adequately in implementation, significant performance improvements can be obtained by using some heuristic ideas about how to compensate for some of the coupling effects between the links

(e.g., how v_2 for the elbow link should be changed based on the acceleration a_1 of the shoulder link so that the effects of the movement of the shoulder link on the elbow link can be tailored so that the endpoint vibrations can be reduced).

We have used least squares methods to identify linear models that attempt to characterize the coupling between the two links; however, we were not able to make these accurate enough so that a coupled controller developed from these models would perform better than those developed as uncoupled controllers without this information. Hence, the case study that follows is a good example of the case where heuristic ideas about how to control a system proved to be more valuable than the models we were able to produce for the system (and significantly less work was needed to specify the heuristic ideas about compensating for coupling effects than what it took to try to construct models and develop controllers based on these).

3.3.2 Uncoupled Direct Fuzzy Control

In this section and the next we investigate the use of two types of direct fuzzy controllers for the flexible robot, one that uses information about the coupling effects of the two links (coupled direct fuzzy control) and one that does not use such information (uncoupled direct fuzzy control). The design scenario we present, although specific to the flexible robot test bed under study, may be viewed as following a general philosophy for fuzzy controller design where we are concerned with loading good control knowledge into the rule-base.

For uncoupled direct fuzzy control, two separate controllers are implemented, one for each of the two links. Each controller has two inputs and one output, as shown in Figure 3.3. The term *uncoupled* is used since the controllers operate independent of each other. No information is transferred between the shoulder and elbow motor controllers. We thus consider the robot to be made up of two separate single-link systems. In Figure 3.3, Θ_{1d} and Θ_{2d} denote the desired positions of the shoulder and elbow links, respectively, and $\Theta_1(t)$ and $\Theta_2(t)$ denote their position at time t, as measured from the optical encoders. The inputs to the shoulder link controller are the position error of the shoulder motor shaft $e_1(t) = \Theta_{1d} - \Theta_1(t)$, and the acceleration information $a_1(t)$ from the shoulder link endpoint. The output of this controller is multiplied by the output gain g_{v1} to generate the voltage signal $v_1(t)$ that drives the shoulder motor amplifier. The inputs to the elbow link controller are the elbow motor shaft position error $e_2(t) = \Theta_{2d} - \Theta_2(t)$ and the acceleration information from the elbow link endpoint $a_2(t)$. The output of this controller is multiplied by the output gain g_{v2} to generate the voltage signal $v_2(t)$ that drives the elbow motor amplifier. We did experiment with using the change in position error of each link as an input to each of the link controllers but found that it significantly increased the complexity of the controllers with very little, if any, improvement in overall performance; hence we did not pursue the use of this controller input. Typically, we use filtered signals from the accelerometers, prior to processing, to enhance their effectiveness.

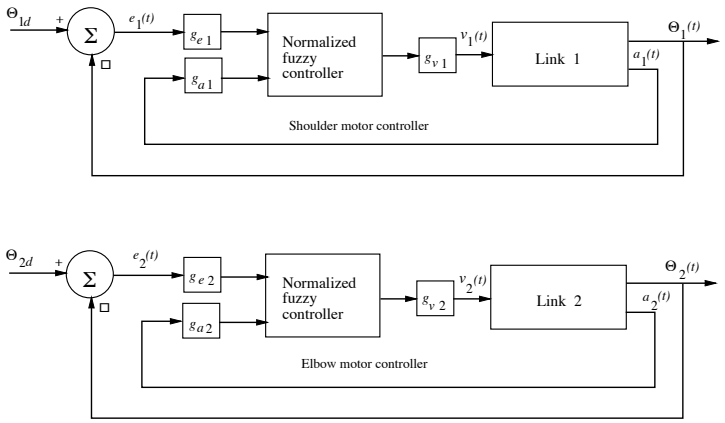


FIGURE 3.3 Fuzzy control system for uncoupled controller (figure taken from [145], © IEEE).

Fuzzy Controller Design

The input and output universes of discourse of the fuzzy controller are normalized on the range $[-1, 1]$. The gains g_{e1}, g_{e2}, g_{a1} and g_{a2} are used to map the actual inputs of the fuzzy system to the normalized universe of discourse $[-1, 1]$ and are called normalizing gains, as was discussed in Chapter 2, Exercise 2.9 on page 107. Similarly g_{v1} and g_{v2} are the output gains that scale the output of the controllers. We use singleton fuzzification and center of gravity (COG) defuzzification throughout this case study, and the minimum operator to represent the premise and implication.

The shoulder controller uses triangular membership functions, as shown in Figure 3.4. Notice that the membership functions for the input fuzzy sets are uniform, but the membership functions for the output fuzzy sets are narrower near zero. This serves to decrease the gain of the controller near the set-point so we can obtain a better steady-state control (since we do not amplify disturbances around the set-point) and yet avoid excessive overshoot (i.e., we have a nonlinear (nonuniform) spacing of the output membership function centers). The membership functions for the elbow controller are similar but have different center values for the membership functions as they use different universes of discourse than the shoulder controller. For the shoulder controller, the universe of discourse for the position error is chosen to be $[-250, +250]$ degrees. Recall from Chapter 2 that we sometimes refer to $[X, Y]$ as being the universe of discourse while in actuality the universe of discourse is made up of all real numbers (e.g., in Figure 3.4 we will refer to the universe of discourse of $e_1(t)$ as $[-250, +250]$). In addition, will refer to $Y - X$ as being the "width" of the universe of discourse (so that the width of the universe of discourse $[-250, +250]$ is 500). Recall also that by specifying the width for the universes of discourse, we are also specifying the corresponding scale factor. For example, if the input universe of discourse for $e_1(t)$ is $[-250, +250]$, then $g_{e1} = \frac{1}{250}$, and if the output universe of discourse for $v_1(t)$ is $[-0.8, +0.8]$, then $g_{v1} = 0.8$. The uni-

verse of discourse for the endpoint acceleration of the shoulder link is $[-4, +4]$ g. This width of 8 g was picked after experimentation with different slews at different speeds, upon observing the output of the acceleration sensor. The output universe of discourse of $[-0.8, +0.8]$ volts was chosen so as to keep the shaft speed within reasonable limits.

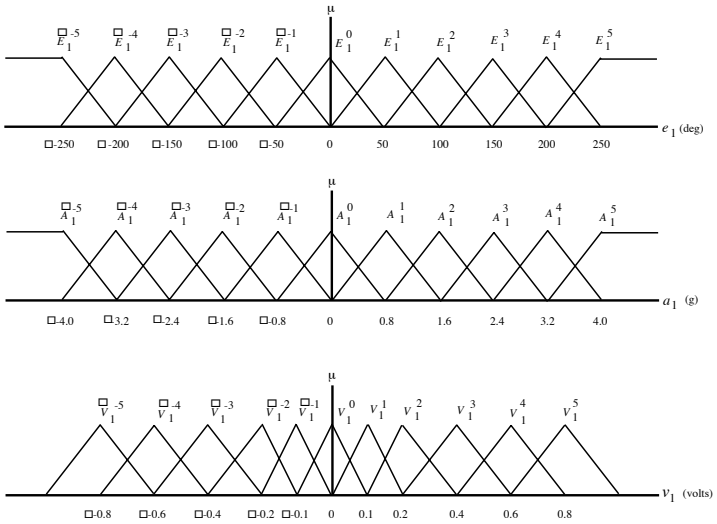


FIGURE 3.4 Membership functions for the shoulder controller (figure taken from [145], © IEEE).

For the elbow motor controller, the universe of discourse for the error input is set to $[-250, +250]$ degrees. This motor is mounted on the shoulder link endpoint and the link movement is limited by the shoulder link. The universe of discourse for the acceleration input is set to $[-8, +8]$ g, which was picked after several experiments. The universe of discourse for the output of the elbow controller is $[-5, +5]$ volts. This universe of discourse is large compared to the shoulder link as this motor is a geared-head motor with a 30:1 reduction in the motor to the output shaft speed.

The rule-base array that we use for the shoulder controller is shown in Table 3.1, and for the elbow link, in Table 3.2. Each rule-base is an 11×11 array, as we have 11 fuzzy sets on the input universes of discourse. The topmost row shows the indices for the eleven fuzzy sets for the acceleration input a_1, and the column at the extreme left shows the indices for the eleven fuzzy sets for the position error input e_1. The bodies of the tables in Tables 3.1 and 3.2 show the indices m for V_1^m in fuzzy implications of the form

$$\textbf{If } E_1^j \textbf{ and } A_1^k \textbf{ Then } V_1^m$$

where E_i^j, A_i^j, and V_i^j denote the j^{th} fuzzy sets associated with e_i, a_i, and v_i, respectively ($i = 1, 2; -5 \leq j \leq +5$). The number of rules used for the uncoupled

direct fuzzy controller is 121 for the shoulder controller, plus another 121 for the elbow controller, giving a total of 242 rules.

What is the rationale for these choices for the rule-bases? First, notice the uniformity of the indices in Tables 3.1 and 3.2. For example, for Table 3.1 if there is a positive error $e_1(t) > 0$ and a positive acceleration $a_1(t) > 0$, then the controller will input a positive voltage $v_1(t) > 0$, since in this case the link is not properly aligned but is moving in the proper direction. As the error $(e_1(t) > 0)$ decreases, and the acceleration $(a_1(t) > 0)$ decreases the controller applies smaller voltages to try to avoid overshoot. The other part of Table 3.1 and all of Table 3.2 can be explained in a similar way. Next, notice that for row $j = 0$ there are three zeros in the center of both Tables 3.1 and 3.2. These zeros have been placed so as to reduce the sensitivity of the controller to the noisy measurements from the accelerometer. Via the interpolation performed by the fuzzy controller, these zeros simply lower the gain near zero to make the controller less sensitive so that it will not amplify disturbances.

TABLE 3.1 Rule-Base for Shoulder Link

V_1^m		A_1^k										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
E_1^j	-5	-5	-5	-5	-4	-4	-3	-3	-2	-2	-1	0
	-4	-5	-5	-4	-4	-3	-3	-2	-2	-1	0	1
	-3	-5	-4	-4	-3	-3	-2	-2	-1	0	1	2
	-2	-4	-4	-3	-3	-2	-2	-1	0	1	2	2
	-1	-4	-3	-3	-2	-2	-1	0	1	2	2	3
	0	-4	-3	-2	-1	0	0	0	1	2	3	4
	1	-3	-2	-2	-1	0	1	2	2	3	3	4
	2	-2	-2	-1	0	1	2	2	3	3	4	4
	3	-2	-1	0	1	2	2	3	3	4	4	5
	4	-1	0	1	2	2	3	3	4	4	5	5
	5	0	1	2	2	3	3	4	4	5	5	5

Experimental Results

The endpoint position response for the uncoupled fuzzy controller is shown in Figure 3.5. The robot was commanded to slew 90 degrees for each link from the initial position (shown by the dashed lines in the inset) to its fully extended position (shown by the solid lines). Other "counterrelative" and small-angle movements produced similar results in terms of the quality of the responses. From the plot in Figure 3.5, we see that the magnitude of the endpoint oscillations is reduced as compared to the "no control" case, and the settling time is also improved (see Figure 3.2 on page 127). In the initial portion of the response (between 0.8 and 2.0 sec), we see large oscillations due to the fact that the controllers are uncoupled. That is, the shoulder link comes close to its set-point at around 0.9 seconds but is still traveling at a high speed. When the controller detects this, it tries to cut

TABLE 3.2 Rule-Base for Elbow Link

V_2^m		A_2^k										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
E_2^j	-5	-5	-5	-4	-4	-3	-3	-3	-2	-2	-1	0
	-4	-5	-4	-4	-3	-3	-3	-2	-2	-1	0	1
	-3	-4	-4	-3	-3	-3	-2	-2	-1	0	1	2
	-2	-4	-3	-3	-3	-2	-2	-1	0	1	2	2
	-1	-4	-3	-3	-2	-2	-1	0	1	2	2	3
	0	-4	-3	-2	-1	0	0	0	1	2	3	4
	1	-3	-2	-2	-1	0	1	2	2	3	3	4
	2	-2	-2	-1	0	1	2	2	3	3	3	4
	3	-2	-1	0	1	2	2	3	3	3	4	4
	4	-1	0	1	2	2	3	3	3	4	4	5
	5	0	1	2	2	3	3	3	4	4	5	5

the speed of the link by applying an opposite voltage at around 0.9 seconds. This causes the endpoint of the elbow link to accelerate due to its inertia, causing it to oscillate with a larger magnitude. When the controller for the elbow link detects this sudden change, it outputs a large control signal in order to move the shaft in the direction of acceleration so as to damp these oscillations. Once the oscillations are damped out, the controller continues to output signals until the set-point is reached.

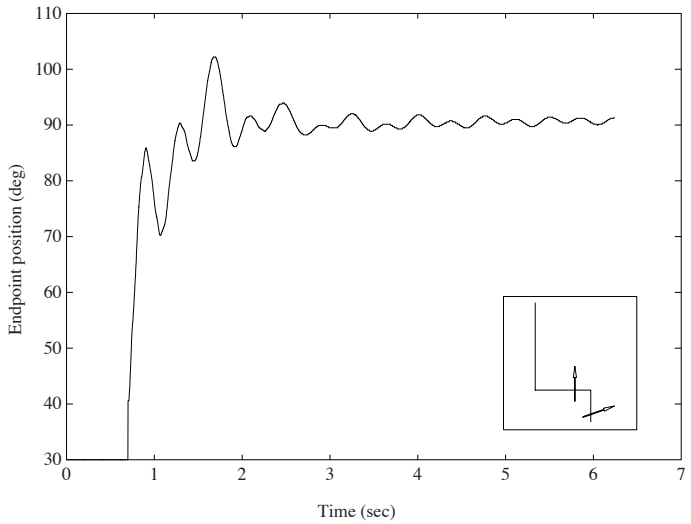


FIGURE 3.5 Endpoint position for uncoupled controller design (figure taken from [145], © IEEE).

Note that a portion of the oscillation is caused by the dead-zone nonlinearity in the gearbox of the elbow motor. The sudden braking of the shoulder link causes the elbow link to jerk, and the link oscillates in the dead-zone, creating what is similar to a limit-cycle effect. One way of preventing these oscillations in the link is to slow down the speed of the elbow link until the shoulder link is moving fast and speed it up as the shoulder link slows down. This would ensure that the elbow link is not allowed to oscillate as the motor is moving fast, and that the driven gear does not operate in the dead-zone. This control technique will be examined in the next section when we couple the acceleration feedback signals from the robot.

Figure 3.6 shows the response of the plant with a payload. The payload used was a 30-gram block of aluminum attached to the elbow link endpoint. A slew of 90 degrees for each link was commanded, as shown in the inset. The payload at the end of the elbow link increases the inertia of the link and reduces the modal frequencies of oscillation. In this case this reduction in the frequency positively affected the controller's ability to dampen the oscillation caused by the dead-zone, as compared to the unloaded case shown in Figure 3.5.

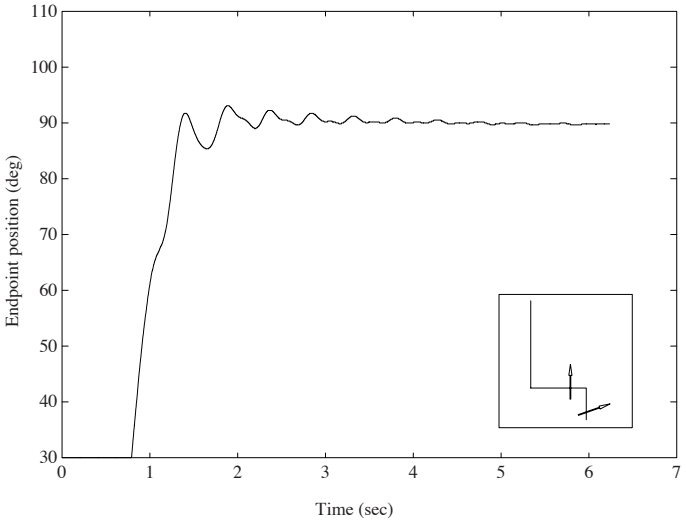


FIGURE 3.6 Endpoint position for uncoupled controller design with payload (figure taken from [145], © IEEE).

3.3.3 Coupled Direct Fuzzy Control

While the two uncoupled controllers provide reasonably good results, they are not able to take control actions that are directly based on the movements in both links. In this section we investigate the possibility of improving the performance by cou-

pling the two controllers; this can be done by using either the position information, the acceleration information, or both. From the tests on the independent controllers, it was observed that the acceleration at the endpoint of the *shoulder* link significantly affected the oscillations of the elbow link endpoint, whereas the acceleration at the endpoint of the *elbow* link did not significantly affect the shoulder link. The position of one link does not have a significant effect on the vibrations in the other. As the primary objective here is to reduce the vibration at the endpoint as much as possible while still achieving adequate slew rates, it was decided to couple the controller for the elbow link to the shoulder link using the acceleration feedback from the endpoint of the shoulder link; this is shown schematically in Figure 3.7. Note that in addition to the six normalizing gains g_{e1}, g_{e2}, g_{a1}, g_{a2}, g_{v1}, and g_{v2}, a seventh gain g_{a12} is added to the system. This gain can also be varied to tune the controller and need not be the same as g_{a1}.

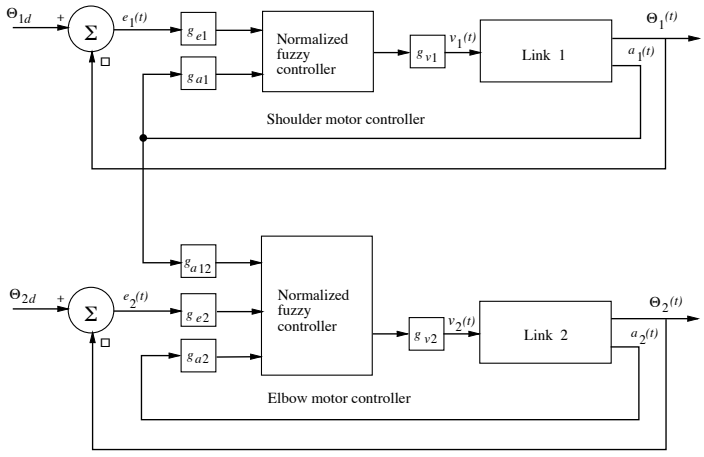


FIGURE 3.7 Coupled fuzzy controller (figure taken from [145], © IEEE).

Fuzzy Controller Design

Essentially, in coupling the controllers we are using our experience and intuition to redesign the fuzzy controller. The rule-base and the membership functions for the shoulder link are kept the same as in Figure 3.4 and Table 3.1, and the rule-base for the elbow link is modified to include the acceleration information from the shoulder link endpoint. Adding a third premise term to the premises of the rules in the rule-base in this manner will, of course, increase the total number of rules. The number of fuzzy sets for the elbow controller was therefore reduced to seven in order to keep the number of rules at a reasonable level. The number of rules for the second link with seven fuzzy sets increased to 343 ($7 \times 7 \times 7$). Hence, the

number of rules used for the coupled direct fuzzy controller is 121 for the shoulder controller, plus 343 for the elbow link controller, for a total of 464 rules.

The membership functions for the elbow controller are shown in Figure 3.8. The universe of discourse for the position error is $[-250, +250]$ degrees, and for the elbow link endpoint acceleration it is $[-8, +8]$ g, as in the uncoupled case. The universe of discourse for the shoulder link acceleration is $[+2, -2]$ g. This smaller range was chosen to make the elbow link controller sensitive to small changes in the shoulder link endpoint oscillation. The universe of discourse for the output voltage is $[-4, +4]$ volts.

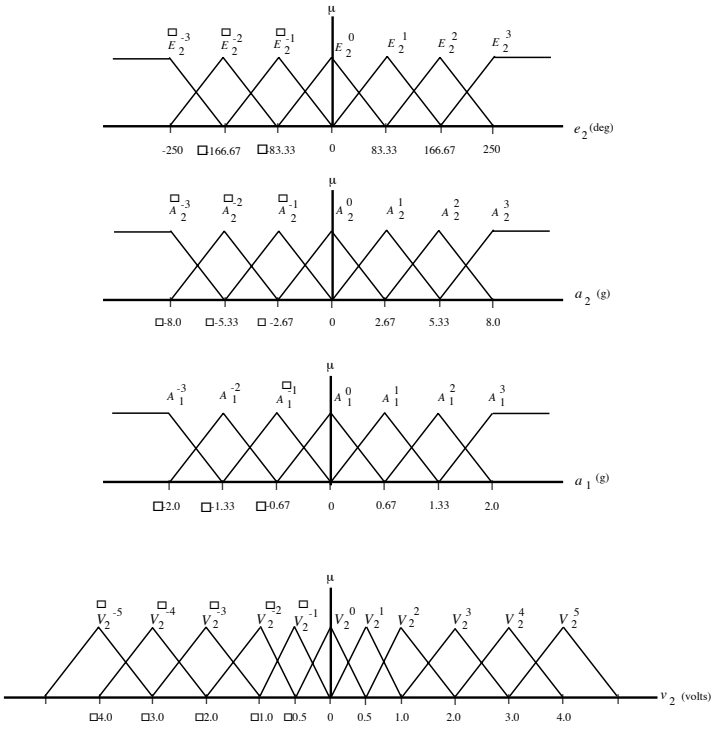


FIGURE 3.8 Membership functions for the elbow controller using coupled control (figure taken from [145], © IEEE).

Tables 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, and 3.9 depict a three-dimensional rule-base table for the elbow link. Table 3.6 represents the case when the acceleration input from the shoulder link is zero, and is the center of the rule-base (the body of the table denotes the indices m for V_2^m). Tables 3.3, 3.4, and 3.5 are for the case when the shoulder endpoint acceleration is negative, and Tables 3.7, 3.8, and 3.9 are for the case when the shoulder endpoint acceleration is positive. The central portion of the rule-base makes use of the entire output universe of discourse. This is the portion

of the rule-base where the acceleration input from the shoulder link endpoint is zero or small. As we move away from the center of the rule-base (to the region where the shoulder link endpoint acceleration is large), only a small portion of the output universe of discourse is used to keep the output of the controller small. Thus the speed of the elbow link is dependent on the acceleration input from the shoulder link endpoint. The speed of the elbow link is decreased if the acceleration is large and is increased as the acceleration input decreases.

TABLE 3.3 A_1^{-3} portion of Rule-Base Array for the Elbow Link

A_1^{-3} / V_2^m		A_2^k						
		-3	-2	-1	0	1	2	3
E_2^j	-3	-3	-3	-2	-2	-1	-1	0
	-2	-3	-2	-2	-1	-1	0	1
	-1	-2	-2	-1	-1	0	1	1
	0	-2	-1	-1	0	1	1	2
	1	-1	-1	0	1	1	2	2
	2	-1	0	1	1	1	2	2
	3	0	1	1	1	2	2	2

TABLE 3.4 A_1^{-2} portion of Rule-Base Array for the Elbow Link

A_1^{-2} / V_2^m		A_2^k						
		-3	-2	-1	0	1	2	3
E_2^j	-3	-3	-3	-3	-2	-2	-1	0
	-2	-3	-3	-2	-1	-1	0	1
	-1	-3	-2	-2	-1	0	1	1
	0	-2	-2	-1	0	1	2	2
	1	-2	-1	0	1	1	2	2
	2	-1	0	1	1	2	2	2
	3	0	1	1	2	2	2	3

Also note that in Tables 3.5, 3.6, and 3.7 there are three zeros in the middle rows to reduce the sensitivity of the controller to the noisy accelerometer signal. This noise is not a significant problem when the endpoint is oscillating, and so the rule-base does not have the zeros in the outer region. Taking the rule-base as a three-dimensional array, we get a central cubical core made up of zeros. Also notice that some parts of the rule-base, especially toward the extremes of the third dimension, are not fully uniform. This has been done to slow down the elbow link when the acceleration input from the shoulder link is very large. Overall, we see that we are incorporating our understanding of the physics of the plant into the

TABLE 3.5 A_1^{-1} portion of Rule-Base Array for the Elbow Link

A_1^{-1} / V_2^m		A_2^k						
		-3	-2	-1	0	1	2	3
E_2^j	-3	-4	-4	-3	-3	-2	-1	0
	-2	-4	-3	-3	-2	-1	0	1
	-1	-3	-3	-2	-1	0	1	1
	0	-2	-2	0	0	0	1	2
	1	-2	-1	0	1	2	2	3
	2	-1	0	1	2	2	3	3
	3	0	1	2	2	3	3	3

TABLE 3.6 A_1^0 portion of Rule-Base Array for the Elbow Link

A_1^0 / V_2^m		A_2^k						
		-3	-2	-1	0	1	2	3
E_2^j	-3	-5	-4	-4	-3	-3	-2	0
	-2	-4	-4	-3	-2	-1	0	1
	-1	-4	-3	-2	-1	0	1	2
	0	-2	-1	0	0	0	1	2
	1	-2	-1	0	1	2	3	4
	2	-1	0	2	2	3	4	4
	3	0	1	2	3	4	4	5

rule-base. We are shaping the nonlinearity of the fuzzy controller to try to improve performance.

The coupled direct fuzzy controller seeks to vary the speed of the elbow link depending on the amplitude of oscillations in the shoulder link. If the shoulder link is oscillating too much, the speed of the elbow link is reduced so as to allow the oscillations in the shoulder link to be damped; and if there are no oscillations in the shoulder link, then the second link speed is increased. We do this to eliminate the oscillation of the elbow link close to the set-point, where the control voltage from the elbow controller is small. This scheme works well as will be shown by the results, but the drawback is that it slows down the overall plant response as compared to the uncoupled case (i.e., it slows the slew rate).

Experimental Results

The experimental results obtained using coupled direct fuzzy control are shown in Figure 3.9. The slew requested here is the same as in the case of the uncoupled direct fuzzy control experiment (Figure 3.5) as shown by the inset—that is, 90 degrees for each link. We also ran experiments for "counterrelative" and small-angle slews and obtained results of a similar nature. Note that there is no overshoot in the response,

TABLE 3.7 A_1^1 portion of Rule-Base Array for the Elbow Link

$\boxed{A_1^1}$ / V_2^m		A_2^k						
		-3	-2	-1	0	1	2	3
E_2^j	-3	-4	-3	-3	-2	-2	-1	0
	-2	-3	-3	-2	-2	-1	0	1
	-1	-3	-2	-2	-1	0	1	2
	0	-2	-1	0	0	0	1	2
	1	-2	-1	0	1	2	3	3
	2	-1	0	1	2	3	3	4
	3	0	1	2	3	3	4	4

TABLE 3.8 A_1^2 portion of Rule-Base Array for the Elbow Link

$\boxed{A_1^2}$ / V_2^m		A_2^k						
		-3	-2	-1	0	1	2	3
E_2^j	-3	-3	-2	-2	-1	-1	-1	0
	-2	-3	-2	-2	-1	-1	0	1
	-1	-2	-2	-1	-1	0	1	2
	0	-2	-1	-1	0	1	1	2
	1	-1	-1	0	1	1	2	3
	2	-1	0	1	1	2	3	3
	3	0	1	2	2	3	3	4

with negligible residual vibrations. The dip in the curve in the initial part of the graph is due to the first link "braking" as it reaches the set-point and is primarily due to the dead-zone nonlinearity in the gears. As the shoulder link brakes, the elbow link is accelerated due to its inertia. The elbow link, which was at one end of its dead-zone while the shoulder was moving, shoots to the other end of the dead-zone, causing the local maxima seen in Figure 3.9 at around 0.9 seconds. The link recoils due to its flexibility and starts moving to the lower end of the dead-zone. By this time the elbow motor speed increases and prevents further oscillation of the elbow link in the dead-zone.

Notice that the multiple oscillations in the elbow link have been eliminated as compared to Figure 3.5 on page 133. This is due to the fact that when the shoulder link reaches its set-point, the elbow link is still away from its set-point, and as the shoulder link slows down, the elbow link motor speeds up and keeps the elbow link at one end of the dead-zone, preventing oscillation. Also notice that the rise time has increased in this case compared to that of the uncoupled case due to the decrease in speed of the second link while the first link is moving. This fact (increase in rise-time) and, especially, the schema embodied in the coupled-controller rule-base contribute to the reduction in endpoint residual vibration.

Experimentally, we have determined that the dip in the curve can be decreased,

TABLE 3.9 A_1^3 portion of Rule-Base Array for the Elbow Link

A_1^3 / V_2^m		A_2^k						
		-3	-2	-1	0	1	2	3
E_2^j	-3	-2	-2	-2	-1	-1	-1	0
	-2	-2	-2	-1	-1	-1	0	1
	-1	-2	-2	-1	-1	0	1	2
	0	-2	-1	-1	0	1	1	2
	1	-1	-1	0	1	1	2	2
	2	-1	0	1	1	2	2	3
	3	0	1	1	2	2	3	3

but not completely eliminated as the rule-base does not have enough "granularity" near zero (i.e., enough membership functions and rules). To alleviate this problem, a "supervisor" can be used to change the granularity of the rule-base as the shoulder link comes close to its desired set-point by changing the universes of discourse and the appropriate normalizing gains. This would produce finer control close to the set-point, resulting in a smoother transition in the speed of the shoulder link (this effect could also be achieved via the addition of more membership functions and hence rules, but this will adversely affect computational complexity). We will investigate the use of such a supervisor in Chapter 7.

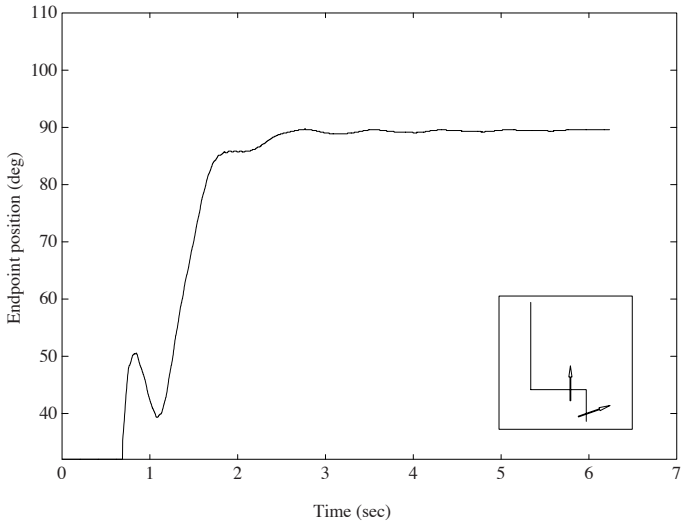


FIGURE 3.9 Endpoint position for coupled controller design (figure taken from [145], © IEEE).

Figure 3.10 shows the endpoint response of the robot with a 30-gram payload attached to its endpoint. The commanded slew is 90 degrees for each link, as shown in the inset. Notice that the dip in the curve (between 1.0 and 1.5 sec) is reduced as compared to the case without a payload (Figure 3.9). This is due to the increased inertia of the elbow link, which reduces the frequency of oscillation of the link, as the elbow link motor speeds up at this point, preventing further oscillations. Obviously, there is performance degradation relative to Figure 3.9 due to the fact that the modal frequencies of the flexible links (particularly the elbow link) have changed with the additional payload attached to the endpoint.

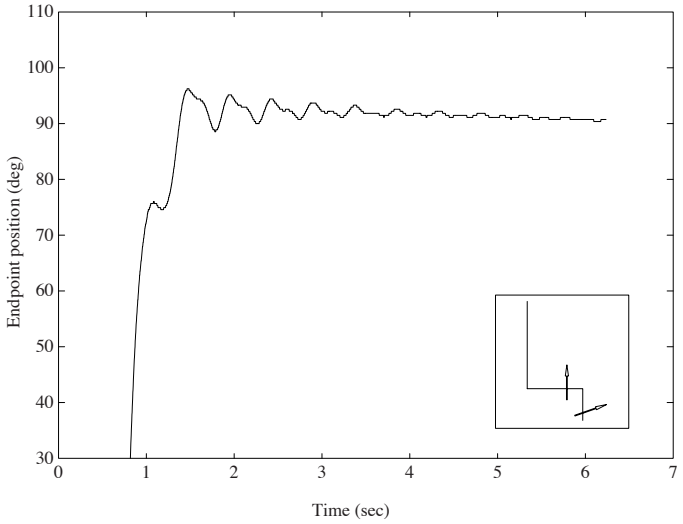


FIGURE 3.10 Endpoint position for coupled controller design with payload (figure taken from [145], © IEEE).

This completes our case study for direct fuzzy control of the flexible robot. The reader should note that while the performance obtained here compares favorably with all previous conventional control approaches studied to date for this experimental apparatus, it is not the best possible. In particular, we will show in Chapter 6 how to use adaptive fuzzy control to synthesize and later tune the fuzzy controller when there are payload variations. Moreover, we will show in Chapter 7 how a supervisory fuzzy control approach can be used to incorporate abstract ideas about how to achieve high-performance control and in fact improve performance over the direct and adaptive fuzzy control approaches (and all past conventional methods).

3.4 Balancing a Rotational Inverted Pendulum

One of the classical problems in the study of nonlinear systems is that of the inverted pendulum. The primary control problem you consider with this system is regulating the position of the pendulum (typically a rod with a mass at the endpoint) to the vertical "up" position (i.e., balancing it). A secondary problem is that of "swinging up" the pendulum from its rest position (vertical "down") to the vertical "up" position. Often, actuation is accomplished via a motor that provides a translational motion to a cart on which the pendulum is attached with a hinge. In this case study actuation of the pendulum is accomplished through *rotation* of a separate, attached link referred to henceforth as the "base."

3.4.1 The Rotational Inverted Pendulum

In this section we describe the laboratory test bed, a model of the pendulum, and a method to swing up the pendulum.

Laboratory Test Bed

The test bed consists of three primary components: (1) the plant, (2) digital and analog interfaces, and (3) the digital controller. The overall system is shown in Figure 3.11. The plant consists of a pendulum and a rotating base made of aluminum rods, two optical encoders as the angular position sensors, and a permanent-magnet DC motor to move the base. As the base rotates through the angle θ_0, the pendulum is free to rotate through its angle θ_1 made with the vertical. Interfaces between the digital controller and the plant consist of two data-acquisition cards and some signal conditioning circuitry. The sampling period for all experiments on this system is 10 ms (smaller sampling times did not help improve performance).

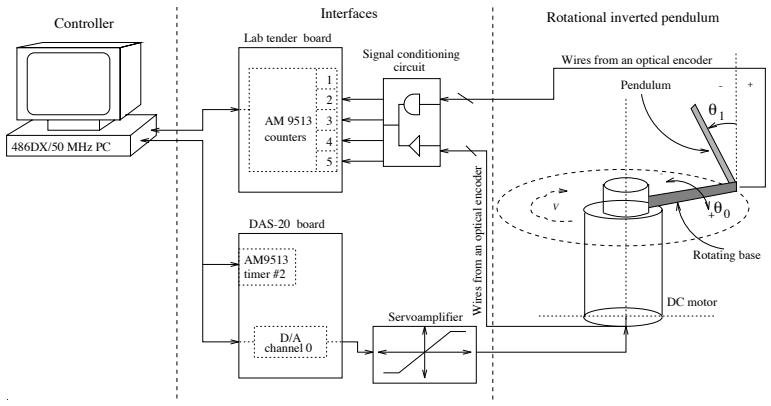


FIGURE 3.11 Hardware setup (figure taken from [235], © IEEE).

Model

The differential equations that approximately describe the dynamics of the plant are given by

$$\ddot{\theta}_0 = -a_p\dot{\theta}_0 + K_p\, v_a$$
$$\ddot{\theta}_1 = -\frac{C_1}{J_1}\dot{\theta}_1 + \frac{m_1\, g\, l_1}{J_1}\sin(\theta_1) + K_1\ddot{\theta}_0$$

where, again, θ_0 is the angular displacement of the rotating base, $\dot{\theta}_0$ is the angular speed of the rotating base, θ_1 is the angular displacement of the pendulum, $\dot{\theta}_1$ is the angular speed of the pendulum, v_a is the motor armature voltage, $K_p = 74.8903$ rad-s^{-2}-v^{-1} and $a_p = 33.0408$ s^{-2} are parameters of the DC motor with torque constant $K_1 = 1.9412 \times 10^{-3}$ kg-m/rad, $g = 9.8066$ m/sec^2 is the acceleration due to gravity, $m_1 = 0.086184$ kg is the pendulum mass, $l_1 = 0.113$ m is the pendulum length, $J_1 = 1.3011 \times 10^{-3}$ N-m-s^2 is the pendulum inertia, and $C_1 = 2.9794 \times 10^{-3}$ N-m-s/rad is a constant associated with friction. Note that the sign of K_1 depends on whether the pendulum is in the inverted or noninverted position. In particular, for $\frac{\pi}{2} < \theta_1 < \frac{3\pi}{2}$ (pendulum hanging down) we have $K_1 = 1.9412 \times 10^{-3}$, and $K_1 = -1.9412 \times 10^{-3}$ otherwise. Hence, to properly simulate the system you change the sign of K_1 depending on the value of θ_1.

For controller synthesis we will require a state-variable description of the pendulum system. This is easily done by defining state variables $x_1 = \theta_0$, $x_2 = \dot{\theta}_0$, $x_3 = \theta_1$, and $x_4 = \dot{\theta}_1$, and control signal $u = v_a$ to get

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -a_p\, x_2 + K_p\, u \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= -\frac{K_1\, a_p}{J_1}x_2 + \frac{m_1\, g\, l_1}{J_1}\sin(x_3) - \frac{C_1}{J_1}x_4 + \frac{K_1\, K_p}{J_1}u
\end{aligned} \tag{3.2}$$

Linearization of these equations about the vertical position (i.e., $\theta_1 = 0$), results in the linear, time-invariant state variable model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -33.04 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 49.30 & 73.41 & -2.29 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 74.89 \\ 0 \\ -111.74 \end{bmatrix} u$$

Clearly, we cannot expect the above models to perfectly represent the physical system. We are ignoring saturation effects, motor dynamics, friction and dead-zone nonlinearities for movement of the links, and other characteristics. We present the model here to give the reader an idea of how the physical system behaves and to make it possible for the reader to study fuzzy controller design and simulation in the design problems at the end of the chapter.

Swing-Up Control

Because we intend to develop control laws that will be valid in regions about the vertical up position, it is necessary to swing the pendulum up so that it is near vertical at near zero angular velocity. Elaborate schemes can be used for this task, but for the purposes of this case study, we choose to use a simple heuristic procedure called an "energy-pumping strategy."

The goal of this simple swing-up strategy is to "pump" energy into the pendulum link in such a way that the energy or magnitude of each swing increases until the pendulum approaches its inverted position. To apply such an approach, consider how you would (intuitively) swing the pendulum from its vertical down position to its vertical up position. If the rotating base is repeatedly swung to the left and then right at an appropriate magnitude and frequency, the magnitude of the pendulum angle θ_1 relative to the down position will increase with each swing. Swinging the pendulum in this fashion is continued until θ_1 is close to zero (vertical up), and we try to design the swing up strategy so that $\dot{\theta}_1$ is also near zero at this point (so that it is nearly balanced for an instant). Then, the swing up controller is turned off and a "balancing controller" is used to catch and balance the pendulum (we switch the swing-up controller off and the balancing controller on when $|\theta_1| < 0.3$ rad). Next, we explain the details of how such a swing-up strategy can be implemented.

Suppose that initially $\theta_1(0) = \pi$ and $\theta_0(0) = 0$. We use a swing-up strategy that has $u = K_p(\theta_0^{ref} - \theta_0)$ where θ_0^{ref} is switched between $+\Gamma$ and $-\Gamma$ where $\Gamma > 0$ is a parameter that specifies the amplitude of the rotating base movement during swing-up. The criterion for switching between $\pm\Gamma$ is that if the pendulum base is moving toward $+\Gamma$ then we use $u = K_p(\Gamma - \theta_0)$ until $\dot{\theta}_1$ is close to zero (indicating that the pendulum has swung up as far as it can for the given movement from the base). Then we switch the control to $u = K_p(-\Gamma - \theta_0)$ to drive the base in the other direction until $\dot{\theta}_1$ is close to zero again. Then the process repeats until the pendulum position is brought close to the vertical up position, where the swing-up control is turned off and the balancing control is switched on. In addition to manual tuning of Γ, it is necessary for the operator of the experiment to perform some initial tuning for the positioning control gain K_p. Basically, the gain K_p is chosen just large enough so that the actuator drives the base fast enough without saturating the control output.

Finally, we note that if the dynamics of the pendulum are changed (e.g., adding extra weight to the endpoint of the pendulum), then the parameter Γ must be retuned by the operator of the experiment. Moreover, retuning is sometimes even needed if the temperature in the room changes.

3.4.2 A Conventional Approach to Balancing Control

Although numerous linear control design techniques have been applied to this particular system, here we consider the performance of only the linear quadratic regulator (LQR) [3, 12]. Our purpose is twofold: First, we wish to form a baseline for comparison to fuzzy control designs to follow, and second, we wish to provide a starting point for synthesis of the fuzzy controller.

Because the linearized system is completely controllable and observable, linear state-feedback strategies, such as the LQR, are applicable. The performance index for the LQR is

$$J = \int_0^\infty (x(t)^\top Q x(t) + u(t)^\top R u(t)) dt$$

where Q and R are the weighting matrices of appropriate dimension corresponding to the state x and input u, respectively. Given fixed Q and R, the feedback gains that optimize the function J can be uniquely determined by solving an algebraic Riccati equation (e.g., in Matlab). Because we are more concerned with balancing the pendulum than regulating the base position, we put the highest priority on controlling θ_1 by choosing the weighting matrices $Q = diag(1, 0, 5, 0)$ (a 4×4 diagonal matrix with zeros off the diagonal) and $R = 1$. The optimal feedback gains corresponding to the weighting matrices Q and R are $k_1 = -1.0$, $k_2 = -1.191$, $k_3 = -9.699$, and $k_4 = -0.961$ (these are easily found in Matlab). Hence, our controller is $u(t) = Kx(t)$ where $K = [k_1, k_2, k_3, k_4]^\top$. Although observers may be designed to estimate the states $\dot{\theta}_1$ and $\dot{\theta}_0$, we choose to use an equally effective and simple backward difference approximation for each derivative.

Using the swing-up control strategy tuned for the nominal system (with $K_p = 0.5$ and $\Gamma = 1.81$ rad), the results of using the LQR controller for balancing, after the pendulum is swung up, are given in Figure 3.12. These are actual implementation results for the case where there is no additional mass added to the endpoint (i.e., what we will call the "nominal case"). The base angle is shown in the top plot, the pendulum angle in the center plot, and the control output in the bottom plot. When the LQR controller gains (k_1 through k_4) are implemented on the actual system, some trial-and-error tuning is required (changing the gains by about 10%) to obtain performance matching the predicted results that we had obtained from simulation. Overall, we see that the LQR is quite successful at balancing the pendulum.

3.4.3 Fuzzy Control for Balancing

Synthesis of the fuzzy controllers to follow is aided by (1) a good understanding of the pendulum dynamics (the analytical model and intuition related to the physical process), and (2) experience with performance of linear control strategies such as a proportional-derivative controller and the above LQR. Aside from serving to illustrate procedures for synthesizing a fuzzy controller, several reasons arise for considering the use of a nonlinear control scheme for the pendulum system. Because linear controllers are designed based on a linearized model of the system, they are inherently valid only for a region about a specific point (in this case, the vertical up position). For this reason, such linear controllers tend to be very sensitive to parametric variations, uncertainties, and disturbances. This is indeed the case for the experimental system under study. When an extra weight or *sloshing liquid* (using a watertight bottle) is attached at the endpoint of the pendulum, the performance of all the linear controllers we tested degrades considerably, often resulting in unstable behavior. Hence, to enhance the performance of the balancing

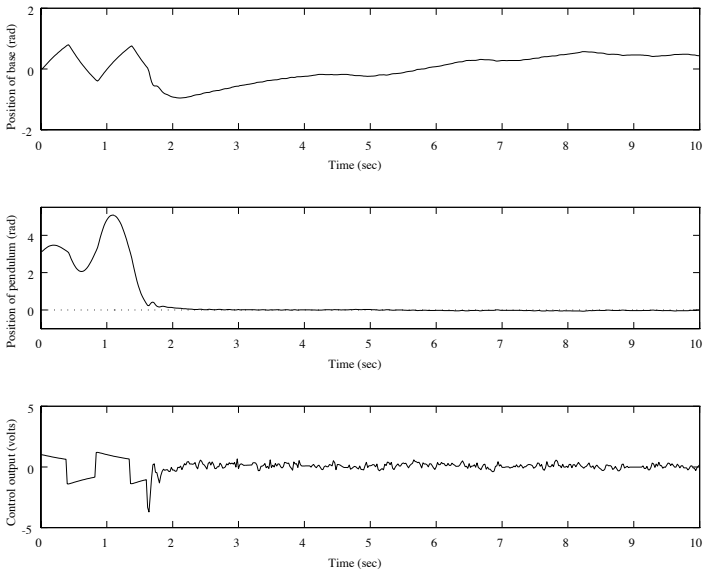


FIGURE 3.12 LQR on the *nominal* system (figure taken from [235], © IEEE).

control, you naturally turn to some nonlinear control scheme that is expected to exhibit improved performance in the presence of nonlinearities, disturbances, and uncertainties in modeling. We will investigate two such nonlinear controllers in this book: in this section we describe how to construct a direct fuzzy controller, and in Chapter 6 we develop an adaptive fuzzy controller.

The Fuzzy Controller

The fuzzy controller is shown in Figure 3.13. Similar to the linear quadratic regulator, the fuzzy controller for the inverted pendulum system will have four inputs and one output. The four inputs to the fuzzy controller are the position error of the base e_1, its change in error e_2, the position error of the pendulum e_3, and the change in error e_4.

Our fuzzy controller utilizes singleton fuzzification and symmetric, triangular membership functions on the controller input and output universes of discourse. We use seven membership functions for each input, uniformly distributed across their universes of discourse, as shown in Figure 3.14 (the choice of the scaling gains that results in the scaling for the horizontal axes is explained below). The linguistic values for the i^{th} input are denoted by $\tilde{E}_i^j$ where $j \in \{-3, -2, -1, 0, 1, 2, 3\}$. Linguistically, we would therefore define $\tilde{E}_i^{-3}$ as "negative large," $\tilde{E}_i^{-2}$ as "negative medium," $\tilde{E}_i^0$ as "zero," and so on. We use the minimum operation to represent the premise and the implication, and COG defuzzification. We need to specify the

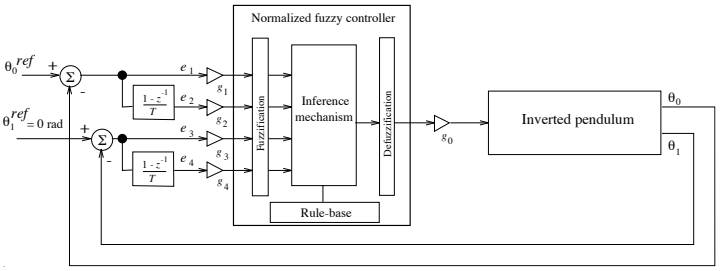


FIGURE 3.13 Block diagram of direct fuzzy controller for the rotational inverted pendulum (figure taken from [235], © IEEE).

output membership functions, the rules, and the gains g_i to complete the design of our fuzzy controller.

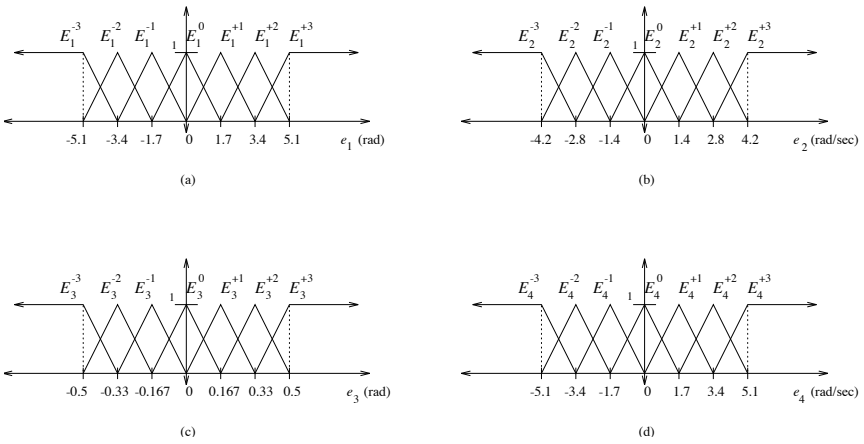


FIGURE 3.14 Four sets of input membership functions: (a) "Base position error" (e_1), (b) "base derivative error" (e_2), (c) "pendulum position error" (e_3), and (d) "pendulum derivative error" (e_4) (figures taken from [235], © IEEE).

To synthesize a fuzzy controller, we pursue the idea of making it match the LQR for small inputs since the LQR was so successful. Then, we still have the added tuning flexibility with the fuzzy controller to shape its control surface so that for larger inputs it can perform differently from the LQR (and, if we get the right knowledge into the rule-base, better).

Fuzzy Controller Design via Copying a Linear Controller

Recall from our discussion in Chapter 2 that a fuzzy system is a static nonlinear map between its inputs and output. Certainly, therefore, a linear map such as the

LQR can be easily approximated by a fuzzy system (at least for small values of the inputs to the fuzzy system). Two components of the LQR are the optimal gains and the summation operation; the optimal gains can be replaced with the scaling gains of a fuzzy system, and the summation can essentially be incorporated into the rule-base of a fuzzy system. By doing this, we can effectively utilize a fuzzy system to expand the region of operation of the controller beyond the "linear region" afforded by the design process that relied on linearization. Intuitively, this is done by making the "gain" of the fuzzy controller match that of the LQR when the fuzzy controller inputs are small, while shaping the nonlinear mapping representing the fuzzy controller for larger inputs (in regions further from zero).

Implementing the summation operation in the rule-base is straightforward. First, we assume that all the input universes of discourse have uniformly distributed triangular membership functions, such as those shown in Figure 3.14, but with effective universes of discourse all given by $[-1, +1]$ (i.e., so that the left-most membership function and the right-most membership function saturate at -1 and $+1$, respectively). Then we arrange the If-Then rules so that the output membership function centers are equal to a scaled sum of the premise linguistic-numeric indices.

Assume that we label the membership functions with linguistic-numeric indices that are integers with zero at the middle (as in our example below). In general, for a fuzzy controller with n inputs and one output, the center of the controller output fuzzy set Y^s membership function would be located at

$$(j + k + ... + l) \times \frac{2}{(N-1)n} \tag{3.3}$$

where $s = j + k + ... + l$ is the index of the output fuzzy set Y^s, $\{j, k, ...l\}$ are the linguistic-numeric indices of the input fuzzy sets, N is the number of membership functions on each input universe of discourse (we assume that there is the same number on each universe of discourse), and n is the number of inputs. This will result in the positioning of a certain number of distinct output membership function centers (the actual number depends on n and N). We choose triangular membership functions for these, with centers given by Equation (3.3), and base widths equal to $\frac{1}{2.5}$.

As a simple example of how to make a rule-base implement a summation operation, assume that we have input membership functions of the form shown in Figure 3.14 but with $N = 5$ and $n = 2$ and effective universes of discourse $[-1, +1]$. In this case Equation (3.3) is given by

$$(j + k)\frac{1}{4}$$

and will result in the rule-base shown in Table 3.10, where the body of the table represents the centers of nine distinct output membership function centers (we assume that their base widths are equal to 0.5 so that they are uniformly distributed on the output universe of discourse).

TABLE 3.10 Rule Table Created for Copying a Linear Controller

Output center		"Input 2" j index				
		-2	-1	0	1	2
"Input 1" k index	-2	-1	-0.75	-0.5	-0.25	0
	-1	-0.75	-0.5	-0.25	0	0.25
	0	-0.5	-0.25	0	0.25	0.5
	1	-0.25	0	0.25	0.5	0.75
	2	0	0.25	0.5	0.75	1

In this case we know that our fuzzy system is normalized (i.e., its effective universe of discourse for the inputs and output are $[-1, +1]$). Also, the fuzzy system will act like a summation operation. All that remains is to explain how to pick the scaling gains so that the fuzzy system implements a weighted sum.

The basic idea in specifying the scaling gains $g_0, \ldots, g_4$ is that for "small" controller inputs (e_i) the local slope (about zero) of the input-output mapping representing the controller should be similar to the LQR gains (i.e., the k_i). We know that by changing the g_i we change the slope of the nonlinearity. Increasing g_i, $i = 1, 2, \ldots, n$ causes the "gain" of the fuzzy controller to increase for small signals (recall the discussions from Chapter 2, Section 2.4.1 on page 78). Increasing g_0, we proportionally increase the "gain" of the fuzzy system. Hence, the approximate gain on the i^{th} input-output pair is $g_i g_0$, so to copy the k_i gains of the state-feedback controller choose

$$g_i g_0 = k_i$$

We can select all the scaling gains via this formula. Recall that the LQR gains are $k_1 = -0.9$, $k_2 = -1.1$, $k_3 = -9.2$, and $k_4 = -0.9$. Transformation of the LQR gains into the scaling gains of the fuzzy system is achieved according to the following simple scheme:

- Choose the controller input that most greatly influences plant behavior and overall control objectives; in our case, we choose the pendulum position θ_1. Next, we specify the operating range of this input (e.g., the interval $[-0.5, +0.5]$ radians, for which the corresponding normalizing input gain $g_3 = 2$).
- Given g_3, the output gain of the fuzzy controller is calculated according to $g_0 = \frac{k_3}{g_3} = -4.6$.
- Given the output gain g_0, the remaining input gains can be calculated according to $g_j = \frac{k_j}{g_0}$, where $j \in \{1, 2, 3, 4\}$, $j \neq i$ (note that $i = 3$). For $g_0 = -4.6$, the input gains g_1, g_2, g_3, and g_4 are 0.1957, 0.2391, 2, and 0.1957, respectively.

The resulting (nonnormalized) input universes of discourse are shown in Figure 3.14.

Experimental Results

If the resulting fuzzy controller that was designed based on the LQR is implemented, we get similar results to the LQR, so we do not include them here. Instead, we will pursue the idea of shaping the nonlinearity induced by the fuzzy controller so that it will be able to perform better than the LQR for the case where a sloshing liquid is added to the endpoint of the pendulum.

The fuzzy controller is a parameterized nonlinearity that can be tuned in a variety of ways. For instance, in Chapter 2 we explained how the output centers can be specified according to a nonlinear function to shape the nonlinearity. Such shaping of the fuzzy controller nonlinearity represents yet another area where intuition (i.e., knowledge about how best to control the process) may be incorporated into the design process. In order to preserve behavior in the "linear" region (i.e., the region near the origin) of the fuzzy controller that we designed using the LQR gains, but at the same time provide a smooth transition from the linear region to its extensions (e.g., regions of saturation), we choose an arctangent-type mapping of the output membership function centers to achieve this rearrangement. Because of the slope of such a mapping near the origin, we expect the fuzzy controller to behave somewhat like the LQR when the states are near the process equilibrium; however, for our particular chosen arctan-type function, we do not expect it to be exactly the same since this warping of the fuzzy controller nonlinearity with the function on the output centers actually changes the slope on the nonlinearity compared to the LQR near the origin. The rationale for this choice of raising the gain near zero will become clear below when we test the fuzzy controller for a variety of conditions on the experimental test bed.

For comparative purposes, we first consider the nominal system—that is, the pendulum alone with no added weight or disturbances. With the pendulum initialized at its hanging position, the swing-up control was tuned to give the best swing-up response (we left K_p the same as for the LQR case but set $\Gamma = 1.71$). The only tuning required for the fuzzy control scheme in transferring it from simulation to implementation was in adjusting the value for g_3 upward to improve performance (recall that the gain g_3 is critical in that it essentially determines the other scaling gains).

Figure 3.15 shows the results for the fuzzy controller on the laboratory apparatus. The response is comparable to that of the LQR controller (compare Figure 3.15 to Figure 3.12 on page 146) in terms of the ability of the controller to balance the pendulum in the vertical up position. Although some oscillation is noticed in the controller output, any difference in the ability to balance the pendulum is only slightly discernible in viewing the operation of the system. (This oscillation on the controller input arises from our use of the arctan-type function since it raises the gain of the controller near zero.)

As a final evaluation of the performance of the fuzzy controller, and to show why we employ the arctan-type function, we illustrate how it performs when a container half-filled with water is attached to the pendulum endpoint. This essentially gives a "sloshing-liquid" effect when the pendulum reaches the balanced position. In

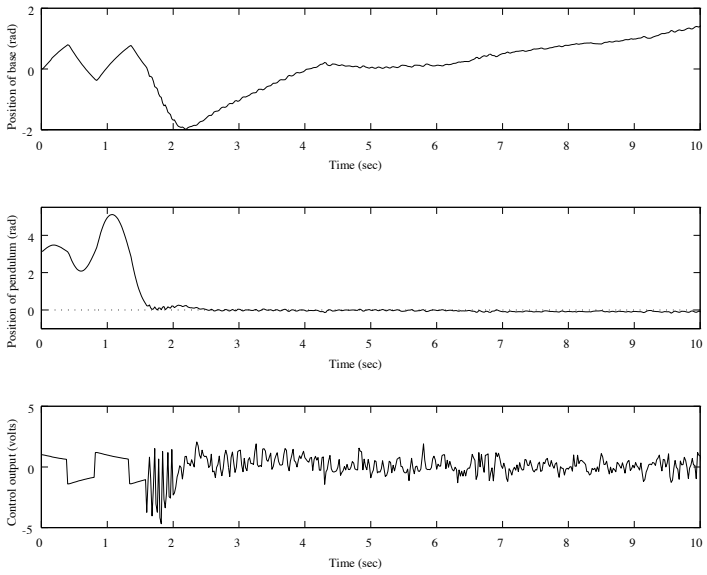


FIGURE 3.15 Direct fuzzy control on the *nominal* rotational inverted pendulum system (figure taken from [235], © IEEE).

addition, the added weight shifts the pendulum center of mass away from the pivot point; as a result, the natural frequency of the pendulum decreases. Furthermore, the effect of friction becomes less dominant because the inertia of the pendulum increases. These effects obviously come to bear on the balancing controller performance, but also significantly affect the swing-up controller as well.

With the sloshing liquid added to the pendulum endpoint, the LQR controller (and, in fact, other linear control schemes we implemented on this system) produced an unstable response and was unable to balance the pendulum, so we do not show their responses here. Of course, the linear control schemes can be tuned to improve performance for the perturbed system, at the expense of degraded performance for the nominal system. Moreover, it is important to note that tuning of the LQR type controller is difficult and ad hoc without additional modeling to account for the added dynamics. Such an attempt on this system produced a controller with stable but poor performance.

It is interesting to note, however, that the fuzzy controller was able to maintain stability in the presence of the additional dynamics and disturbances caused by the sloshing liquid, without tuning. These results are shown in Figure 3.16, where some degradation of controller performance is apparent. Basically, due to the added flexibility in tuning the fuzzy controller nonlinearity, we are able to make it behave similarly to the LQR for the nominal case, but also make it perform reasonably well for the case where the sloshing-liquid disturbance is added. Moreover, there is nothing mystical about the apparent "robustness" of the fuzzy controller: The

shaping of the nonlinearity near zero with the arctan-type function provides a higher gain that counteracts the effects of the sloshing liquid.

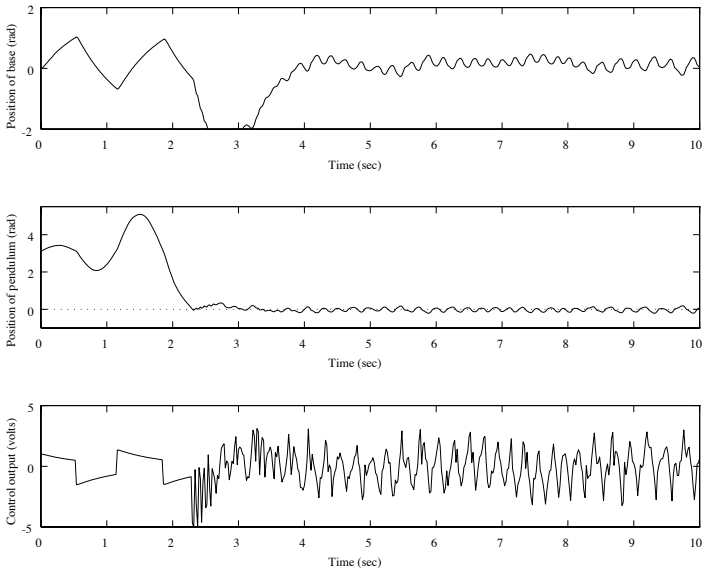


FIGURE 3.16 Direct fuzzy control on the rotational inverted pendulum with sloshing liquid at its endpoint (figure taken from [235], © IEEE).

In Chapter 6 we will show how to design an adaptive fuzzy controller that can automatically reshape its control surface to compensate for endpoint disturbances. This controller will try to optimize its own performance for both the nominal and added-weight cases; we will demonstrate how it will improve the performance of the direct fuzzy controller.

3.5 Machine Scheduling

The flexible manufacturing system (FMS) that we consider in this case study is a system composed of several machines, such as the one shown in Figure 3.17. The system processes several different part-types (indicated by Pi, i= $1, 2, 3$ in Figure 3.17). Each part-type enters the system at a prespecified rate and is routed in the system through a sequence of machines (indicated by Mi, i= $1, 2, ..., 6$ in Figure 3.17) over the transportation tracks (the arrows in Figure 3.17). A part-type may enter the same machine more than once for processing (i.e., the FMS is "nonacyclic"). The length of processing time for each part-type at each machine is also prespecified. The same part-type may have different processing times for the same machine at different visits—that is, a machine may process a part-type longer

at its first visit than at its second. Each part that arrives at a machine is stored in a buffer until the machine is ready to process the part. There are prespecified "set-up times" (delays) when the machine switches from processing one part-type to another. Each scheduler on each machine tries to minimize the size of the "backlog" of parts by appropriately scheduling the sequence of parts to be processed. The goal is to specify local scheduling policies that maximize the throughput of each part-type and hence minimize the backlog and the overall delay incurred in processing parts through the FMS.

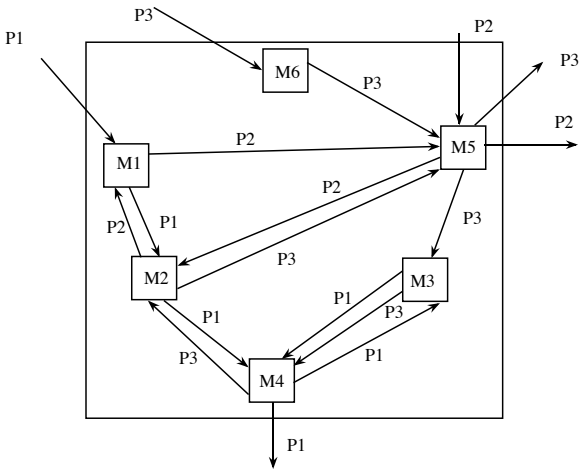


FIGURE 3.17 Example flexible manufacturing system.

In this section we focus on showing how to design a fuzzy controller (scheduler) for a single machine. We use simulations to illustrate that its performance is comparable to conventional scheduling policies. We note that the fuzzy scheduler we develop here is quite different from the ones shown in the two previous case studies. This case study helps to show how fuzzy controllers can be used in nontraditional control problems as general decision makers.

3.5.1 Conventional Scheduling Policies

Figure 3.18 illustrates a single machine that operates on P different part-types. The value of d_p represents the arrival rate of part-type p, and τ_p represents the amount of time it takes to process a part of type p. Parts of type p that are not yet processed by the machine, are stored in buffer b_p. The single machine can process only one part at a time. When the machine switches processing from one part-type p to another part-type p', it will consume a set-up time $\delta_{p,p'}$. For convenience, we will assume that all the set-up times are equal to a single fixed value δ.

If a scheduling policy does not appropriately choose which part to process

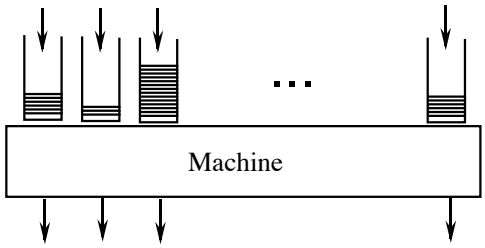


FIGURE 3.18 Single machine with P part-types.

next, the buffer levels of the parts that are not processed often enough may rise indefinitely high, which can result in buffer overflow. To avoid that problem, the machine must have a proper scheduler (controller). In addition to keeping the buffer levels finite, the scheduler must also increase the *throughput* of each part-type, and decrease the buffer levels (i.e., decrease the *backlog*).

Scheduling Policies

A block diagram of a single machine with its controller (scheduler) is shown in Figure 3.19. The inputs to the scheduler are the buffer levels x_p of each part-type. The output from the scheduler is $p*$, which represents the next part-type to process. In order to minimize the idle time due to set-ups, the machine will *clear* a buffer before it starts to process parts from another buffer. There are three clearing policies proposed in [168]: (1) clear largest buffer (CLB), (2) clear a fraction (CAF), and (3) an unnamed policy in Section IV of [168], which we will refer to as "CPK," after the authors, Perkins and Kumar.

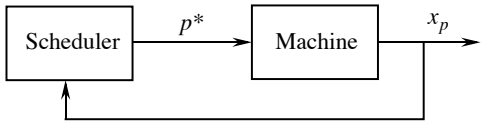


FIGURE 3.19 Machine with its controller (scheduler).

Let $x_p(T_n)$ represent the buffer level of b_p at T_n, the time at which the scheduler selects the next buffer of part-type $p*$ to clear. Let γ_p be any positive weighting factors (throughout this case study, we set the γ_p to 1 so that the "AWBL," to be defined below, is "average work"). Each of the three clearing policies is briefly described as follows:

1. *CLB:* Select $p*$ such that $x_{p*}(T_n) \geq x_p(T_n)$ for all p (i.e., select the buffer to process that has the highest number of parts in it).

2. *CAF:* Select $p*$ such that

$$x_{p*}(T_n) \geq \epsilon \sum_{p=1}^{P} x_p(T_n)$$

where ϵ is a small number, often set to $\frac{1}{P}$ (i.e., when $\epsilon = \frac{1}{P}$, select any buffer to process that has greater than the average number of parts in the buffers).

3. *CPK:* Select $p*$ such that

$$p* = \arg\max_p \left\{ \frac{x_p(T_n) + \delta d_p}{d_p \sqrt{\gamma_p \rho_p^{-1}(1-\rho_p)}} \right\}$$

where $\rho_p = d_p \tau_p$.

In addition to these clearing policies, there exist many other policies that are used in FMS (e.g., first-come first-served (FCFS)).

Machine Properties

A single machine is "stable" if the buffer level for each part-type is bounded. In this case there exists $m_p > 0$, $p = 1, 2, \ldots, P$, such that

$$\sup_t x_p(t) \leq m_p < +\infty \quad for\ p = 1, 2, \ldots, P$$

A necessary condition for stability is that the *machine load* $\rho = \sum_{p=1}^{P} \rho_p < 1$ where $\rho_p = d_p \tau_p$. For the single-machine case, the authors in [168] prove that all three policies described above cause the machine to be stable.

There are various ways to measure the performance of a scheduling policy. We can measure the average delays incurred when a part is processed in the machine. We can also measure the maximum value of each buffer level. The performance criterion proposed in [168] is a quantity called the average weighted buffer level (AWBL), defined as follows:

$$\text{AWBL} = \liminf_{t \to \infty} \frac{1}{t} \int_0^t \left[\sum_p \gamma_p \tau_p x_p(s) \right] ds$$

For any stable scheduling policy, the average weighted buffer level has a lower bound (LB), defined in [168] as follows:

$$LB = \frac{\delta \left[\sum_p \sqrt{\gamma_p \rho_p \ (1-\rho_p)} \right]^2}{2(1-\rho)}.$$

Let $\eta = \frac{AWBL}{LB}$ be a measure of how close a scheduling policy is to optimal. An optimal scheduling policy has η equal to 1; any scheduling policy has $\eta \geq 1$. To compute the value of AWBL, we will of course have to choose some finite value of t to terminate our simulations.

Stabilizing Mechanism

The universally stabilizing supervisory mechanism (USSM) introduced in [100] is a mechanism that is used to govern any scheduling policy. There are two sets of parameters employed by the mechanism for the single machine—namely, γ and z_p, where it must be the case that

$$\gamma > \frac{\sum_b \max_{b'} \delta_{b',b}}{1-\rho}$$

and z_p can be chosen arbitrarily. The single machine will process parts of type p for exactly $\gamma d_p \tau_p$ units of time unless it is cleared first (if a part is currently being processed when this amount of time is up, the processing on this part is finished). Once the machine takes $\gamma d_p \tau_p$ units of time to process parts of type p or the parts of type p are cleared before $\gamma d_p \tau_p$ elapses, the machine will schedule another part to be processed next. In addition, the USSM has a first-in-first-out queue Q. When a buffer level x_p exceeds z_p, and the buffer is not being processed or set up, that buffer will be placed into Q. When there is some buffer in the queue overruling the scheduling policy, the next buffer scheduled to be processed is the first buffer in the queue. Once that first buffer is processed, it leaves the queue, then any remaining buffers in the queue are processed. Hence, the USSM stabilizes any scheduling policy by truncating long production runs and by giving priority to buffers that become excessively high. Note that x_p is not exactly bounded by z_p since x_p can still increase while it is listed in the queue. However, x_p is affected by z_p. The larger z_p is, the larger the maximum of x_p tends to be. Also, note that if the system is already stable (i.e., without the USSM) and the values of γ and z_p are large enough, the mechanism will not be invoked.

3.5.2 Fuzzy Scheduler for a Single Machine

In this section we will show how to perform scheduling via a fuzzy scheduler. The fuzzy scheduler is designed to be a clearing policy just as CLB, CAF, and CPK are. There is no guarantee of stability when operating by itself; therefore, the fuzzy scheduler is always augmented with the USSM.

As for the conventional scheduling policies CLB, CAF, and CPK, the inputs to the fuzzy scheduler policy are the buffer levels x_p. The output of the fuzzy scheduler is simply an index $p*$ indicating which one of the buffers will be processed next. The universe of discourse for each x_p is $[0,\infty)$. The universe of discourse of each x_p has several fuzzy sets. The membership function for each fuzzy set is triangular except at the extreme right, as shown in Figure 3.20. Figure 3.20 shows the membership functions μ for the case where the universe of discourse for x_p has three fuzzy sets. These fuzzy sets, indexed as 1, 2, and 3, indicate how "small," "medium," and "large," respectively, the value of x_p is. If the buffer level x_p exceeds M_p, the value

of x_p is assumed to be M_p by the fuzzy scheduler, where M_p must be predetermined. We will call this parameter M_p the *saturation value* of the fuzzy scheduler for x_p and will use M_p as a tuning parameter.

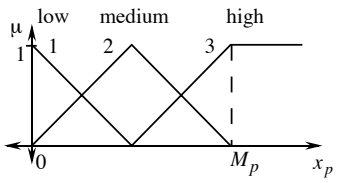


FIGURE 3.20 Three membership functions for x_p.

Table 3.11 shows a rule-base of a fuzzy scheduler for a single machine that has 3 part-types using the fuzzy sets shown in Figure 3.20. In each rule, I_{x_p} represents the index of the fuzzy set and J represents the part-type that is selected by the rule. Then, for instance, rule number 2 takes on the form

$$\textbf{If } x_1 \text{ is small } \textbf{and } x_2 \text{ is small } \textbf{and } x_3 \text{ is medium } \textbf{Then } p* = 3$$

In other words, if the buffer levels of b_1 and b_2 are small and the buffer level of b_3 is medium then process part-type 3. The part of type J that is selected in each rule has buffer level x_J that falls into a fuzzy set that has index I_{x_J} the largest compared to the other indices. In some rules, there are indices of fuzzy sets of several part-types that have equal largest value. In these cases, one of these part-types is selected arbitrarily in our rule-base. For example, the first rule in Table 3.11 is fixed to select part-type 1 even though the fuzzy set indices of all part-types in the rule are equal to 1. Therefore, this rule is *biased* toward part-type 1. We note that our fuzzy scheduler essentially "fuzzifies" the operation of the CLB policy; however, due to the interpolation inherent in the implementation of the fuzzy scheduler it will behave quite differently from the conventional CLB (as the simulation results below indicate).

Throughout the simulation studies in the next subsection, if we use more fuzzy sets on the universe of discourse we will utilize a similar structure for the rule-base (i.e., uniformly distributed and symmetric membership functions). The output universe of discourse (the positive integers) has P membership functions denoted by μ_p where for each $p \in \{1, 2, ..., P\}$, $\mu_p(i) = 1$ for $i = p$ and $\mu_p(i) = 0$ for $i \neq p$ (i.e., singletons). We use singleton fuzzification, minimum for the premise and implication, and max-defuzzification to pick $p*$, given the rule-base and particular values of x_p.

For P buffers and m fuzzy sets, the size of memory needed to store the rules is on the order of P^m; hence, the CLB, CPK, and CAF policies are simpler than the fuzzy scheduler. We will, however, show that with the use of this more complex scheduler we can get enhanced performance in some cases.

TABLE 3.11 Rule-Base of a Fuzzy Scheduler with 3 Inputs and 3 Fuzzy Sets on Each Universe of Discourse

Rule No.	I_{x_1}	I_{x_2}	I_{x_3}	J
1	1	1	1	1
2	1	1	2	3
3	1	1	3	3
4	1	2	1	2
5	1	2	2	2
6	1	2	3	3
7	1	3	1	2
8	1	3	2	2
9	1	3	3	2
10	2	1	1	1
11	2	1	2	1
12	2	1	3	3
13	2	2	1	1
14	2	2	2	1
15	2	2	3	3
16	2	3	1	2
17	2	3	2	2
18	2	3	3	3
19	3	1	1	1
20	3	1	2	1
21	3	1	3	3
22	3	2	1	1
23	3	2	2	1
24	3	2	3	1
25	3	3	1	1
26	3	3	2	1
27	3	3	3	3

It is possible to expand the fuzzy scheduler to use the information about arrival rates, processing times, and set up times also. There may be significant improvements in performance if this information is represented with the control rules; however, the memory size can significantly increase too. In the interest of ensuring that the fuzzy scheduler will be implementable in real time we did not present this variation in this case study.

3.5.3 Fuzzy Versus Conventional Schedulers

Next, we simulate a single machine that uses CLB, CAF, CPK, and the fuzzy scheduler so that we can compare their performance. The machine parameters are as follows: $d_1 = 7$, $d_2 = 9$, $d_3 = 3$, $\tau_1 = 1/100$, $\tau_2 = 1/51$, $\tau_3 = 1/27$, and $\delta = 1$. Figures 3.21 and 3.22 show the plots of the buffer levels of a single machine with three part-types for the first 10 production runs (a production run is defined as

setting up for and processing all the parts in a buffer) and the last 30 production runs when CPK and the fuzzy scheduler are used (note that CLB and CAF did not perform as well, so we do not include their plots). The parameters $M_1 = 35$, $M_2 = 35$, and $M_3 = 12$ are selected based on the maximum value x_p obtains when the CPK policy is used. Note that the first 10 production runs of the fuzzy scheduler are very different from CPK. However, for large values of t they are quite similar but not exactly the same, as indicated by the last 30 production runs when CPK and the fuzzy scheduler are used. Even though the buffer levels are maintained at nearly the same heights, the periodic sequence of scheduling the part-types by CPK is $1, 3, 2, 1, 3, 2, \ldots$, whereas the sequence by the fuzzy scheduler is $1, 2, 3, 1, 2, 3, \ldots$.

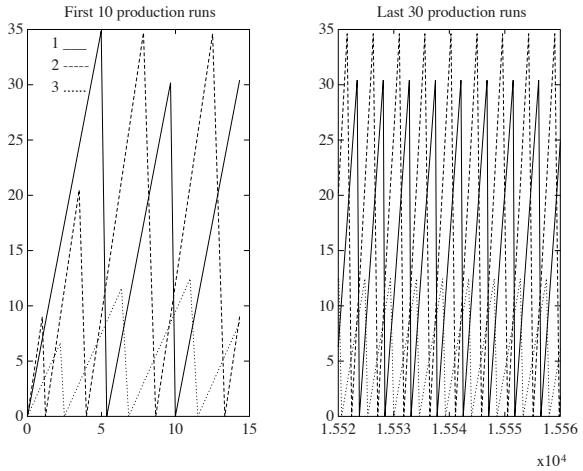


FIGURE 3.21 Buffer levels using the CPK scheduling policy (figure taken from [4], © IEEE).

Among the three schedulers—namely, CLB, CAF, and CPK—CPK often yields the best performance—that is, its η is closest to one [168]. The performance of the fuzzy scheduler is compared to that of CLB, CAF, and CPK for several single machines below. The number of fuzzy sets is set to 3, 5, and 7 for each universe of discourse x_p, so as to observe how the number of fuzzy sets can affect the performance of the fuzzy scheduler. The first two machines are chosen from Section IV of [168].

Machine 1: $d_1 = 7$, $d_2 = 9$, $d_3 = 3$, $\tau_1 = 1/100$, $\tau_2 = 1/51$, $\tau_3 = 1/27$, $\rho = 0.35758$.

- CLB: $\eta = 1.0863484$
- CAF: $\eta = 1.2711257$

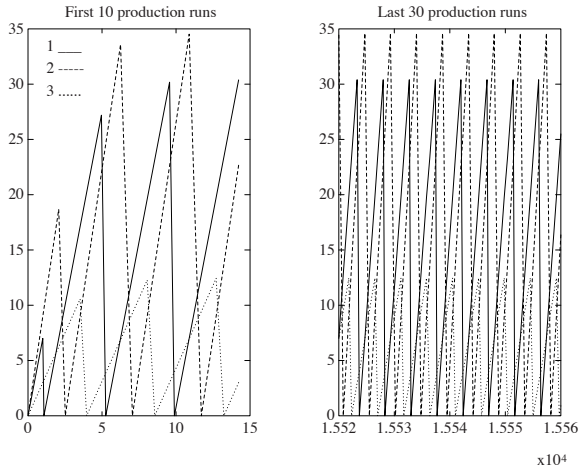


FIGURE 3.22 Buffer levels using the fuzzy scheduler (figure taken from [4], © IEEE).

- CPK: $\eta = 1.0262847$
- Fuzzy scheduler: $M_1 = 35$, $M_2 = 35$, $M_3 = 12$; $\gamma = 34.0$, $z_1 = 30$, $z_2 = 30$, $z_3 = 30$
  For 3 fuzzy subsets, $\eta = 1.0263256$
  For 5 fuzzy subsets, $\eta = 1.0262928$
  For 7 fuzzy subsets, $\eta = 1.0262928$

These simulations show that a fuzzy scheduler can perform nearly as well as can CPK. Note also that we cannot significantly improve η by simply increasing the number of fuzzy subsets for the same M_p (for this machine).

Machine 2: $d_1 = 18$, $d_2 = 3$, $d_3 = 1$, $\tau_1 = 1/35$, $\tau_2 = 1/7$, $\tau_3 = 1/20$, $\rho = 0.99286$.

- CLB: $\eta = 1.1738507$
- CAF: $\eta = 1.179065$
- CPK: $\eta = 1.0017406$
- Fuzzy scheduler: $M_1 = 3375$, $M_2 = 626$, $M_3 = 665$; $\gamma = 1000.0$, $z_1 = 5000$, $z_2 = 5000$, $z_3 = 5000$.
  For 3 fuzzy subsets, $\eta = 1.0027945$
  For 5 fuzzy subsets, $\eta = 1.0027945$
  For 7 fuzzy subsets, $\eta = 1.0013173$

These simulations show that with the machine load closer to one, the fuzzy

scheduler can work even better than CPK provided that there are enough fuzzy sets on the input space. Next, we create a new machine that has a lower machine load, and compare the performance of the scheduling policies.

Machine 3: $d_1 = 3.5$, $d_2 = 4.5$, $d_3 = 1.5$, $\tau_1 = 1/100$, $\tau_2 = 1/51$, $\tau_3 = 1/27$, $\rho = 0.17879$.

- CLB: $\eta = 1.0841100$
- CAF: $\eta = 1.3456014$
- CPK: $\eta = 1.0306833$
- Fuzzy scheduler: $M_1 = 23.6$, $M_2 = 25.1$, $M_3 = 5.6$; $\gamma = 100.0$, $z_1 = 5000$, $z_2 = 5000$, $z_3 = 5000$.
  For 3 fuzzy subsets, $\eta = 1.0307992$
  For 5 fuzzy subsets, $\eta = 1.0319630$
  For 7 fuzzy subsets, $\eta = 1.0306972$
- Fuzzy scheduler with 3 fuzzy subsets; $M_1 = 50$, $M_2 = 50$, $M_3 = 20$; $\gamma = 100.0$, $z_1 = 5000$, $z_2 = 5000$, $z_3 = 5000$: $\eta = 1.2273009$

These simulations show that the fuzzy scheduler cannot perform any better than CPK when the machine load is small for this machine. Also note that if the parameters M_p are not set properly, the performance of the fuzzy scheduler can degrade.

Our experience in simulation above shows that it is possible to tune the fuzzy scheduler by choosing the values of M_p and the fuzzy sets to minimize η. We have used the following procedure to tune the fuzzy scheduler to get smaller η: (1) use i fuzzy sets and set the M_p all to unity, (2) run a simulation, (3) replace M_p with the maximum buffer levels obtained in x_p and rerun the simulation, and (4) repeat as necessary with $i + 1$ fuzzy sets, $i + 2$ fuzzy sets, and so on. Using this tuning approach for the above machine we find that for 3-buffer machines the results are as good as those of CPK, and for some 5-buffer machines the tuning method converges to a good result, even though the result is not quite as good as that of CPK. Note that our experiences in tuning allowed us to develop the on-line adaptive fuzzy scheduler technique that is studied in Chapter 6.

3.6 Fuzzy Decision-Making Systems

A fuzzy controller is constructed to make decisions about what the control input to the plant should be given processed versions of the plant outputs and reference input. It is a form of artificial (i.e., nonbiological) decision-making system. Decision-making systems find wide application in many areas, not only the ones that have been traditionally studied in control systems. For instance, the machine scheduling case study of the previous section shows a nontraditional application of feedback control where a fuzzy system can play a useful role as a decision-making system.

There are many other areas in which fuzzy decision-making systems can be used including the following:

- *Manufacturing:* Scheduling and planning materials flow, resource allocation, routing, and machine and equipment design.
- *Traffic systems:* Routing and signal switching.
- *Robotics:* Path planning, task scheduling, navigation, and mission planning.
- *Computers:* Memory allocation, task scheduling, and hardware design.
- *Process industries:* Monitoring, performance assessment, and failure diagnosis.
- *Science and medicine:* Medical diagnostic systems, health monitoring, and automated interpretation of experimental data.
- *Business:* Finance, credit evaluation, and stock market analysis.

This list is by no means exhaustive. Virtually any computer decision-making system has the potential to benefit from the application of fuzzy logic to provide for "soft" decisions when there is the need for decision making under uncertainty.

In this section we focus on the design of fuzzy decision-making systems for problems other than feedback control. We begin by showing how to construct fuzzy systems that provide warnings for the spread of an infectious disease. Then we show how to construct a fuzzy decision making system that will act as a failure warning system in an aircraft.

3.6.1 Infectious Disease Warning System

In this section we study a biological system where a fuzzy decision-making system is used as a warning system to produce alarm information. To model a form of biological growth, one of Volterra's population equations is used. A simple model representing the spread of a disease in a given population is given by

$$\frac{dx_1(t)}{dt} = -ax_1(t) + bx_1(t)x_2(t) \tag{3.4}$$

$$\frac{dx_2(t)}{dt} = -bx_1(t)x_2(t) \tag{3.5}$$

where $x_1(t)$ is the density of the infected individuals, $x_2(t)$ is the density of the noninfected individuals, $a > 0$, and $b > 0$. These equations are only valid for $x_1(t) \geq 0$ and $x_2(t) \geq 0$. The initial conditions $x_1(0) \geq 0$ and $x_2(0) \geq 0$ must also be specified.

Equation (3.5) intuitively means that the noninfected individuals become infected at a rate proportional to $x_1(t)x_2(t)$. This term is a measure of the interaction between the two groups. The term $-ax_1(t)$ in Equation (3.4) represents the rate at which individuals die from disease or survive and become forever immune. The term

$bx_1(t)x_2(t)$ in Equation (3.4) represents the rate at which previously noninfected individuals become infected.

Here, we design a fuzzy system to produce alarms if certain conditions occur in the diseased population—that is, a simple warning system. The fuzzy system uses $x_1(t)$ and $x_2(t)$ as inputs, and its output is an indication of what type of warning condition occurred along with the certainty that this warning condition has occurred. To specify the types of alarms we would like the fuzzy system to output, we first begin by using conventional (nonfuzzy) logic and "decision regions" to specify the alarms. In particular, we would like indications of the following alarms:

1. *"Warning:* The density of infected individuals is unsafe"; this occurs if $x_1(t) > \alpha_1$ where α_1 is some positive real number (here $x_1(t) > \alpha_1$ specifies a "decision region" for where we could like the warning to be given).

2. *"Caution:* The density of infected individuals is unsafe, and the number of infected individuals is greater than the number of noninfected individuals"; this occurs if $x_1(t) > \alpha_1$ and $x_1(t) \geq x_2(t) + \alpha_2$ but $x_1(t) < x_2(t) + \alpha_3$, where α_2 and α_3 are positive real numbers such that $\alpha_2 < \alpha_3$.

3. *"Critical:* The density of infected individuals is unsafe, and the number of infected individuals is much greater than the number of noninfected individuals"; this occurs if $x_1(t) > \alpha_1$ and $x_1(t) \geq x_2(t) + \alpha_3$.

The three alarms represent certain warnings characterized by the decision regions shown in Figure 3.23. The darkest region plus the other lighter shaded regions represent the first warning's decision region, the slightly lighter one represents the second warning, and the lightest shaded represents the third warning.

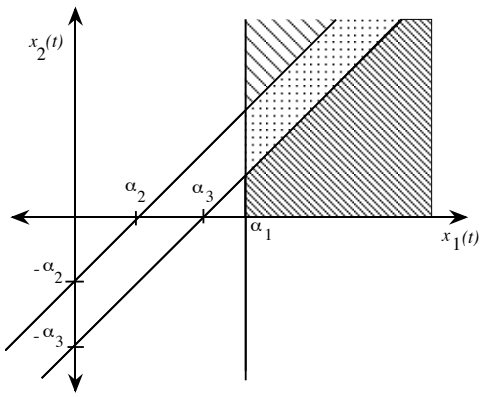


FIGURE 3.23 Decision regions for the biological system (figure taken from [164], © IEEE).

We could simply use the above inequalities to implement a system that would take as inputs $x_1(t)$ and $x_2(t)$ and output an indication of which warning above has occurred. Then, as the differential equation evolves, the values of $x_1(t)$ and $x_2(t)$ change and different warning conditions will hold (when none hold, there is no warning). Here, we will implement a fuzzy decision-making system by using fuzzy logic to "soften" the decision boundaries. We do this since we are not certain about the positions of these boundaries and since we would like an earlier indication when we are near a boundary and therefore near having another condition begin to hold.

To construct the fuzzy system, we would like to implement fuzzy versions of the following three rules:

1. **If** $x_1(t) > \alpha_1$ **Then** warning is "Warning"
2. **If** $x_1(t) > \alpha_1$ **and** $x_1(t) \geq x_2(t) + \alpha_2$ **and** $x_1(t) < x_2(t) + \alpha_3$ **Then** warning is "Caution"
3. **If** $x_1(t) > \alpha_1$ **and** $x_1(t) \geq x_2(t) + \alpha_3$ **Then** warning is "Critical"

While the rules we used in the fuzzy controllers in Chapter 2 were different, we can still use fuzzy logic to quantify these rules. First, we need to quantify the meaning of each of the premise terms. Then we will be able to use the standard fuzzy logic approach to quantify the meaning of the "**and**" in the premises.

First, notice that the premise term $x_1(t) > \alpha_1$ can be quantified with the membership function shown in Figure 3.24 (study the shape of the membership function carefully and convince yourself that this is the case). The membership functions in Figure 3.25 quantify the meaning of $x_1(t) \geq x_2(t) + \alpha_2$ and $x_1(t) < x_2(t)+\alpha_3$. Notice that we have made the positioning of the membership functions in Figure 3.25 dependent on the value of $x_2(t)$; hence, to compute the certainty of the statement $x_1(t) \geq x_2(t)+\alpha_2$, we would first position the membership function with the given value of $x_2(t)$, then we would compute the certainty of the statement (i.e., its membership value). You can avoid this shifting of the membership functions by simply making the two inputs to the fuzzy system $x_1(t)$ and $x_1(t) - x_2(t)$ rather than $x_1(t)$ and $x_2(t)$ (since then you can use a similar characterization to that which was used for the first alarm—why?). We can quantify the third alarm in a similar way to the second one.

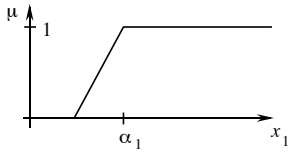


FIGURE 3.24 Membership function representing $x_1(t) > \alpha_1$.

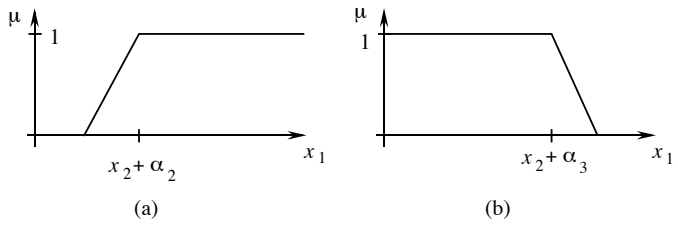


FIGURE 3.25 Membership functions representing (a) $x_1(t) \geq x_2(t) + \alpha_2$ and (b) $x_1(t) < x_2(t) + \alpha_3$.

Next, we need to use fuzzy logic to quantify the consequents of the three rules. To do this, suppose that we let the universe of discourse for "warning" be the interval of the real line $[0, 10]$. Then we simply use the membership functions shown in Figure 3.26. There, the membership function on the left represents "Warning," the one in the middle represents "Caution," and the one on the right represents "Critical" (note that all of these have finite area). Suppose that we use minimum to quantify the premise and implication, and that we use COG defuzzification (be careful with COG since the output membership functions are not symmetric).

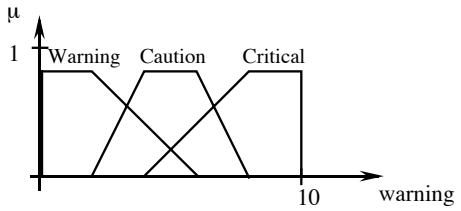


FIGURE 3.26 Membership functions to quantify the consequents.

This completes the definition of the fuzzy warning system for the biological system. We leave it to the reader to simulate the biological system and verify that the fuzzy system will provide the proper values for the output "warning." Note that to interpret the output of the fuzzy system you will want to have a list of the three failures "Warning," "Caution," and "Critical" and their associated certainties of being true. Define the certainty of each warning being true as the minimum certainty of any premise term in the premise of the rule that corresponds to the warning. The output of the fuzzy system, "warning" will also provide a numerical rating of the severity of the warning. In this way the fuzzy system provides both a linguistic and numeric quantification of the warning.

3.6.2 Failure Warning System for an Aircraft

Consumer and governmental demands have provided the impetus for an extraordinary increase in the complexity of the systems that we use. For instance, in automotive and aircraft systems, governmental demands have called for (1) highly accurate air-to-fuel-ratio control in automobiles to meet pollution standards, and (2) highly technological aircraft capable of achieving frequent flights with very little maintenance downtime. Similarly, consumer demands have driven (1) the development of antiskid braking systems for increased stopability, steerability, and stability in driving and (2) the need for increased frequency of commercial flights such that travel must occur under all weather conditions in a timely manner.

While engineers have, in general, been able to meet these demands by enhancing the functionality of high-technology systems, this has been done at the risk of significant failures (it is generally agreed that "the more complex a system is the more likely it is to fail in some way"). For automotive and aircraft systems, some of the failures that are of growing concern include the following:

- Failures and/or degradation of performance of the emissions control systems (failures or degradation leads to a significant increase in the level of pollutants).
- "Cannot duplicate" failures where a failure is detected while the aircraft is in flight that cannot be duplicated during maintenance, which lengthens the downtime.
- Actuator, sensor, and other failures in aircraft systems that cause commercial aircraft crashes in adverse weather conditions.
- A system failure in an integrated vehicle handling, braking and traction control system, which can lead to a loss of control by the driver.

Automotive and aircraft systems provide excellent examples of how failures in high-technology systems can result in catastrophic failures. In addition, the effect of undetected system faults can lead to costly downtime or catastrophic failures in manufacturing systems, nuclear power plants, and process control problems. As history indicates, the probability of some of the system failures listed above is sometimes high. There is then the need for detecting, identifying, and providing appropriate warnings about failures that occur on automobiles, aircraft, and other systems so that corrective actions can be taken before there is a loss of life or other undesirable consequences.

Experience in developing on-line failure warning systems has indicated that there is no uniform approach to solving all problems; solutions are "problem-dependent." This makes the fuzzy system particularly well suited for this application. You simply have to load different knowledge into a fuzzy system for different applications. Next, we look at a simple example of how to construct a fuzzy warning system for an aircraft.

The simple warning system for an aircraft uses the aircraft's measurable inputs and outputs. Suppose the aircraft's input vector u has two components, the elevator δ_e (deg), and thrust δ_t (deg). The output vector y has three components, pitch rate q

(deg/sec), pitch angle θ (deg), and load factor η_z (g). Four aircraft failure modes are considered here. To define the modes, we take the same approach as in the previous section and define decision regions using conventional logic and inequalities. Later, we will soften the decision boundaries and define the fuzzy decision-making system.

To define the decision boundaries, each input and output is discretized into five regions with four boundaries associated with the real number line. For example, the elevator δ_e is discretized as follows:

- Region R_1: $\delta_e \leq \delta_{R_1}$
- Region Y_1: $\delta_{R_1} < \delta_e \leq \delta_{Y_1}$
- Region G: $\delta_{Y_1} < \delta_e \leq \delta_{Y_2}$
- Region Y_2: $\delta_{Y_2} < \delta_e \leq \delta_{R_2}$
- Region R_2: $\delta_e \geq \delta_{R_2}$

where δ_{R_1} and δ_{Y_1} are negative constants with δ_{R_1} larger in magnitude than δ_{Y_1}, and δ_{Y_2} and δ_{R_2} are positive constants with $\delta_{Y_2} \leq \delta_{R_2}$. The G (for Green) region denotes an area of safe operation, the Y_1 and Y_2 (for Yellow) regions denote areas of warning, and the R_1 and R_2 (for Red) regions denote areas of unsafe operation. Suppose that using a similar notation we define such regions for all the other aircraft input and output variables. For simplicity we will then sometimes say that other variables lie in the regions R_1, Y_1, G, Y_2, and R_2 with the understanding that there can be different values of the constants used to define the intervals on the real line.

Using the defined regions for the aircraft inputs and outputs, four failure modes for the aircraft are identified as follows:

1. Load factor is in region R_2.
2. Load factor is in region Y_2.
3. Load factor is in region Y_2 and elevator is in region Y_1
4. (Pitch rate is in Y_1 and Pitch angle is in Y_1) or (Pitch rate is in Y_2 and Pitch angle is in Y_2).

The decision regions for the fourth failure mode are shown as the shaded areas in Figure 3.27 (notice that we use an appropriate notation for the constants that define the boundaries).

The fuzzy system's inputs are the aircraft inputs and outputs, and its outputs are the four failure warnings. Suppose that the output of the fuzzy system is either 1, 2, 3, or 4, representing the failure warning mode, $i = 1, 2, 3, 4$. Now, we want to define a fuzzy system that will give a proper indication of the above failure warning modes. To define the fuzzy system, we use the same approach as in the previous section. We can define rules representing each of the four failure warning modes.

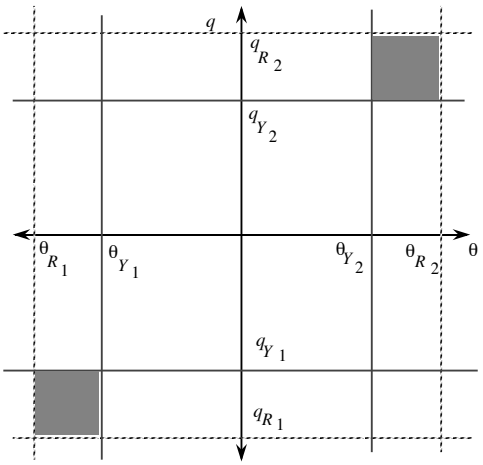


FIGURE 3.27 Decision regions for aircraft failure mode four (figure taken from [164], © IEEE).

These will have the proper logical combinations of the inequalities in the premises, and the consequents will be, for example, "failure warning = 1."

For the first mode (load factor is in region R_2), you can use the same approach as in the last section to specify a membership function to represent the single premise term; the same for the second and third failure warning modes. For the fourth failure warning mode, we can use fuzzy logic in the characterization of the "and" in the premise as we did in the last section. Also, we can use the fuzzy logic characterization of "or" to represent the combination of the two terms in the premise of the rule for the fourth failure warning mode. You can use singletons positioned at $i = 1, 2, 3, 4$ for the i^{th} failure warning mode rule. Then use center-average defuzzification to complete the specification of the fuzzy warning system. We leave the details of constructing this fuzzy decision-making system to the reader.

3.7 Summary

In this chapter we provided an overview of the design methodology for fuzzy control systems and showed how to design fuzzy controllers in the two-link flexible robot and the rotational inverted pendulum case studies. We used the machine control problem and fuzzy decision-making systems to illustrate how the fuzzy system can also be useful in nontraditional control applications. Each problem provided different challenges, and in two of the case studies we showed actual implementation results. In two of the cases we compared the fuzzy controller to conventional approaches, which highlights the advantages and disadvantages of fuzzy control.

Upon completing this chapter, the reader should understand the following:

- The general design methodology for fuzzy controllers.
- How to design a fuzzy controller for a flexible-link robot and how the use of additional, more-detailed knowledge can improve performance (e.g., the uncoupled versus coupled cases) but increases the complexity of the controller (e.g., the number of rules increased).
- How to design a swing-up controller and LQR for balancing for the rotational inverted pendulum, how to use the LQR design to provide the first guess at the fuzzy controller (that may later be tuned), and how to use a nonlinear mapping to set the positions of the output membership function centers.
- How to specify a fuzzy controller that can schedule the processing of parts at a machine and perform at comparable levels to good conventional schedulers.
- How to design fuzzy decision-making systems, particularly for failure warning systems.

Essentially, this is a checklist for the major topics covered in this chapter. The reader should be sure to understand each of the above concepts or approaches before proceeding on to more-advanced chapters, especially the ones on adaptive and supervisory fuzzy control, where the first three case studies examined here are further investigated.

3.8 For Further Study

There are many conference and journal papers that focus on the application of direct fuzzy control—indeed, too many to mention here. Here, we simply highlight a few case studies that are particularly interesting or instructive [125, 91, 21, 35, 25] and refer the interested reader to several books that have focused on industrial applications of fuzzy control, including [240, 137, 175, 206] (these also have extensive lists of references that the interested reader may want to follow up on). Also, there are some recent books [47, 154] and papers (e.g., [218]) that focus on some new design methodologies for fuzzy controllers that the reader may be interested in. One of these is based on sliding-mode control [217], and the other is related to gain-scheduling-type control.

The case study in this chapter on the two-link flexible robot was taken directly from [145, 144]; the interested reader should see those papers (and the references within them) to obtain a more complete treatment of work related to the case study. Since the literature abounds with work on the modeling and control of flexible robots, both from a theoretical (simulation-based) and an experimental point of view, we refer the interested reader to Chapter 8 of [193] for an overview of the literature on conventional approaches. Some studies that are particularly relevant to our case study are in [69, 242, 243].

The case study for the rotational inverted pendulum was taken from [235, 244]. The literature abounds in research and implementations of the linear-translational

inverted pendulum. The approach of using the linear controller to initialize the fuzzy controller that is used for the rotational inverted pendulum was first introduced in [104], where it was used for an aircraft application. It is interesting to note that in [235] it is shown how a fuzzy system can be used to automate the swing-up control so that the manual tuning of the above parameters is not needed even if additional mass is added to the endpoint of the pendulum.

The machine control case study was taken directly from [6]. The work was inspired by the earlier work of P.R. Kumar and his colleagues (see, for example, [168, 100]) on the development of distributed scheduling policies for flexible manufacturing systems. The failure warning systems are fuzzy versions of the ones developed in [164]; for a more detailed study of aircraft failure diagnostic systems, see [161]. Fuzzy decision-making systems are discussed in some more detail in [206, 175].

The motor control design problem in the problems at the end of the chapter is part of a control laboratory at Ohio State University (developed over the years by many people, including Ü. Özgüner, L. Lenning, and S. Brown). The ship steering problem comes from [11] and [112]. The rocket velocity control problem was taken directly from [113]. The design problem on the acrobot was taken directly from [27] and builds directly on earlier work performed by M. Spong and his colleagues, who have focused on the development of conventional controllers for the acrobot. Their work in [190] and [191] serves as an excellent introduction to the acrobot and its dynamics. The dynamics of a simple acrobot are also described in both works; however, a more complete development of the acrobot dynamics may be found in [192]. The base-braking control problem is taken from [75, 66] and was based on years of contracted research with Delphi Chassis Division of General Motors. Previous research on the brake system has been conducted using proportional-integral-derivative (PID), lead-lag, autotuning, and model reference adaptive control (MRAC) techniques [66]. The particular problem description we use for the brakes was taken from [118].

3.9 Exercises

Exercise 3.1 (Simulation of General Fuzzy Systems): Write a program in high-level language that will simulate a general fuzzy controller with the following characteristics:

(a) n inputs and one output (i.e., so that the user can input n).

(b) Triangular membership functions (with appropriately saturated ones at the endpoints of the input universes of discourse).

(c) Gaussian membership functions (with appropriately saturated ones at the endpoints of the input universes of discourse).

(d) Trapezoidal membership functions (with appropriately saturated ones at the endpoints of the input universes of discourse).

(e) The use of product or minimum for representing the premise and implication.

(f) The use of center-average or COG defuzzification.

Exercise 3.2 (Efficient Simulation of Fuzzy Systems): Write a program in high-level language that will simulate a general fuzzy controller with the following characteristics:

- n inputs and one output.
- Triangular membership functions (with appropriately saturated ones at the outermost regions of the input universes of discourse) that are uniformly distributed across the universes of discourse so that there are at most two of them overlapping at any one point.
- The use of minimum for representing the premise and implication.
- The use of COG defuzzification.

Exploit the fact that no more than two membership functions overlap at any one point to make the code as efficient as possible. Use ideas from Chapter 2 where we discuss simulation of fuzzy systems and real-time implementation issues.

Exercise 3.3 (Fuzzy Systems: Computational Complexity): Fuzzy controllers can at times require significant computational resources to compute operations in real-time. Define a "computing step" as the act of performing a basic mathematical operation (e.g., addition, subtraction, multiplication, division, or finding the maximum or minimum of a set of numbers). For the first inverted pendulum controller that we designed in Chapter 2 (i.e., the one using triangular membership functions with $R = 25$ rules), using this measure, determine the number of computing steps that it takes to perform the following operations (assume that you code it efficiently, exploiting the fact that only two membership functions overlap at any point so at most four rules are on):

(a) COG defuzzification—assuming that you are already given the values of the premise membership function certainties.

(b) Center-average defuzzification—assuming that you are already given the values of the premise membership function certainties.

(c) Assume that we switch to using Gaussian membership functions as in Exercise 2.3 on page 102. Does this increase or decrease computational complexity? Why?

Exercise 3.4 (Fuzzy Controller Design Using Linear Controllers): Suppose that you have a PD controller that generates the plant input $u = K_p e + K_d \frac{d}{dt} e$ ($e = r - y$ where r is the reference input and y is the plant output) and that it performs well for small values of its inputs, but that for larger values you happen to know some additional heuristics that can be used to improve performance. To capture this information, suppose that you decide to use a two-input, one-output fuzzy controller. Rather than throwing out all the work you have done to tune the PD gains K_p and K_d, you would like to make the fuzzy controller behave similarly to the PD controller. Suppose that $K_p = 2$ and $K_d = 5$. Design a fuzzy controller that will approximate these same gains for small values of e

and $\frac{d}{dt}e$. Demonstrate that the two are close by providing a three-dimensional plot of the control surfaces for both the PD and the fuzzy controller (note that the PD controller surface looks like a plane in three dimensions).

Exercise 3.5 (Fuzzy Control Design Trade-Offs)*: List all the trade-offs involved in choosing fuzzy versus conventional control and, for the application of your choice, provide a written analysis of whether you think fuzzy control is a viable approach for your problem. Fully support your conclusions. You may choose your own application, but if you do you must fully describe the control problem that you study and provide at least simulation studies to back up your conclusions. Alternatively, you may choose one of the case study examples in this chapter (or one of the design problems) for your analysis.

3.10 Design Problems

Design Problem 3.1 (Inverted Pendulum: Use of a CAD Package): In this problem you will learn to use a CAD package (such as the one available in Matlab) for the development and analysis of fuzzy control systems.

(a) Use a CAD package to solve Exercise 2.3 on page 102.

(b) Use a CAD package to solve Exercise 2.4 on page 103.

(c) Use a CAD package to solve Design Problem 2.1 on page 110.

Design Problem 3.2 (Single-Link Flexible Robot): This problem focuses on the design of a fuzzy controller for a single-link flexible robot. To perform the designs, use the model provided in Section 3.3.1 on page 127 (in particular, Equation (3.1)); hence, the plant input is v_1 and the plant output is θ_1. Command a 90-degree step change in the position to test your closed-loop system. Use the saturation nonlinearities that were provided for the voltage input and link position. The goals are fast slewing with minimal endpoint vibrations and no steady-state tracking error. Use a 20-ms sampling period and discrete time controllers.

(a) Design a fuzzy controller for the single-link flexible robot and evaluate its performance.

(b) Design the best linear controller that you can for the flexible robot and compare its performance to that of the fuzzy controller.

(c) Compare the performance that was obtained in (a) to that obtained in (b). Identify which characteristics of your simulation responses are different from the implementation responses for the two-link robot, and try to provide reasons for these differences.

Design Problem 3.3 (Rotational Inverted Pendulum): This problem focuses on the design of fuzzy controllers for the rotational inverted pendulum that was studied in this chapter. To perform the designs, use the model provided in Section 3.4.1 on page 143. You should seek to obtain performance comparable to that seen in the implementation results for the rotational inverted pendulum.

(a) Design an "energy-pumping" swing-up strategy for the rotational inverted pendulum, and develop a LQR controller for balancing the pendulum. Demonstrate its performance in simulation.

(b) Design a fuzzy controller for balancing the pendulum, and, using the same swing-up strategy as in (a), demonstrate its performance in simulation.

(c) For both (a) and (b), compare the performance that was obtained to that which was found in implementation. Identify characteristics of your simulation responses that are different from the implementation responses, and provide a reason for these differences.

Design Problem 3.4 (Machine Scheduling): Here, we focus on the design of fuzzy schedulers for the machine scheduling problem that was studied in this chapter. To perform the designs, use the model provided in Section 3.5.1 on page 153. Suppose that we define "Machine 4" to have the following characteristics: $d_1 = 1$, $d_2 = 1$, $d_3 = 1/0.9$, $d_4 = 1$, $d_5 = 1$, $\tau_1 = 0.15$, $\tau_2 = 0.2$, $\tau_3 = 0.05$, $\tau_4 = 0.1$, $\tau_5 = 0.2$, $\rho = 0.7055556$ (i.e., it has five buffers).

(a) Develop CLB, CAF, and CPK schedulers, simulate them, and determine the value for η for each of these.

(b) Develop a fuzzy scheduler using the same approach as in the case study. Find the value of η for the cases where 3, 5, and 7 fuzzy sets are used on each input universe of discourse. Be careful to properly tune the values of the M_i and use $\gamma = 100.0$, and $z_i = 30$, $i = 1, 2, 3, 4, 5$. You should tune the M_i so that the fuzzy scheduler performs better than the ones in (a) as measured by η.

Design Problem 3.5 (Motor Control): In this problem we study control of the Pittman GM9413H529 DC motor with a simulated inertial load (aluminum disk). The simulated moment of inertia is small, and is considerably less than the actual motor moment of inertia. The effective gear ratio is 7860:18 (from the motor armature shaft to the actual load); therefore, the reflected load inertia seen by the motor is very small. The equivalent circuit diagram of the DC motor system is shown in Figure 3.28.

The DC motor has a single measurable signal: the motor's rotational velocity. This velocity is sensed using an optical encoder mounted on the shaft of the motor. An optical encoder outputs square wave pulses with a frequency proportional to rotational velocity. The pulses from the encoder are counted by

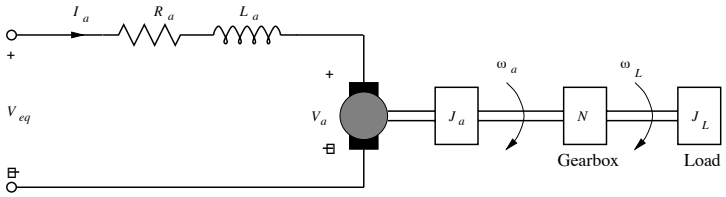


FIGURE 3.28 Equivalent circuit diagram of the DC motor system (figure drawn by Scott C. Brown).

a data-acquisition card's counter/timer, and translated to a rotational velocity of the inertial plate. Pulse width modulation (PWM) is used to vary the input voltage to the motor. PWM varies the duty cycle of a constant magnitude square wave to achieve an approximation of a continuous control input. A diagram of the motor experimental setup is shown in Figure 3.29.

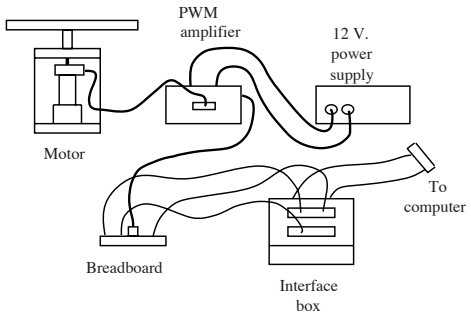


FIGURE 3.29 Motor experimental setup (figure drawn by Scott C. Brown).

The transfer function of the motor can be derived from the following data (taken from the Pittman motor spec sheets for winding 114T32):

$$
\begin{aligned}
R_a &= \text{armature resistance} = 8.33\ \Omega \\
L_a &= \text{armature inductance} = 6.17\ \text{mH} \\
K_e &= \text{back emf constant} = 4.14\ \text{V/krpm} = 3.953 \times 10^{-2}\ \text{V/(rad/s)} \\
K_t &= \text{torque constant} = 5.60\ \text{oz} \cdot \text{in/A} = 0.03954\ \text{N} \cdot \text{m/A} \\
J_a &= \text{armature inertia} = 3.9 \times 10^{-4}\ \text{oz} \cdot \text{in} \cdot \text{s}^2 = 2.75 \times 10^{-6}\ \text{Kg} \cdot \text{m}^2 \\
J_L &= \text{load inertia} = 0.0137\ \text{Kg} \cdot \text{m}^2 \\
J &= \text{total inertia} = \text{motor} + \text{load} = J_a + \frac{J_L}{N^2} = 2.82 \times 10^{-6}\ \text{Kg} \cdot \text{m}^2 \\
N &= \text{Gear ratio} = 7860 : 18
\end{aligned}
$$

The aluminum disk has a radius of 15.24 cm, a thickness of 0.6 cm and a density of 2699 Kg/m^3.

Using these parameters, the following system time constants can be determined:

$$\begin{array}{rcccl} 1/T_e & = & R_a/L_a & = & 1350 \text{ (rad/sec)} \\ 1/T_m & = & \frac{K_e K_t}{R_a J} & = & 66.43 \text{ (rad/sec)} \end{array}$$

Since $L_a \ll \frac{R_a^2 J}{K_e K_t}$,

$$\begin{aligned} G_1(s) = \frac{\omega_a(s)}{V_{eq}(s)} &= \frac{1/K_e}{(1+T_e s)(1+T_m s)} \\ &= \frac{25.3}{(1+\frac{s}{1350})(1+\frac{s}{66.4})} \\ &= \frac{2.27 \times 10^6}{(s+1350)(s+66.4)} \left(\frac{\text{rad/sec}}{\text{V}}\right) \end{aligned}$$

$$\begin{aligned} G_2(s) = \frac{\omega_L(s)}{V_{eq}(s)} = \frac{1}{N} G(s) &= \frac{\omega_L(s)}{V_{eq}(s)} \\ &= \frac{5194}{(s+1350)(s+66.4)} \left(\frac{\text{rad/sec}}{\text{V}}\right) \end{aligned}$$

$$\begin{aligned} G_3(s) = \frac{\omega_L(s)}{V_{eq}(s)} &= \frac{5194 \cdot (\frac{60}{2\pi})}{(s+1350)(s+66.4)} \\ &= \frac{49.60 \times 10^3}{(s+1350)(s+66.4)} \left(\frac{\text{rpm}}{\text{V}}\right) \end{aligned}$$

$G_1(s)$ specifies the transfer function from voltage input to motor speed, while $G_2(s)$ and $G_3(s)$ specify the transfer function from voltage input to load speed (in different units). $G_3(s)$ is the transfer function of interest for this system, as the reference input to track is specified in rpm. Note that the maximum system input is ± 12 volts.

We will study the development of controllers with a focus on implementation; hence, we will develop a discrete model for the simulations. To simulate the system $G_3(s)$, it is converted to a state-space realization, and then a discrete state-space realization (we use the zero order hold (ZOH) method for continuous to discrete-time conversion). Use a sampling period of 0.01 sec. The discrete-time model

$$\begin{aligned} x(k+1) &= A\,x(k) + B\,u(k) \\ y(k) &= C\,x(k) + D\,u(k) \end{aligned}$$

can be simulated where u is the system input, or controller output. Note that this model is a relatively accurate representation of the actual physical experiment in our laboratory, shown in Figure 3.29 (the main difference lies in the presence of more noise in the actual experiment).

We developed and implemented a fuzzy controller that we consider to be only marginally acceptable for the motor and show its response in Figure 3.30. We consider this plot to provide an indication of the type of performance we expect from controller designs for this plant.

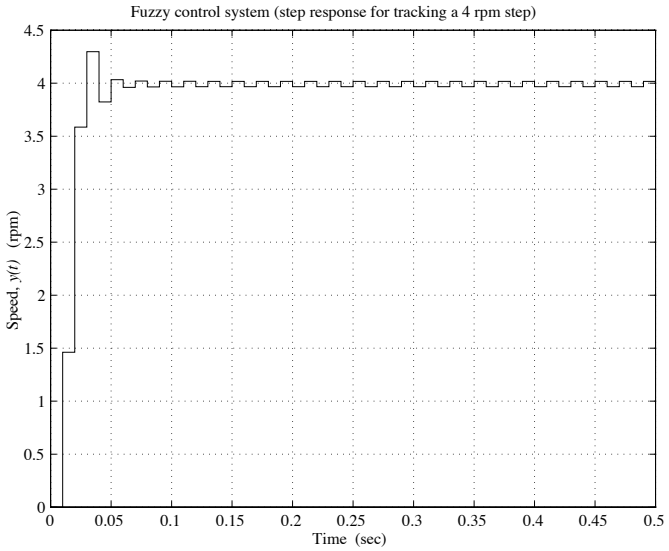


FIGURE 3.30 Motor step response (plot created by Scott C. Brown).

(a) Design a linear controller for the motor, and demonstrate that you can do better than the performance shown in Figure 3.30.

(b) Define a two-input, single-output fuzzy system to control the motor. Use e and $\int_0^t e\,dt$ (use the trapezoidal approximation to integration to approximate $\int_0^t e\,dt$) as inputs to your fuzzy system where error is defined as

$$\text{error} \triangleq \text{reference input} - \text{system output}$$

Use triangular input and output membership functions (with appropriately saturated ones at the outermost edges). Be sure that your fuzzy system output is in the range $[-12.0, 12.0]$. Simulate the system with the fuzzy controller by tracking a 4-rpm step input.

(c) Tune your fuzzy system to obtain a better step response. Try changing the number of input and output membership functions, as well as the gains multiplying the fuzzy controller inputs and outputs. (Remember, your output should saturate at ± 12 volts.) Can you do better than the response shown in Figure 3.30? Obtain plots of the input and output membership functions, and simulated step response for your best response.

(d) Using your best fuzzy controller in (c), simulate the system tracking a 1-rpm step input.

Design Problem 3.6 (Cargo Ship Steering): In this problem we study the development of fuzzy controllers for a ship steering problem. The model for the ship is given in Chapter 6, Section 6.3.1 on page 333. Use the nonlinear model of the ship provided in Equation (6.5) in all the simulation evaluations for the control systems that you develop. Note that we would like to make the closed-loop system for the ship steering system behave like the reference model provided in Chapter 6 for the ship. Note that to simulate the system given in Equation (6.5) on page 334, you will have to convert the third-order nonlinear ordinary differential equation to three first-order ordinary differential equations, as is explained in Chapter 6.

(a) Develop a fuzzy controller for the ship steering problem that will result in achieving the performance specified by the reference model (it may be off slightly during transients). That is, you should achieve nearly the same performance as that shown in Figure 6.6 on page 342.

(b) Design a linear controller for the ship steering problem that will result in achieving the performance specified by the reference model (it may be off slightly during transients).

(c) Compare the results in (a) and (b).

Design Problem 3.7 (Base Braking Control): Antilock braking systems (ABS) are designed to stop vehicles as safely and quickly as possible. Safety is achieved by maintaining lateral stability (and hence steering effectiveness) and reducing braking distances over the case where the brakes are controlled by the driver. Current ABS designs typically use wheel speed compared to the velocity of the vehicle to measure when wheels lock (i.e., when there is "slip" between the tire and the road) and use this information to adjust the duration of brake signal pulses (i.e., to "pump" the brakes). Essentially, as the wheel slip increases past a critical point where it is possible that lateral stability (and hence our ability to steer the vehicle) could be lost, the controller releases the brakes. Then, once wheel slip has decreased to a point where lateral stability is increased and braking effectiveness is decreased, the brakes are reapplied. In this way the ABS cycles the brakes to achieve an optimum trade-off between braking effectiveness and lateral

stability. Inherent process nonlinearities, limitations on our abilities to sense certain variables, and uncertainties associated with process and environment (e.g., road conditions changing from wet asphalt to ice) make the ABS control problem challenging. Many successful proprietary algorithms exist for the control logic for ABS. In addition, several conventional nonlinear control approaches have been reported in the literature.

In this problem, we do not consider brake control for a "panic stop," and hence for this problem the brakes are in a non-ABS mode. Instead, we consider what is referred to as the "base-braking" control problem, where we seek to have the brakes perform consistently as the driver (or an ABS) commands, even though there may be aging of components or environmental effects (e.g., temperature or humidity changes) that can cause "brake grab" or "brake fade." We seek to design a controller that will ensure that the braking torque commanded by the driver (related to how hard we hit the brakes) is achieved by the brake system. Clearly, solving the base-braking problem is of significant importance since there is a direct correlation between safety and the reliability of the brakes in providing the commanded stopping force. Moreover, base-braking algorithms would run in parallel with ABS controllers so that they could also enhance braking effectiveness while the brakes are in an ABS mode.

Figure 3.31 shows the diagram of the base-braking system, as developed in [66, 118]. The input to the system, denoted by $r(kT)$, is the braking torque (in ft-lbs) requested by the driver. The output, $y(kT)$ (in ft-lbs), is the output of a torque sensor, which directly measures the torque applied to the brakes. Note that while torque sensors are not available on current production vehicles, there is significant interest in determining the advantages of using such a sensor. The signal $e(kT)$ represents the error between the reference input and output torques, which is used by the controller to create the input to the brake system, $u(kT)$. A sampling interval of $T = 0.005$ seconds is used.

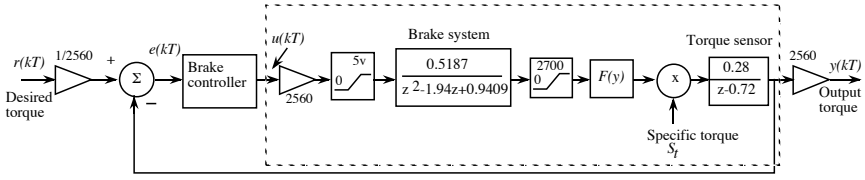


FIGURE 3.31 Brake control system (figure taken from [118], © IEEE).

The General Motors braking system used in this problem is physically limited to processing signals between [0, +1] volts, while the braking torque can range from 0 to 2700 ft-lbs. For this reason and other hardware specific reasons [66], the input torque is attenuated by a factor of 2560 and the output is amplified by the same factor. After $u(kT)$ is multiplied by 2560, it is passed through a saturation nonlinearity where if $2560u(kT) \leq 0$, the brake system receives a zero

input and if $2560u(kT) \geq 5$, the input is 5. The output of the brake system passes through a similar nonlinearity that saturates at zero and 2700. The output of this nonlinearity passes through $F(y)$, which is defined as

$$F(y) = \frac{y}{2502.4419} + 0.0139878$$

The function $F(y)$ was experimentally determined and represents the relationship between brake fluid pressure and the stopping force on the car. Next, $F(y)$ is multiplied by the "specific torque" S_t. This signal is passed through an experimentally determined model of the torque sensor, the signal is scaled, and $y(kT)$ is output.

The specific torque S_t in the braking process reflects the variations in the stopping force of the brakes as the brake pads increase in temperature. The stopping force applied to the wheels is a function of the pressure applied to the brake pads and the coefficient of friction between the brake pads and the wheel rotors. As the brake pads and rotors increase in temperature, the coefficient of friction between the brake pads and the rotors increases. The result is that less pressure on the brake pads is required for the same amount of braking force. The specific torque S_t of this braking system has been found experimentally to lie between two limiting values so that

$$0.85 \leq S_t \leq 1.70$$

To conduct simulations for this problem, you should use the specific methodology that we present next to represent the fact that as you repeatedly apply your brakes, they heat up—which is represented by increasing the value of S_t. In particular, a repeating 4-second input reference signal should be used where each period of this signal corresponds to one application of the brakes. The input reference begins at 0 ft-lbs, increases linearly to 1000 ft-lbs by 2 seconds, and remains constant at 1000 ft-lbs until 4 seconds. After 4 seconds the states of the brake system and controller should be cleared, and the simulation can be run again. For part (d) below the first two 4-second simulations are run with $S_t = 0.85$, corresponding to "cold brakes" (a temperature of 125° F for the brake pads). The next two 4-second simulations are run with S_t increasing linearly from 0.85 at 8 seconds to 1.70 after 12 seconds. Finally, two more 4-second simulations are run with $S_t = 1.7$, corresponding to "hot brakes" (a temperature of 250° F for the brake pads).

(a) Develop a fuzzy controller for the base-braking control problem assuming that the brakes always stay cold (i.e., $S_t = 0.85$).

(b) Develop a fuzzy controller for the base-braking control problem assuming that the brakes always stay hot (i.e., $S_t = 1.7$).

(c) Test the fuzzy controller developed for the cold brakes on the hot ones, and vice-versa.

(d) Next, test the performance of the controller developed for the cold brakes on brakes that heat up over time. Use the simulation methodology outlined above.

(e) Repeat (a)–(d) using a conventional control approach and compare its performance to that of the fuzzy controllers.

Design Problem 3.8 (Rocket Velocity Control): A mathematical model that is useful in the study of the control of the velocity of a single-stage rocket is given by (see [16] and [136])

$$\frac{d\ v(t)}{dt} = c(t)\left(\frac{m}{M - m\ t}\right) - g\left(\frac{R}{R + y(t)}\right) - 0.5\ v^2(t)\left(\frac{\rho_a\ A\ C_d}{M - m\ t}\right) \tag{3.6}$$

where $v(t)$ is the rocket velocity at time t (the plant output), $y(t)$ is the altitude of the rocket (above sea level), and $c(t)$ (the plant input) is the velocity of the exhaust gases (in general, the exhaust gas velocity is proportional to the cross-sectional area of the nozzle, so we take it as the input). Also,

- $M = 15000.0$ kg = the initial mass of the rocket and fuel.
- $m = 100.0\ \frac{\text{kg}}{\text{S}}$ = the exhaust gases mass flow rate (approximately constant for some solid propellant rockets).
- $A = 1.0\ \text{meter}^2$ = the maximum cross-sectional area of the rocket.
- $g = 9.8\ \frac{\text{meters}}{\text{S}^2}$ = the the acceleration due to gravity at sea level.
- $R = 6.37 \times 10^6$ meters = the radius of the earth.
- $\rho_a = 1.21\ \frac{\text{kg}}{\text{m}^3}$ = the density of air.
- $C_d = 0.3$ = the drag coefficient for the rocket.

Due to the loss of fuel resulting from combustion and exhaust, the rocket has a time-varying mass.

To specify the performance objectives, we use a "reference model." That is, we desire to have the closed-loop system from r to v behave the same as a reference model does from r to v_m. In our case, we choose the reference model to be

$$\frac{dv_m(t)}{dt} = -0.2v_m(t) + 0.2r(t)$$

where $v_m(t)$ specifies the desired rocket velocity. This shows that we would like a first-order-type response due to a step input.

(a) Pick an altitude trajectory $y(t)$ that you would like to follow.

(b) Develop a fuzzy controller for the rocket velocity control problem and demonstrate that it meets the performance specifications via simulation.

(b) Develop a controller using conventional methods and demonstrate that it meets the performance objectives. Compare its performance to that of the fuzzy controller.

Design Problem 3.9 (An Acrobot)*: An acrobot is an underactuated, unstable two-link planar robot that mimics the human acrobat who hangs from a bar and tries to swing up to a perfectly balanced upside-down position with his or her hands still on the bar (see Figure 3.32). In this problem we apply direct fuzzy control to two challenging robotics control problems associated with the acrobot, swing-up and balancing, and use different controllers for each case. Typically, a heuristic strategy is used for swing-up, where the goal is to force the acrobot to reach its vertical upright position with near-zero velocity on both links. Then, when the links are close to the inverted position, a balancing controller is switched on and used to maintain the acrobot in the inverted position (again, see Figure 3.32). Such a strategy was advocated in earlier work in [191] and [190].

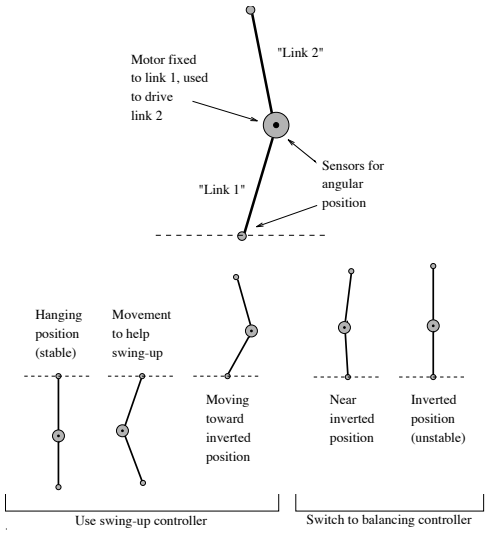


FIGURE 3.32 The acrobot (figure taken from [27], © Kluwer Academic Pub.).

The acrobot has a single actuator at the elbow and no actuator at the shoulder; the system is "underactuated" because we desire to control two links of the acrobot (each with a degree of freedom) with only a single system input. The configuration of a simple acrobot, from which the system dynamics are obtained, is shown in Figure 3.33. The joint angles q_1 and q_2 serve as the generalized system coordinates; m_1 and m_2 specify the mass of the links; l_1 and l_2 specify the link lengths; l_{c1} and l_{c2} specify the distance from the axis of rotation of each link to

its center of mass; and I_1 and I_2 specify the moment of inertia of each link taken about an axis coming out of the page and passing through its center of mass. The single system input τ is defined such that a positive torque causes q_2 to increase (move in the counterclockwise direction).

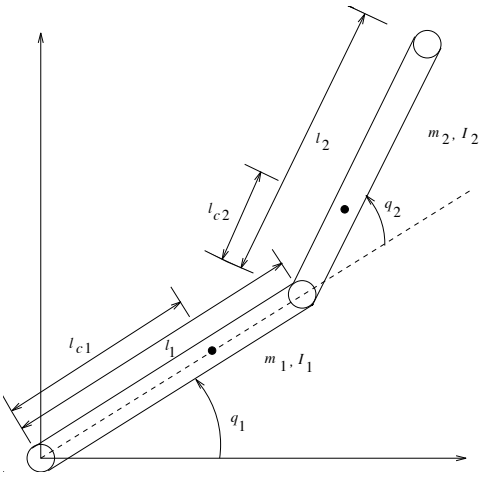


FIGURE 3.33 Simple acrobot notation (figure taken from [27], © Kluwer Academic Pub.).

The dynamics of the simple acrobot may be obtained by determining the Euler-Lagrange equations of motion for the system. This is accomplished by finding the Lagrangian of the system, or the difference between the system's kinetic and potential energies. Indeed, determining the kinetic and potential energies of each link is the most difficult task in obtaining the system dynamics and requires forming the manipulator Jacobian (see Chapters 5 and 6 of [192] for more details). In [192], Spong has developed the equations of motion of a planar elbow manipulator; this manipulator is identical to the acrobot shown in Figure 3.33, except that it is actuated at joints one *and* two. The dynamics of the acrobot are simply those of the planar manipulator, with the term corresponding to the input torque at the first joint set equal to zero. The acrobot dynamics may be described by the two second-order differential equations

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + h_1 + \phi_1 = 0 \tag{3.7}$$

$$d_{12}\ddot{q}_1 + d_{22}\ddot{q}_2 + h_2 + \phi_2 = \tau \tag{3.8}$$

where the coefficients in Equations (3.7) and (3.8) are defined as

$$d_{11} = m_1 l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos(q_2)) + I_1 + I_2$$

$$
\begin{aligned}
d_{22} &= m_2 l_{c2}^2 + I_2 \\
d_{12} &= m_2(l_{c2}^2 + l_1 l_{c2}\cos(q_2)) + I_2 \\
h_1 &= -m_2 l_1 l_{c2}\sin(q_2)\dot{q}_2^2 - 2m_2 l_1 l_{c2}\sin(q_2)\dot{q}_2\dot{q}_1 \\
h_2 &= m_2 l_1 l_{c2}\sin(q_2)\dot{q}_1^2 \\
\phi_1 &= (m_1 l_{c1} + m_2 l_1)g\cos(q_1) + m_2 l_{c2} g\cos(q_1 + q_2) \\
\phi_2 &= m_2 l_{c2} g\cos(q_1 + q_2)
\end{aligned}
$$

In our acrobot model, we have also limited the range for joint angle q_2 to $[-\pi, \pi]$ (i.e., the second link is not free to rotate in a complete revolution—it cannot cross over the first link). We have also cascaded a saturation nonlinearity between the controller output and plant input to limit the input torque magnitude to 4.5 N-m. The model parameter values are given in Table 3.12.

TABLE 3.12 Acrobot Model Parameters Used in Simulations

Parameter	Value
m_1	1.9008 kg
m_2	0.7175 kg
l_1	0.2 m
l_2	0.2 m
l_{c1}	1.8522×10^{-1} m
l_{c2}	6.2052×10^{-2} m
I_1	4.3399×10^{-3} kg·m^2
I_2	5.2285×10^{-3} kg·m^2

When you simulate the acrobot, be sure to use a good numerical simulation algorithm with a small enough integration step size. For instance, use a fourth-order Runge-Kutta technique with an integration step size of 0.0025 seconds. To simulate the effects of implementing the controllers on a digital computer, sample the output signals with a period $T = 0.01$ seconds, and only update the control input every T seconds (holding the value constant in between updates).

(a) Find a linear model about the equilibrium inverted position ($q_1 = \pi/2$, $q_2 = 0$, $\dot{q}_1 = 0$, $\dot{q}_2 = 0$) with $\tau = 0$ (note that there is actually a continuum of equilibrium positions). Define the state vector $x = [q_1 - \pi/2, q_2, \dot{q}_1, \dot{q}_2]^\top$ to transform the balancing control problem to a regulation problem. The acrobot dynamics linearized about $x = [0, 0, 0, 0]^\top$ may be described by

$$
\begin{aligned}
\dot{x} &= Ax + B\tau \\
y &= Cx + D\tau
\end{aligned}
$$

Find the numerical values for the A, B, C, and D matrices and verify that the system is unstable. Design a linear quadratic regulator (LQR), and illustrate

its performance in simulation for an initial condition $q_1(0) = \pi/2 + 0.04$, $q_2(0) = -0.0500, \dot{q}_1(0) = -0.2000$, and $\dot{q}_2(0) = 0.0400$. This initial condition is such that the first link is approximately 2.29° beyond the inverted position, while the second link is displaced approximately -2.86° from the first link. The initial velocities are such that the first link is moving away from the inverted position, while the second link is moving into line with the first link.

(b) Next we study the development of a fuzzy controller for the acrobot. Suppose that your fuzzy controller has four inputs: $g_0(q_1 - \pi/2)$, $g_1 q_2$, $g_2 \dot{q}_1$, and $g_3 \dot{q}_2$; and a single output. Here, g_0–g_3 are scaling gains, and the output of the fuzzy controller is scaled by a gain h. Test your controller in simulation using the same initial conditions as in part (a). Hint: Use the approach of copying the LQR gains as we did for the rotational inverted pendulum. Also, consider specifying the output membership function centers via a nonlinear function.

Design Problem 3.10 (Fuzzy Warning Systems): In this problem you will fully develop the fuzzy decision-making systems that are used as warning systems for an infectious disease and an aircraft.

(a) Fully develop the fuzzy system that will serve as a warning system for the infectious disease warning system described in Section 3.6.1 on page 162. Test the performance of the system by showing that it can provide proper warnings for each of the warning conditions.

(b) Repeat (a) but for the aircraft failure warning system described in Section 3.6.2 on page 166.

Design Problem 3.11 (Automobile Speed Warning System)*: In this problem you will study the development of a fuzzy decision-making system for "intelligent vehicle and highway systems" where there is a focus on the development of "automated highway systems" (AHS). In AHS it is envisioned that vehicles will be automatically driven by an on-board computer that interacts with a specially designed roadway. Such AHS offer significant improvements in safety and overall roadway efficiency (i.e., they increase vehicle throughput). It is evident that the AHS will evolve by the sequential introduction of increasingly advanced automotive and roadway subsystems. One such system that may be used is a speed advisory system to be placed on the vehicle to enhance safety, as shown in Figure 3.34. There is a vehicle and a changeable electronic sign that displays the speed limit for the current weather and traffic conditions, and in addition transmits the current speed limit to passing vehicles. There is a receiver in the vehicle that can collect this speed limit information. The problem is to design a speed advisory system that can display warnings to the driver about the dangers of exceeding the speed limit. We will use this problem to illustrate the

development of a fuzzy decision-making system that can emulate the manner in which a human safety expert would warn the driver about traveling at dangerous speeds if such a person could be available on each and every vehicle.

FIGURE 3.34 Scenario for an automobile speed advisory system.

The first step in the design of the speed advisory system is to specify the types of advice that the safety expert should provide. Then the expert should indicate what variables need to be known to in order to provide such advice. This will help define the inputs and outputs of the fuzzy system. Suppose specifications dictate that the advisory system is to provide (1) an indication of the likelihood (on a scale of zero to one, with one being very likely) that the vehicle will exceed the current speed limit (which we assume is fixed at 55 mph for our example), and (2) a numeric rating between one and ten (ten being the highest) of how dangerous the current operating speed is. Suppose that to provide such information the safety expert indicates that the error between the current vehicle speed and the speed limit and the derivative of the error between the current vehicle speed and the speed limit will be needed. Clearly, the fuzzy system will then have two inputs and two outputs.

To develop a fuzzy speed warning system, we need to have the engineer interview the safety expert to determine how to decide what warnings should be provided to the driver. The safety expert will provide a linguistic description of her or his approach. First, define the universe of discourse for the speed error input to the fuzzy system to be $[-100, 100]$ mph (where 100 mph is the highest speed that the vehicle can attain) and universe of discourse for the change in speed input to be $[-10, 10]$ mph/sec (so that the vehicle can accelerate or decelerate at most 10 mph in one second). The universe of discourse for the output that indicates the likelihood to exceed the speed limit is $[0, 1]$, and the universe of discourse for the danger rating output is $[0, 10]$, with 10 representing the most dangerous situation. We use e to denote the speed error, $\dot{e}$ for the derivative of the error, s for the likelihood that the speed limit will be exceeded, and d for the danger rating for the current speed. The linguistic variables for the inputs could be = "error" and = "error-deriv," and for the outputs they could be = "likely-to-exceed-limit" and = "danger-rating."

Examples of linguistic rules for the fuzzy system could be the following: (1) If error is "possmall" and error-deriv is "neglarge" Then likely-to-exceed-limit is "medium" (i.e., if the speed is below the limit, but it is approaching the limit quickly, then there is some concern that the speed limit will be exceeded), (2) If

error is "zero" and error-deriv is "neglarge" Then likely-to-exceed-limit is "large" (i.e., if the speed is currently at the speed limit and it is increasing rapidly, then there is a significant concern that the speed limit will be exceeded), and (3) If error is "possmall" and error-deriv is "neglarge" Then danger-rating is "small" (i.e., if the speed is below the limit, but it is approaching the limit quickly, Then there is some danger because the limit is likely to at least slightly exceed what experts judge to be a safe driving speed).

(a) Develop a fuzzy decision-making system that can serve as a speed advisory system for automobiles.

(b) Develop a test scenario for the fuzzy system. Clearly explain how you will test the system.

(c) Test the system according to your plan in (b), and show your results (these should include showing that the system can provide warnings under the proper conditions).

Design Problem 3.12 (Design of Fuzzy Decision-Making Systems)*: In this problem you will assist in both defining the problem and the solution. The problem focuses on the development of fuzzy decision-making systems that are not necessarily used in the control of a system.

(a) Suppose that you wish to buy a used car. You have various priorities with regard to price, color, safety features, the year the car was made, and the make of the car. Quantify each of these characteristics with fuzzy sets and load appropriate rules into a fuzzy decision-making system that represents your own priorities in purchasing a used car. For instance, when presented with N cars in a row, the fuzzy system should be able to provide a value that represents your ranking of the desirability of purchasing each of the cars. Demonstrate in simulation the performance of the system (i.e., that it properly represents your decision-making strategy for purchasing a used car).

(b) Suppose that you wish to design a computer program that will guess which team wins in a football game (that has already been played) when it is given only total passing yards, total rushing yards, and total time of possession for each team. Design a fuzzy decision-making system that will guess at the outcome (score) of a game based on these inputs. Test the performance of the system by using data from actual games played by your favorite team.

(c) An alternative, perhaps more interesting system to develop than the one described in (b), would be one that would predict who would win the game *before* it was played. How would you design such a system?

CHAPTER 4

Nonlinear Analysis

So far as the laws of mathematics refer to reality,
they are not certain. And so far as they are certain,
they do not refer to reality.
–Albert Einstein

4.1 Overview

As we described it in Chapters 1–3, the standard control engineering methodology involves repeatedly coordinating the use of modeling, controller design, simulation, mathematical analysis, implementations, and evaluations to develop control systems. In Chapters 2 and 3 we showed via examples how modeling is used, and we provided guidelines for controller design. Moreover, we discussed how to simulate fuzzy controllers, highlighted some issues that are encountered in implementation, and showed case studies that illustrated the design, simulation, and implementation of fuzzy control systems. In this chapter we show how to perform mathematical analysis of various properties of fuzzy control systems so that the designer will have access to all steps of the basic control design methodology.

Basically, we use the mathematical model of the plant and nonlinear analysis to enhance our confidence in the reliability of a fuzzy control system by verifying stability and performance specifications and possibly redesigning the fuzzy controller. We emphasize, however, that mathematical analysis alone cannot provide definitive answers about the reliability of the fuzzy control system since such analysis proves properties about the model of the process, not the actual physical process. Indeed, it can be argued that a mathematical model is never a perfect representation of a physical process; hence, while nonlinear analysis may appear to provide definitive statements about control system reliability, you must understand that such statements are only accurate to the extent that the mathematical model is accurate. Nonlinear analysis does not replace the use of common sense and evalua-

tion via simulations and experimentation. It simply assists in providing a rigorous engineering evaluation of a fuzzy control system before it is implemented.

It is important to note that the advantages of fuzzy control often become most apparent for very complex problems where we have an intuitive idea about how to achieve high-performance control (e.g., the two-link flexible robot case study in Chapter 3). In such control applications, an accurate mathematical model is so complex (i.e., high order, nonlinear, stochastic, with many inputs and outputs) that it is sometimes not very useful for the analysis and design of conventional control systems since assumptions needed to utilize conventional control design approaches are often violated. The conventional control engineering approach to this problem is to use an approximate mathematical model that is accurate enough to characterize the essential plant behavior in a certain region of operation, yet simple enough so that the necessary assumptions needed to apply the analysis and design techniques are satisfied. However, due to the inaccuracy of the model, upon implementation the developed controllers often need to be tuned via the "expertise" of the control engineer.

The fuzzy control approach, where explicit characterization and utilization of control expertise is used earlier in the design process, largely avoids the problems with model complexity that are related to design. That is, for the most part, fuzzy control system design does not depend on a mathematical model unless it is needed to perform simulations to gain insight into how to choose the rule-base and membership functions. However, the problems with model complexity that are related to analysis have not been solved (i.e., analysis of fuzzy control systems critically depends on the form of the mathematical model); hence, it is often difficult or impossible to apply nonlinear analysis techniques to the applications where the advantages of fuzzy control are most apparent!

For instance, existing results for stability analysis of fuzzy control systems typically require that the plant model be deterministic and satisfy some continuity constraints, and sometimes require the plant to be linear or have a very specific mathematical form. The most general approaches to the nonlinear analysis of fuzzy control systems are those due to Lyapunov (his direct and indirect methods). On the other hand, for some stability analysis approaches (e.g., for absolute stability), the only results for analysis of steady-state tracking error of fuzzy control systems, and the existing results on the use of describing functions for analysis of limit cycles, essentially require a linear time-invariant plant (or one that has a special form so that the nonlinearities can be bundled into one nonlinear component in the loop).

These limitations in the theory help to show that fuzzy control technology is in a sense leading the theory; the practitioner will go ahead with the design and implementation of many fuzzy control systems without the aid of nonlinear analysis. In the meantime, theorists will continue to develop a mathematical theory for the verification and certification of fuzzy control systems. This theory will have a synergistic effect by driving the development of fuzzy control systems for applications where there is a need for highly reliable implementations.

Overall, the objectives of this chapter are as follows:

- To help teach sound techniques for the construction of fuzzy controllers by alerting the designer to some of the pitfalls that can occur if a rule-base is improperly constructed (e.g., instabilities, limit cycles, and steady-state errors).
- To provide insights into how to modify the fuzzy controller rule-base to guarantee that performance specifications are met (thereby helping make the fuzzy controller design process more systematic).
- To provide examples of how to apply the theory to some simple fuzzy control system analysis and design problems.

We provide an introduction to the use of Lyapunov stability analysis in Section 4.3. In particular, we introduce Lyapunov's direct and indirect methods and illustrate the use of these via several examples and an inverted pendulum application. Moreover, we show how to use Lyapunov's direct method for the analysis of stability of plants represented with Takagi-Sugeno fuzzy systems that are controlled with a Takagi-Sugeno form of a fuzzy controller.

We introduce analysis of absolute stability in Section 4.4, steady-state error analysis in Section 4.5, and describing function analysis in Section 4.6. In each of these sections we show how the methods can aid in picking the membership functions in a fuzzy controller to avoid limit cycles, and instabilities, and ultimately to meet a variety of closed-loop specifications. Since most fuzzy control systems are "hybrid" in that the controller contains a linear portion (e.g., an integrator or differentiator) as well as a nonlinear portion (a fuzzy system), we will show how to use nonlinear analysis to design both of these portions of the fuzzy control system.

Overall, while we highly recommend that you study this chapter carefully, if you are not concerned with the verification of the behavior of a fuzzy control system, you can skip to the next chapter. Indeed, there is no direct dependence of any topic in the remaining chapters of this book on the material in this chapter. This chapter simply tends to deepen your understanding of the material studied in Chapters 1–3.

4.2 Parameterized Fuzzy Controllers

In this section we will introduce the fuzzy control system to be investigated and briefly examine the nonlinear characteristics of the fuzzy controller. Except in Section 4.3, the closed-loop systems in this chapter will be as shown in Figure 4.1 (where we assume that $G(s)$ is a single-input single-output (SISO) linear system) or they will be modified slightly so that the fuzzy controller is in the feedback path.[1] We will be using both SISO and MISO (multi-input single-output) fuzzy controllers as they are defined in the next subsections.

1. We assume throughout this chapter that the fuzzy controller is designed so that the existence and uniqueness of the solution of the differential equation describing the closed-loop system is guaranteed.

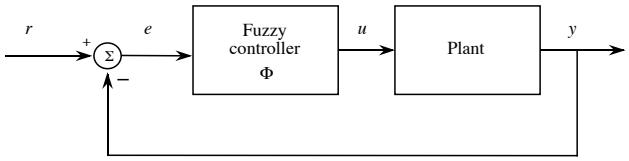


FIGURE 4.1 Fuzzy control system.

4.2.1 Proportional Fuzzy Controller

For the "proportional fuzzy controller," as the SISO fuzzy controller in Figure 4.1 is sometimes called, the rule-base can be constructed in a symmetric fashion with rules of the following form:

1. **If** e is NB **Then** u is NB
2. **If** e is NM **Then** u is NM
3. **If** e is NS **Then** u is NS
4. **If** e is ZE **Then** u is ZE
5. **If** e is PS **Then** u is PS
6. **If** e is PM **Then** u is PM
7. **If** e is PB **Then** u is PB

where NB, NM, NS, ZE, PS, PM, and PB are linguistic values representing "negative big," "negative medium," and so on.

The membership functions for the premises and consequents of the rules are shown in Figure 4.2. Notice in Figure 4.2 that the widths of the membership functions are parameterized by A and B. Throughout this chapter, unless it is indicated otherwise, the same rule-base and similar uniformly distributed membership functions will be used for all applications (where if the number of input and output membership functions and rules increase, our analysis approaches work in a similar manner). The fuzzy controller will be adjusted by changing the values of A and B. The manner in which these values affect the nonlinear map that the fuzzy controller implements will be discussed below. The fuzzy inference mechanism operates by using the product to combine the conjunctions in the premise of the rules and in the representation of the fuzzy implication. Singleton fuzzification is used, and defuzzification is performed using the center-average method.

The SISO fuzzy controller described above implements a static nonlinear input-output map between its input $e(t)$ and output $u(t)$. As we discussed in Chapter 2, the particular shape of the nonlinear map depends on the rule-base, inference strategy, fuzzification, and defuzzification strategy utilized by the fuzzy controller. Consider the input-output map for the above fuzzy controller shown in Figure 4.3 with $A = B = 1$. Modifications to the fuzzy controller can provide an infinite variety

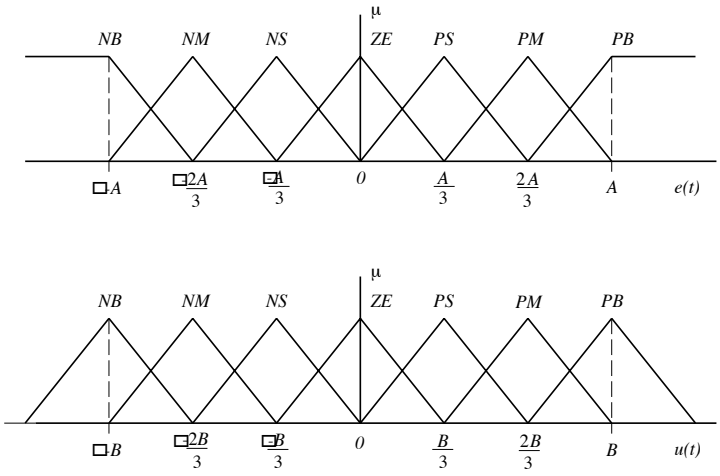


FIGURE 4.2 Membership functions for $e(t)$ and $u(t)$ (figure taken from [83], © John Wiley and Sons).

of such input-output maps (e.g., by changing the consequents of the rules). Notice, however, that there is a marked similarity between the input-output map in Figure 4.3 and the standard *saturation* nonlinearity. In fact, the parameters A and B from the fuzzy controller are similar to the saturation parameters of the standard saturation nonlinearity—that is, B is the level at which the output saturates, and A is the value of $e(t)$ at which the saturation of $u(t)$ occurs. Because the input-output map of the fuzzy controller is odd, $-B$ is the saturation level for $e(t) \leq -A$, and $-A$ is the value of $e(t)$ where the saturation occurs. By modifying A and B (and hence moving the input and output membership functions), we can change the input-output map nonlinearity and its effects on the system. Throughout this chapter, except in Section 4.3, we will always use rules in the form described above. We emphasize, however, that the nonlinear analysis techniques used in this chapter will work in the same manner for other types of rule-bases (and different fuzzification, inference, and defuzzification techniques).

4.2.2 Proportional-Derivative Fuzzy Controller

There are many different types of fuzzy controllers we could examine for the MISO case. Here, aside from Section 4.3, we will constrain ourselves to the two input "proportional-derivative fuzzy controller" (as it is sometimes called). This controller is similar to our SISO fuzzy controller with the addition of the second input, $\dot{e}$. In fact, the membership functions on the universes of discourses and linguistic values NB, NM, NS, ZE, PS, PM, and PB for e and u are the same as they are shown in Figure 4.2 and will still be adjusted using the parameters A and B, respectively. The membership functions on the universe of discourse and the linguistic values for the second input, $\dot{e}$, are the same as for e with the exception that the adjustment

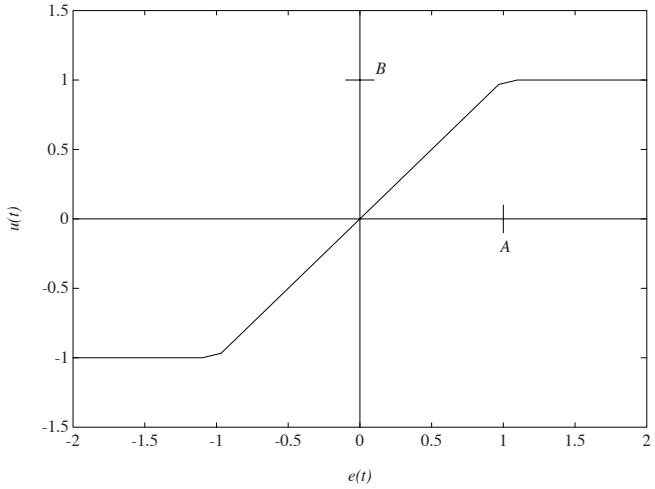


FIGURE 4.3 Input-output map of the proportional fuzzy controller (figure taken from [83], © John Wiley and Sons).

parameter will be denoted by D. Therefore, there are now three parameters for changing the fuzzy controller: A, B, and D. Assuming that there are seven membership functions on each input universe of discourse, there are 49 possible rules that can be put in the rule-base. A typical rule will take on the form

If e is NB **and** $\dot{e}$ is NB **Then** u is NB

The complete set of rules is shown in tabulated form in Table 4.1. In Table 4.1 the premises for the input $\dot{e}$ are represented by the linguistic values found in the top row, the premises for the input e are represented by the linguistic values in the left-most column, and the linguistic values representing the consequents for each of the 49 rules can be found at the intersections of the row and column of the appropriate premises (note that this is a slightly different tabular form than what we used earlier since we list the actual linguistic values here). The upper left-hand corner of the body of Table 4.1 is the representation of the above rule, "**If** e is NB **and** $\dot{e}$ is NB **Then** u is NB." The remainder of the MISO fuzzy controller is similar to the SISO fuzzy controller (i.e., singleton fuzzification, the product for the premise and implication, and center-average defuzzification are used). Notice that there is a type of pattern of rules in Figure 4.1 that will result in a particular (somewhat irregular shaped) nonlinearity. This particular rule-base was chosen for an application we study in describing function analysis later in the chapter.

We can also construct an input-output map for this MISO fuzzy controller. The parameters A and B have the same effect as with the SISO fuzzy controller. By changing the values A, B, and D, we can change the effect of the MISO fuzzy controller on the closed-loop system without reconstructing the rule-base or any

TABLE 4.1 Rule Table for PD Fuzzy Controller

"output" u		"change-in-error" $\dot{e}$						
		NB	NM	NS	ZE	PS	PM	PB
"error" e	NB	NB	NS	PS	PB	PB	PB	PB
	NM	NB	NM	ZE	PM	PM	PB	PB
	NS	NB	NM	NS	PS	PM	PB	PB
	ZE	NB	NM	NS	ZE	PS	PM	PB
	PS	NB	NB	NM	NS	PS	PM	PB
	PM	NB	NB	NM	NM	ZE	PM	PB
	PB	NB	NB	NB	NB	NS	PS	PB

other portion of the fuzzy controller. Again, we emphasize that while we use this particular fuzzy controller, which is conveniently parameterized by A, B, and D, the approaches to nonlinear analysis in this chapter work in a similar manner for fuzzy controllers that use other membership functions, rule-bases, inference mechanisms, and fuzzification and defuzzification strategies.

4.3 Lyapunov Stability Analysis

Often the designer is first concerned about investigating the stability properties of a fuzzy control system, since it is often the case that if the system is unstable there is no chance that any other performance specifications will hold. For example, if the fuzzy control system for an automobile cruise control is unstable, you would be more concerned with the possibility of unsafe operation than with how well it regulates the speed at the set-point. In this section we provide an overview of two approaches to stability analysis of fuzzy control systems: Lyapunov's direct and indirect methods. We provide several examples for each of the methods, including the application of Lyapunov's direct method to the stability analysis of Takagi-Sugeno fuzzy systems. In the next section we show how to use the circle criterion in the analysis of absolute stability of fuzzy control systems.

4.3.1 Mathematical Preliminaries

Suppose that a dynamic system is represented with

$$\dot{x}(t) = f(x(t)) \tag{4.1}$$

where $x \in \Re^n$ is an n vector and $f : D \to \Re^n$ with $D = \Re^n$ or $D = B(h)$ for some $h > 0$ where

$$B(h) = \{x \in \Re^n : |x| < h\}$$

is a ball centered at the origin with a radius of h and $|\cdot|$ is a norm on $\Re^n$ (e.g., $|x| = \sqrt{(x^\top x)}$). If $D = \Re^n$ then we say that the dynamics of the system are defined globally, while if $D = B(h)$ they are only defined locally. Assume that for every x_0

the initial value problem

$$\dot{x}(t) = f(x(0)), \quad x(0) = x_0 \tag{4.2}$$

possesses a unique solution $\phi(t, x_0)$ that depends continuously on x_0. A point $x_e \in \Re^n$ is called an "equilibrium point" of Equation (4.1) if $f(x_e) = 0$ for all $t \geq 0$. An equilibrium point x_e is an "isolated equilibrium point" if there is an $h' > 0$ such that the ball around x_e,

$$B(x_e, h') = \{x \in \Re^n : |x - x_e| < h'\}$$

contains no other equilibrium points besides x_e. As is standard, we will assume that the equilibrium of interest is an isolated equilibrium located at the origin of $\Re^n$. This assumption results in no loss of generality since if $x_e \neq 0$ is an equilibrium of Equation (4.1) and we let $\bar{x}(t) = x(t) - x_e$, then $\bar{x} = 0$ is an equilibrium of the transformed system

$$\dot{\bar{x}}(t) = \bar{f}(\bar{x}(t)) = f(\bar{x}(t) + x_e)$$

(for an example of this idea, see Section 4.3.4).

The equilibrium $x_e = 0$ of (4.1) is "stable" (in the sense of Lyapunov) if for every $\epsilon > 0$ there exists a $\delta(\epsilon) > 0$ such that $|\phi(t, x_0)| < \epsilon$ for all $t \geq 0$ whenever $|x_0| < \delta(\epsilon)$ (i.e., it is stable if when it starts close to the equilibrium it will stay close to it). The notation $\delta(\epsilon)$ means that δ depends on ϵ. A system that is not stable is called "unstable."

The equilibrium $x_e = 0$ of Equation (4.1) is said to be "asymptotically stable" if it is stable and there exists $\eta > 0$ such that $\lim_{t\to\infty} \phi(t, x_0) = 0$ whenever $|x_0| < \eta$ (i.e., it is asymptotically stable if when it starts close to the equilibrium it will converge to it).

The set $X_d \subset \Re^n$ of all $x_0 \in \Re^n$ such that $\phi(t, x_0) \to 0$ as $t \to \infty$ is called the "domain of attraction" of the equilibrium $x_e = 0$ of Equation (4.1). The equilibrium $x_e = 0$ is said to be "globally asymptotically stable" if $X_d = \Re^n$ (i.e., if no matter where the system starts, its state converges to the equilibrium asymptotically).

As an example, consider the scalar differential equation

$$\dot{x}(t) = -2x(t)$$

which is in the form of Equation (4.2). For this system, $D = \Re^1$ (i.e., the dynamics are defined on the entire real line, not just some region around zero). We have $x_e = 0$ as an equilibrium point of this system since $0 = -2x_e$. Notice that for any x_0, we have the solution

$$\phi(t, x_0) = x_0 e^{-2t} \to 0$$

as $t \to \infty$ so that the equilibrium $x_e = 0$ is stable since if you are given any $\epsilon > 0$ there exists a $\delta > 0$ such that if $|x_0| < \delta$, $\phi(t, x_0) \leq \epsilon$. To see this, simply

choose $\delta = \epsilon$ for any $\epsilon > 0$ that you choose. Also note that since for any $x_0 \in \Re^n$, $\phi(t, x_0) \to 0$, the system is globally asymptotically stable. While determining if this system possesses certain stability properties is very simple since the system is so simple, for complex nonlinear systems it is not so easy. One reason for this is that for complex nonlinear systems, it is difficult to even solve the ordinary differential equations (i.e., to find $\phi(t, x_0)$ for all t and x_0). However, Lyapunov's methods provide two techniques that allow you to determine the stability properties without solving the ordinary differential equations.

4.3.2 Lyapunov's Direct Method

The stability results for an equilibrium $x_e = 0$ of Equation (4.1) that we provide next depend on the existence of an appropriate "Lyapunov function" $V : D \to \Re$ where $D = \Re^n$ for global results (e.g., global asymptotic stability) and $D = B(h)$ for some $h > 0$, for local results (e.g., stability in the sense of Lyapunov or asymptotic stability). If V is continuously differentiable with respect to its arguments, then the derivative of V with respect to t along the solutions of Equation (4.1) is

$$\dot{V}_{(4.1)}(x(t)) = \nabla V(x(t))^\top f(x(t))$$

where

$$\nabla V(x(t)) = \left[\frac{\partial V}{\partial x_1}, \frac{\partial V}{\partial x_2}, \ldots, \frac{\partial V}{\partial x_n}\right]^\top$$

is the gradient of V with respect to x. Using the subscript on $\dot{V}$ is sometimes cumbersome, so we will at times omit it with the understanding that the derivative of V is taken along the solutions of the differential equation that we are studying the stability of.

Lyapunov's direct method is given by the following:

1. Let $x_e = 0$ be an equilibrium for Equation (4.1). Let $V : B(h) \to \Re$ be a continuously differentiable function on $B(h)$ such that $V(0) = 0$ and $V(x) > 0$ in $B(h) - \{0\}$, and $\dot{V}_{(4.1)}(x) \leq 0$ in $B(h)$. Then $x_e = 0$ is stable. If, in addition, $\dot{V}_{(4.1)}(x) < 0$ in $B(h) - \{0\}$, then $x_e = 0$ is asymptotically stable.

2. Let $x_e = 0$ be an equilibrium for Equation (4.1). Let $V : \Re^n \to \Re$ be a continuously differentiable function such that $V(0) = 0$ and $V(x) > 0$ for all $x \neq 0$, $|x| \to \infty$ implies that $V(x) \to \infty$, and $\dot{V}_{(4.1)}(x) < 0$ for all $x \neq 0$. Then $x_e = 0$ is globally asymptotically stable.

As an example, consider the system

$$\dot{x}(t) = -2x^3$$

that has an equilibrium $x_e = 0$. Choose

$$V(x) = \frac{1}{2}x^2$$

With this choice we have

$$\dot{V} = \frac{\partial V}{\partial x}\frac{dx}{dt} = x\dot{x} = -2x^4$$

so that clearly if $x \neq 0$ then $-2x^4 < 0$, so that by Lyapunov's direct method $x_e = 0$ is asymptotically stable. Notice that $x_e = 0$ is in fact globally asymptotically stable.

While Lyapunov's direct method has found wide application in conventional control, it is important to note that it is not always easy to find the "Lyapunov function" V that will have the above properties so that we can guarantee that the system is stable. Next, we introduce Lyapunov's indirect method.

4.3.3 Lyapunov's Indirect Method

Let $\frac{\partial f}{\partial x} = \left[\frac{\partial f_i}{\partial x_j}\right]$ denote the $n \times n$ "Jacobian matrix." For the next result, assume that $f : D \to \Re^n$ where $D \subset \Re^n$, that $x_e \in D$, and that f is continuously differentiable.

Lyapunov's indirect method is given by the following: Let $x_e = 0$ be an equilibrium point for the nonlinear system Equation (4.1). Let the $n \times n$ matrix

$$\bar{A} = \left.\frac{\partial f}{\partial x}(x)\right|_{x=x_e=0}$$

then

1. The origin $x_e = 0$ is asymptotically stable if $\mathrm{Re}[\lambda_i] < 0$ (the real part of λ_i) for all eigenvalues λ_i of $\bar{A}$.
2. The origin $x_e = 0$ is unstable if $\mathrm{Re}[\lambda_i] > 0$ for one or more eigenvalues of $\bar{A}$.
3. If $\mathrm{Re}[\lambda_i] \leq 0$ for all i with $\mathrm{Re}[\lambda_i] = 0$ for some i where the λ_i are the eigenvalues of $\bar{A}$, then we cannot conclude anything about the stability of $x_e = 0$ from Lyapunov's indirect method.

Lyapunov's indirect method has also found wide application in conventional control. Note that the term "indirect" is used since we arrive at our conclusions about stability indirectly by first linearizing the system about an operating point. The indirect method is sometimes called Lyapunov's "first method," while the direct method is referred to as his "second method."

As an example, consider the system

$$\dot{x} = -x^2$$

that has an equilibrium $x_e = 0$. We have

$$\bar{A} = \frac{\partial f}{\partial x}(x)\bigg|_{x=x_e=0} = -2x = 0$$

so that we can conclude nothing about stability. In the next section we will show a simple example where both Lyapunov techniques can be used to draw conclusions about stability for a fuzzy control system.

4.3.4 Example: Inverted Pendulum

In this section we will illustrate the use of Lyapunov's indirect method for stability analysis of an inverted pendulum (one with a model different from the ones used in Chapters 2 and 3).

A simple model of the pendulum shown in Figure 4.4 is given by

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\tfrac{g}{\ell}\sin(x_1) - \tfrac{k}{m}x_2 + \tfrac{1}{m\ell^2}T \end{aligned} \tag{4.3}$$

where $g = 9.81$, $\ell = 1.0$, $m = 1.0$, $k = 0.5$, x_1 is the angle (in radians) shown in Figure 4.4, x_2 is the angular velocity (in radians per second), and T is the control input.

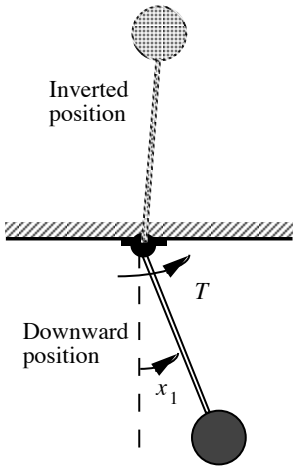


FIGURE 4.4 Pendulum.

If we assume that $T = 0$, then there are two distinct isolated equilibrium points, one in the downward position $[0, 0]^\top$ and one in the inverted position $[\pi, 0]^\top$. Since we are interested in the control of the pendulum about the inverted position, we

need to translate the equilibrium by letting $\bar{x} = x - [\pi, 0]^\top$. From this we obtain

$$\begin{aligned} \dot{\bar{x}}_1 &= \bar{x}_2 = \bar{f}_1(\bar{x}) \\ \dot{\bar{x}}_2 &= \tfrac{g}{\ell}\sin(\bar{x}_1) - \tfrac{k}{m}\bar{x}_2 + \tfrac{1}{m\ell^2}T = \bar{f}_2(\bar{x}) \end{aligned} \tag{4.4}$$

where if $T = 0$ then $\bar{x} = 0$ corresponds to the equilibrium $[\pi, 0]^\top$ in the original system in Equation (4.3), so studying the stability of $\bar{x} = 0$ corresponds to studying the stability of the fuzzy control system about the inverted position. Now, it is traditional to omit the cumbersome bar notation in Equation (4.4) and study the stability of $x = 0$ for the system

$$\begin{aligned} \dot{x}_1 &= x_2 = f_1(x) \\ \dot{x}_2 &= \tfrac{g}{\ell}\sin(x_1) - \tfrac{k}{m}x_2 + \tfrac{1}{m\ell^2}T = f_2(x) \end{aligned} \tag{4.5}$$

with the understanding that we are actually studying the stability of Equation (4.4). Assume that the fuzzy controller denoted by $T = \Phi(x_1, x_2)$, which utilizes x_1 and x_2 as inputs to generate T as an output, is designed so that f (i.e., the closed-loop dynamics) are continuously differentiable and so that D is a neighborhood of the origin.

Application of Lyapunov's Direct Method

Assume that for the fuzzy controller $\Phi(0, 0) = 0$ so that the equilibrium is preserved. Choose

$$V(x) = \frac{1}{2}x^\top x = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2$$

so that

$$\nabla V(x(t)) = [x_1, x_2]^\top$$

and

$$\dot{V} = [x_1, x_2]\begin{bmatrix} x_2 \\ \tfrac{g}{\ell}\sin(x_1) - \tfrac{k}{m}x_2 + \tfrac{1}{m\ell^2}\Phi(x_1, x_2) \end{bmatrix}$$

and we would like $\dot{V} < 0$ to prove asymptotic stability (i.e., to show that the fuzzy controller can balance the pendulum). We have

$$x_2\left(x_1 + \frac{g}{\ell}\sin(x_1) - \frac{k}{m}x_2 + \frac{1}{m\ell^2}\Phi(x_1, x_2)\right) < -\beta$$

if for some fixed $\beta > 0$ (note that $x_2 \neq 0$)

$$x_1 + \frac{g}{\ell}\sin(x_1) - \frac{k}{m}x_2 + \frac{1}{m\ell^2}\Phi(x_1, x_2) < -\frac{\beta}{x_2}$$

Rearranging this equation, we see that we need

$$\Phi(x_1, x_2) \leq m\ell^2 \left(-\frac{\beta}{x_2} + \frac{k}{m} x_2 - x_1 - \frac{g}{\ell} \sin(x_1) \right)$$

on $x \in B(h)$ for some $h > 0$ and $\beta > 0$. As a graphical approach, we can plot the right-hand side of this equation, design the fuzzy controller $\Phi(x_1, x_2)$, and find $h > 0$ and $\beta > 0$ so that the given inequality holds and hence asymptotic stability holds.

We must emphasize that this is a local result. This means that we have shown that there exists an h and hence a ball $B(h)$ such that if we start our initial conditions in this ball (i.e., $x(0) \in B(h)$), then the fuzzy controller will balance the pendulum. The theory does not say how large h is; hence, it can be very small so that you may have to start the initial condition very close to the vertical equilibrium point for it to balance.

Application of Lyapunov's Indirect Method

For Equation (4.5)

$$\bar{A} = \left[\begin{array}{cc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{array} \right] \Bigg|_{x=0} = \left[\begin{array}{cc} 0 & 1 \\ \frac{g}{\ell} + \frac{1}{m\ell^2} \frac{\partial T}{\partial x_1} & -\frac{k}{m} + \frac{1}{m\ell^2} \frac{\partial T}{\partial x_2} \end{array} \right] \Bigg|_{x=0} \tag{4.6}$$

The eigenvalues of $\bar{A}$ are given by the determinant of $\lambda I - \bar{A}$. To ensure that the eigenvalues λ_i, $i = 1, 2$, of $\bar{A}$ are in the left half of the complex plane, it is sufficient that

$$\lambda^2 + \left(\frac{k}{m} - \frac{1}{m\ell^2} \frac{\partial T}{\partial x_2} \right) \lambda + \left(-\frac{g}{\ell} - \frac{1}{m\ell^2} \frac{\partial T}{\partial x_1} \right) = 0 \tag{4.7}$$

where $x = 0$, has its roots in the left half-plane. Equation (4.7) will have its roots in the left half-plane if each of its coefficients are positive. (Why?) Hence, if we substitute the values of the model parameters, we need

$$\begin{array}{l} \frac{\partial T}{\partial x_2}\Big|_{x=0} < k\ell^2 = 0.5, \\ \frac{\partial T}{\partial x_1}\Big|_{x=0} < g\ell m = -9.81 \end{array} \tag{4.8}$$

to ensure asymptotic stability.

Using simple intuitive knowledge about the dynamics of the inverted pendulum, we can design a fuzzy controller that uses triangular membership functions and 25 rules that meets these constraints. Rather than provide the details of the development of the fuzzy controller, which differs slightly from the one introduced in Section 4.2.1 on page 190, we would simply like to emphasize that any fuzzy controller that satisfies the above conditions in Equation (4.8) will result in a stable closed-loop system (we have designed and tested one). One easy way to design

such a fuzzy controller is to construct one in the usual way (i.e., heuristically) then plot the controller surface and check that the above conditions in Equation (4.8) are met (by simply inspecting the plot to make sure that its slope near zero (i.e., $x = 0$) satisfies the above constraints).

Finally, it is important that the reader not overgeneralize the stability result that is obtained via Lyapunov's indirect method. For the pendulum, Lyapunov's indirect method simply says that if the pendulum position begins close enough to the inverted position, the fuzzy controller will be guaranteed to balance it in the upright position; whereas if the pendulum starts too far from the balanced condition, the fuzzy controller may not balance it.

4.3.5 Example: The Parallel Distributed Compensator

While in the remainder of this chapter we consider nonlinear analysis of fuzzy control systems where the fuzzy controller is the "standard" one, in this one subsection we consider the case where the plant and controller are Takagi-Sugeno fuzzy systems. While here we will study the continuous-time case, in Exercise 4.6 on page 227 we will focus on the discrete-time case.

Plant, Controller, and Closed-Loop System

In particular, consider the plant introduced in Chapter 2, Section 2.3.7, in Equation (2.23) on page 76, where if we let $z(t) = x(t)$ we have

$$\dot{x}(t) = \left(\sum_{i=1}^{R} A_i \xi_i(x(t)) \right) x(t) + \left(\sum_{i=1}^{R} B_i \xi_i(x(t)) \right) u(t) \tag{4.9}$$

which is a Takagi-Sugeno fuzzy system. We could let $u(t) = 0$, $t \geq 0$ and study the stability of Equation (4.9). Instead, we will consider the case where we use a controller to generate $u(t)$.

Assume that we can measure $x(t)$ and that the controller is another Takagi-Sugeno fuzzy system with R rules (the same number of rules as was used to describe the plant) of the form

$$\textbf{If } \tilde{x}_1 \text{ is } \tilde{A}_1^j \textbf{ and } \tilde{x}_2 \text{ is } \tilde{A}_2^k \textbf{ and}, \ldots, \textbf{and } \tilde{x}_n \text{ is } \tilde{A}_n^l \textbf{ Then } u^i = K_i x(t)$$

where K_i, $i = 1, 2, \ldots, R$, are $1 \times n$ vectors of control gains and the premises of the rules are identical to the premises of the plant rules that were used to specify Equation (4.9). In this case

$$u(t) = \sum_{j=1}^{R} K_j \xi_j(x(t)) x(t) \tag{4.10}$$

This controller is sometimes referred to as a "parallel distributed compensator" since some think of the i^{th} rule in the controller as controlling the i^{th} rule of the plant, as this may be how you think of the design methodology for the K_i

gains. In this context some people think of the compensator as being "parallel" and "distributed" since the rules can be viewed as parallel and since some view the construction of the gain K_i as separate from the construction of the gain K_j where $i \neq j$. We will, however, see below that the construction of each K_i gain is not necessarily independent of the construction of other K_j gains, $i \neq j$.

If we connect the controller to the plant in Equation (4.9), we get a closed-loop system

$$\dot{x}(t) = \left(\sum_{i=1}^{R} A_i \xi_i(x(t)) + \left(\sum_{i=1}^{R} B_i \xi_i(x(t)) \right) \left(\sum_{j=1}^{R} K_j \xi_j(x(t)) \right) \right) x(t) \qquad (4.11)$$

which is in the form of Equation (4.1). We assume that μ_i and hence ξ_i are defined so that Equation (4.11) possesses a unique solution that is continuously dependent on $x(0)$.

Stability Analysis

For stability analysis we use the direct method of Lyapunov. Choose a (quadratic) Lyapunov function

$$V(x) = x^\top P x$$

where P is a "positive definite matrix" (denoted by $P > 0$) that is symmetric (i.e., $P = P^\top$). Given a symmetric matrix P we can easily test if it is positive definite. You simply find the eigenvalues of P, and if they are all strictly positive, then P is positive definite. If P is positive definite, then for all $x \neq 0$, $x^\top P x > 0$. Hence, we have $V(x) > 0$ and $V(x) = 0$ only if $x = 0$. Also, if $|x| \to \infty$, then $V(x) \to \infty$.

To show that the equilibrium $x = 0$ of the closed-loop system in Equation (4.11) is globally asymptotically stable, we need to show that $\dot{V}(x) < 0$ for all x. Notice that

$$\dot{V}(x) = x^\top P \dot{x} + \dot{x}^\top P x$$

so that since

$$\xi_i(x(t)) = \frac{\mu_i(x(t))}{\sum_{i=1}^{R} \mu_i(x(t))}$$

we have

$$\dot{V}(x) = x^\top P \left[\frac{\sum_{i=1}^{R} A_i \mu_i(x(t))}{\sum_{i=1}^{R} \mu_i(x(t))} + \left(\frac{\sum_{i=1}^{R} B_i \mu_i(x(t))}{\sum_{i=1}^{R} \mu_i(x(t))} \right) \left(\frac{\sum_{j=1}^{R} K_j \mu_j(x(t))}{\sum_{j=1}^{R} \mu_j(x(t))} \right) \right] x$$
$$+ x^\top \left[\frac{\sum_{i=1}^{R} A_i \mu_i(x(t))}{\sum_{i=1}^{R} \mu_i(x(t))} + \left(\frac{\sum_{i=1}^{R} B_i \mu_i(x(t))}{\sum_{i=1}^{R} \mu_i(x(t))} \right) \left(\frac{\sum_{j=1}^{R} K_j \mu_j(x(t))}{\sum_{j=1}^{R} \mu_j(x(t))} \right) \right]^\top P x$$

$$
\begin{aligned}
&= x^\top P \left[\frac{\sum_{i=1}^R A_i \mu_i(x(t)) \sum_{j=1}^R \mu_j(x(t))}{\sum_{i=1}^R \mu_i(x(t)) \sum_{j=1}^R \mu_j(x(t))} + \left(\frac{\sum_{i=1}^R B_i \mu_i(x(t))}{\sum_{i=1}^R \mu_i(x(t))} \right) \left(\frac{\sum_{j=1}^R K_j \mu_j(x(t))}{\sum_{j=1}^R \mu_j(x(t))} \right) \right] x \\
&\quad + x^\top \left[\frac{\sum_{i=1}^R A_i \mu_i(x(t)) \sum_{j=1}^R \mu_j(x(t))}{\sum_{i=1}^R \mu_i(x(t)) \sum_{j=1}^R \mu_j(x(t))} \right. \\
&\quad \left. + \left(\frac{\sum_{i=1}^R B_i \mu_i(x(t))}{\sum_{i=1}^R \mu_i(x(t))} \right) \left(\frac{\sum_{j=1}^R K_j \mu_j(x(t))}{\sum_{j=1}^R \mu_j(x(t))} \right) \right]^\top Px
\end{aligned}
$$

Now, if we let $\sum_{i,j}$ denote the sum over all possible combinations of i and j, $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, R$, we get

$$
\begin{aligned}
\dot{V}(x) &= x^\top P \left[\frac{\sum_{i,j} A_i \mu_i(x(t)) \mu_j(x(t))}{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t))} + \frac{\sum_{i,j} B_i K_j \mu_i(x(t)) \mu_j(x(t))}{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t))} \right] x \\
&\quad + x^\top \left[\frac{\sum_{i,j} A_i \mu_i(x(t)) \mu_j(x(t))}{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t))} + \frac{\sum_{i,j} B_i K_j \mu_i(x(t)) \mu_j(x(t))}{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t))} \right]^\top Px \\
&= x^\top P \left[\frac{\sum_{i,j} (A_i + B_i K_j) \mu_i(x(t)) \mu_j(x(t))}{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t))} \right] x \\
&\quad + x^\top \left[\frac{\sum_{i,j} (A_i + B_i K_j) \mu_i(x(t)) \mu_j(x(t))}{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t))} \right]^\top Px \\
&= x^\top \left[P \left[\frac{\sum_{i,j} (A_i + B_i K_j) \mu_i(x(t)) \mu_j(x(t))}{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t))} \right] \right. \\
&\quad \left. + \left[\frac{\sum_{i,j} (A_i + B_i K_j) \mu_i(x(t)) \mu_j(x(t))}{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t))} \right]^\top P \right] x \\
&= x^\top \left[\frac{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t)) \left[P(A_i + B_i K_j) + (A_i + B_i K_j)^\top P \right]}{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t))} \right] x
\end{aligned}
$$

Now, since

$$
0 \leq \frac{\mu_i(x(t)) \mu_j(x(t))}{\sum_{i,j} \mu_i(x(t)) \mu_j(x(t))} \leq 1
$$

we have

$$
\dot{V}(x) \leq \sum_{i,j} x^\top \left(P(A_i + B_i K_j) + (A_i + B_i K_j)^\top P \right) x
$$

Hence, if

$$
x^\top \left(P(A_i + B_i K_j) + (A_i + B_i K_j)^\top P \right) x < 0 \tag{4.12}
$$

then $\dot{V}(x) < 0$.

Let

$$Z = P(A_i + B_i K_j) + (A_i + B_i K_j)^\top P$$

Notice that since P is symmetric Z is symmetric so that $Z^\top = Z$. Equation (4.12) holds if Z is a "negative definite matrix." For a symmetric matrix Z, we say that it is negative definite (denoted $Z < 0$) if $x^\top Z x < 0$ for all $x \neq 0$. If Z is symmetric, then it is negative definite if the eigenvalues of Z are all strictly negative. Hence, to show that the equilibrium $x = 0$ of Equation (4.11) is globally asymptotically stable, we must find a *single* $n \times n$ positive definite matrix P such that

$$P(A_i + B_i K_j) + (A_i + B_i K_j)^\top P < 0 \tag{4.13}$$

for all $i = 1, 2, \ldots, R$ and $j = 1, 2, \ldots, R$.

Notice that in Equation (4.13) finding the common P matrix such that the R^2 matrices are negative definite is not trivial to compute by hand if n and R are large. Fortunately, "linear matrix inequality" (LMI) methods can be used to find P if it exists, and there are functions in a Matlab toolbox for solving LMI problems. If, however, via these methods there does not exist a P, this does not mean that there does not exist a Takagi-Sugeno fuzzy controller that can stabilize the plant; it simply means that the quadratic Lyapunov function approach (i.e., our choice of $V(x)$ above) did not lead us to find one. If you pick a different Lyapunov function, you may be able to find a Takagi-Sugeno controller that will stabilize the plant. It is in this sense that Lyapunov techniques often are called "conservative" in that conditions can often be relaxed beyond what the Lyapunov method would say for a given Lyapunov function and stability is still maintained. This does not, however, give us the license to ignore the conditions set up by the Lyapunov method; it simply is something that the designer must keep in mind in designing stable controllers with a Lyapunov method.

In our use of the Lyapunov method for constructing a Takagi-Sugeno fuzzy controller, it is evident that the overall approach must be conservative due partially to the use of the quadratic Lyapunov function and also since the stability test in Equation (4.13) depends in no way on the membership functions that are chosen for the plant representation that are used in the controller. In other words the results indicate that no matter what membership functions are used to represent the plant, and these are what allow for the modeling of nonlinear behavior, the stability test is the same. In this sense the test is for all possible membership functions that can be used. Clearly, then, we are not exploiting all of the known nonlinear structure of the plant and hence we are opening the possibility that the resulting stability analysis is conservative.

Regardless of the conservativeness, the above approach to controller construction and stability analysis can be quite useful for practical applications where you may ignore the stability analysis and simply use the type of controller that the method suggests (i.e., the controller in Equation (4.10) that is a nonlinear inter-

polation between R linear controllers). We will discuss the use of this controller in more detail in Chapter 7, Section 7.2.2, when we discuss gain scheduling since you can view the Takagi-Sugeno fuzzy controller as a nonlinear interpolator between R linear controllers for R linear plants represented by the Takagi-Sugeno model of the plant.

Simple Stability Analysis Example

As a simple example of how to use the stability test in Equation (4.13), assume that $n = 1$, $R = 2$, $A_1 = -1$, $B_1 = 2$, $A_2 = -2$, and $B_2 = 1$. These provide the parameters describing the plant. We do not provide the membership functions as any that you choose (provided that they result in a differential equation with a unique solution that depends continuously on $x(0)$) will work for the stability analysis that we provide.

Equation (4.13) says that to stabilize the plant with the Takagi-Sugeno fuzzy controller in Equation (4.10), we need to find a scalar $P > 0$ and gains K_1 and K_2 such that

$$\begin{gathered} P(-1+2K_1)+(-1+2K_1)P<0 \\ P(-1+2K_2)+(-1+2K_2)P<0 \\ P(-2+K_1)+(-2+K_1)P<0 \\ P(-2+K_2)+(-2+K_2)P<0 \end{gathered}$$

Choose any $P > 0$ such as $P = 0.5$. The stability test indicates that we need K_1 and K_2 such that $K_1 < 0.5$ and $K_2 < 2$ to get a globally asymptotically stable equilibrium $x = 0$. If you simulated the closed-loop system for some $x(0) \neq 0$, you would find that $x \to 0$ as $t \to \infty$.

4.4 Absolute Stability and the Circle Criterion

In this section we will examine the use of the Circle Criterion for testing and designing to ensure the stability of a fuzzy control system. The methods of this section provide an alternative (to the ones described in the previous section) for when the closed-loop system is in a special form to be defined next.

4.4.1 Analysis of Absolute Stability

Figure 4.5 shows a basic regulator system. In this system $G(s)$ is the transfer function of the plant and is equal to $C(sI - A)^{-1}B$ where (A, B, C) is the state variable description of the plant (x is the n-dimensional state vector). Furthermore, (A, B) is controllable and (A, C) is observable [54]. The function $\Phi(t, y)$, represents a memoryless, possibly time-varying nonlinearity—in our case, the fuzzy controller. Here, the fuzzy controller does not change with time, so we denote it by $\Phi(y)$. Even though the fuzzy controller is in the feedback path rather than the feed-forward

path in this system, we will be able to use the same SISO fuzzy controller described in Section 4.2.1 since it represents an odd function (i.e., for our illustrative example with the SISO fuzzy controller $\Phi(-y) = -\Phi(y)$ so we can transform Figure 4.5 into Figure 4.2). It is assumed that $\Phi(y)$ is piecewise continuous in t and locally Lipschitz [141].

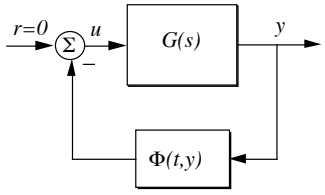


FIGURE 4.5 Regulator system.

If Φ is bounded within a certain region as shown in Figure 4.6 so that there exist α, β, a, b, ($\beta > \alpha$, $a < 0 < b$) for which

$$\alpha y \leq \Phi(y) \leq \beta y \tag{4.14}$$

for all $t \geq 0$ and all $y \in [a, b]$ (i.e., it fits between two lines that pass through zero) then Φ is said to be a "sector nonlinearity" or it is said to "lie on a sector." If Equation (4.14) is true for all $y \in (-\infty, \infty)$, then the sector condition holds globally; and if certain conditions hold (to be listed below), the system is "absolutely stable" (i.e., $x = 0$ is (uniformly) globally asymptotically stable). For the case where Φ only satisfies Equation (4.14) locally (i.e., for some a and b), if certain conditions (to be listed below) are met, then the system is "absolutely stable on a finite domain" (i.e., $x = 0$ is asymptotically stable).

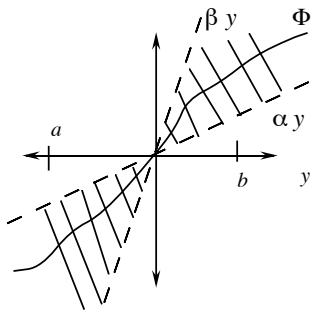


FIGURE 4.6 Sector-bounded nonlinearity.

Recall that in Section 4.2.1 we explained how the fuzzy controller is often similar to a saturation nonlinearity. Clearly, the fuzzy controller can be sector-bounded in the same manner as the saturation nonlinearity with either $\alpha = 0$ for the global case, or for local stability, with some $\alpha > 0$. To see this, consider how you would bound the plot of the fuzzy controller input-output map in Figure 4.3 on page 192 with two lines as shown in Figure 4.6.

Last, we define $D(\alpha, \beta)$ to be a closed disk in the complex plane whose diameter is the line segment connecting the points $-\frac{1}{\alpha} + j0$ and $-\frac{1}{\beta} + j0$. A picture of this disk is shown in Figure 4.7.

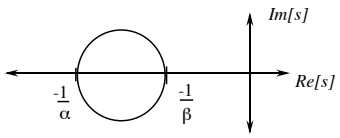


FIGURE 4.7 Disk.

Circle Criterion: With Φ satisfying the sector condition in Equation (4.14), the regulator system in Figure 4.5 is absolutely stable if one of the following three conditions is met:

1. If $0 < \alpha < \beta$, the Nyquist plot of $G(j\omega)$ is bounded away from the disk $D(\alpha, \beta)$ and encircles it m times in the counterclockwise direction where m is the number of poles of $G(s)$ in the open right half-plane.

2. If $0 = \alpha < \beta$, $G(s)$ is Hurwitz (i.e., has its poles in the open left half plane) and the Nyquist plot of $G(j\omega)$ lies to the right of the line $s = -\frac{1}{\beta}$.

3. If $\alpha < 0 < \beta$, $G(s)$ is Hurwitz and the Nyquist plot of $G(j\omega)$ lies in the interior of the disk $D(\alpha, \beta)$ and is bounded away from the circumference of $D(\alpha, \beta)$.

If Φ satisfies Equation (4.6) only on the interval $y \in [a, b]$ (i.e., it only lies between the two lines in a region around zero), then the above conditions ensure absolute stability on a finite domain. It is important to note that the above conditions are only sufficient conditions for stability and hence there is the concern that they are conservative. In [223] it is shown how the circle criterion can be adjusted such that the conditions are sufficient and necessary in a certain way. We introduce these next.

It is necessary to begin by providing some mathematical preliminaries. For each real $p \in [1, \infty)$, the set L_p consists of functions $f(\cdot) : [0, \infty) \to \Re$ such that

$$\int_0^\infty |f(t)|^p \, dt < \infty \tag{4.15}$$

For instance, if $f(t) = e^{-t}$, then we can say that $f(t) \in L_1$. The set L_∞ denotes the set of all functions $f(t)$ such that $\sup_t\{f(t)\} < \infty$ (i.e., the set of all bounded

functions). Clearly, $e^{-t} \in L_\infty$ also.

Let

$$f_T(t) = \begin{cases} f(t), & 0 \le t \le T \\ 0, & T < t \end{cases} \tag{4.16}$$

be a truncated version of $f(t)$. Let the set L_{pe}, the *extension of* L_p, consist of all functions $f_T : [0, \infty) \to \Re$, such that $f_T \in L_p$ for all finite T. Finally, let

$$\| f(\cdot) \|_p = \left[\int_0^\infty |f(t)|^p \, dt \right]^{1/p}$$
$$\| f(\cdot) \|_{Tp} = \| f(\cdot)_T \|_p$$

If $\bar{R}$ is a binary relation on L_{pe}, then $\bar{R}$ is said to be L_p-stable if

$$(x, y) \in \bar{R},\ x \in L_p \Rightarrow y \in L_p \tag{4.17}$$

For example, if x is the input to a system and y is the output, this quantifies a type of input-output stability. $\bar{R}$ is "L_p-stable with finite gain" if it is L_p-stable, and in addition there exist finite constants γ_p and b_p such that

$$(x, y) \in \bar{R},\ x \in L_p \Rightarrow \|y\|_p \le \gamma_p \|x\|_p + b_p \tag{4.18}$$

$\bar{R}$ is "L_p-stable with finite gain and zero bias" if it is L_p-stable, and in addition there exists a finite constant γ_p such that

$$(x, y) \in \bar{R},\ x \in L_p \Rightarrow \|y\|_p \le \gamma_p \|x\|_p \tag{4.19}$$

Assume that we are given the regulator system shown in Figure 4.5 (with G defined as above) except that now Φ is in general defined by $\Phi : L_{2e} \to L_{2e}$ (more general than above so it can still represent a fuzzy controller). Φ belongs to the open sector (α, β) if it belongs to the sector $[\alpha + \epsilon, \beta - \epsilon]$ for some $\epsilon > 0$ with the sector bound defined as

$$\|\Phi x - [(\beta + \alpha)/2]x\|_{T2} \le \tfrac{(\beta - \alpha)}{2} \|x\|_{T2}, \text{for all } T \ge 0, \text{for all } x \in L_{2e} \tag{4.20}$$

In actuality, this definition of the sector $[\alpha, \beta]$ is the same as our previous definition in Equation (4.14) if Φ is memoryless (i.e., it has no dynamics, and it does not use past values of its inputs, only its current input). Hence, since we have a memoryless fuzzy controller, we can use the sector condition from Equation (4.14). Next, we state a slightly different version of the circle criterion that we will call the circle criterion with sufficient and necessary conditions (SNC).

Circle Criterion with Sufficient and Necessary Conditions (SNC): For the system of Figure 4.5 with Φ defined as $\Phi : L_{2e} \to L_{2e}$, which satisfies Equation (4.20), and α, β two given real numbers with $\alpha < \beta$, the following two statements are equivalent [223]:

1. The feedback system is L_2-stable with finite gain and zero bias for every Φ belonging to the sector (α, β).
2. The transfer function G satisfies one of the following conditions as appropriate:
   (a) If $\alpha\beta > 0$, then the Nyquist plot of $G(j\omega)$ does not intersect the interior of the disk $D(\alpha, \beta)$ and encircles the interior of the disk $D(\alpha, \beta)$ exactly m times in the counterclockwise direction, where m is the number of poles of G with positive real part.
   (b) If $\alpha = 0$, then G has no real poles with positive real part, and $\text{Re}[G(j\omega)] \geq -\frac{1}{\beta}$ for all ω.
   (c) If $\alpha\beta < 0$, then G is a stable transfer function and the Nyquist plot of $G(j\omega)$ lies *inside* the disk $D(\alpha, \beta)$ for all ω.

If the conditions in statement 2 are satisfied, the system is L_2-stable and the result is similar to the circle criterion with sufficient conditions only. Negation of statement 2 infers negation of statement 1, and we can state that the system will not be L_2-stable for *every* nonlinearity in the sector (it may not be apparent which of the nonlinearities in a sector will cause the instability). Hence, if a given fuzzy control system does not satisfy any of the conditions of statement 2, then we do not know that it will result in an unstable system. All we know is that there is a way to define the fuzzy controller (perhaps one you would not pick) that will result in an unstable closed-loop system.

4.4.2 Example: Temperature Control

Suppose that we are given the thermal process shown in Figure 4.8, where τ_e is the temperature of a liquid entering the insulated chamber, τ_o is the temperature of the liquid leaving the chamber, and $\tau = \tau_o - \tau_e$ is the temperature difference due to the thermal process. The heater/cooling element input is denoted with q. The desired temperature is τ_d. Suppose that the plant model is

$$\frac{\tau(s)}{q(s)} = \frac{1}{s+2}$$

(note that we are slightly abusing the notation by showing τ as a function of the Laplace variable). Suppose that we wish to track a unit step input τ_d. We wish to design a stable fuzzy control system and would like to try to make the steady-state error go to zero.

Suppose that the control system that we use is shown in Figure 4.9. The controller $G_c(s)$ is a post compensator for the fuzzy controller. Suppose that we begin by choosing $G_c(s) = K = 2$ and that we simply consider this gain to be part of the plant. Furthermore, for the SISO fuzzy controller we use input membership functions shown in Figure 4.10 and output membership functions shown in Figure 4.11. Note that we denote the variable that is output from the fuzzy controller (and input to $G_c(s)$) as q'. We use 11 rules in the rule-base. For instance,

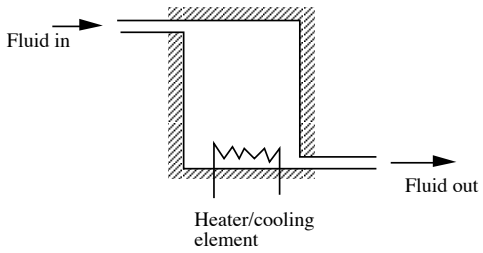


FIGURE 4.8 Thermal process.

- **If** e is positive small **Then** q is positive small
- **If** e is zero **Then** q is zero
- **If** e is negative big **Then** q is negative big

are rules in the rule-base (the others are similar in that they associate one fuzzy set on the input universe of discourse with one on the output universe of discourse). We use minimum to represent the premise and implication, singleton fuzzification, and COG defuzzification (different from our parameterized fuzzy controller in Section 4.2.1).

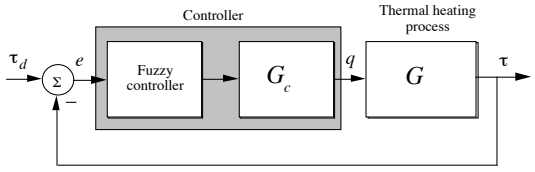


FIGURE 4.9 Thermal process control system.

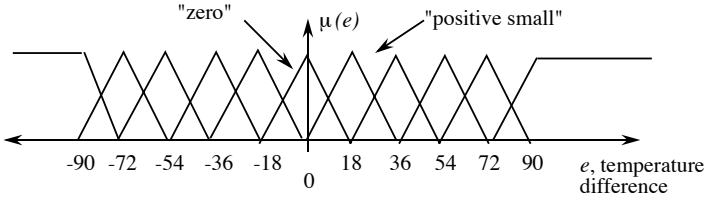


FIGURE 4.10 Input fuzzy sets.

A plot of the nonlinear surface for the fuzzy controller, which looks similar to a saturation nonlinearity, can be used to show that $\alpha = 0$ (it must be since the fuzzy controller output is saturated and the only line that will fit under the saturation

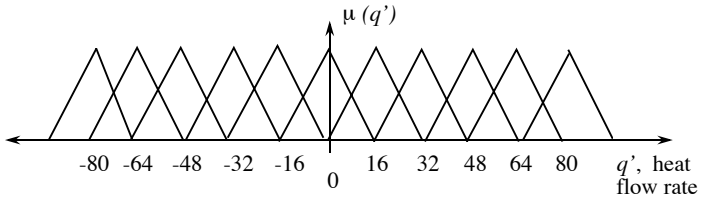


FIGURE 4.11 Output fuzzy sets.

is one with zero slope) and $\beta = \frac{4}{3}$ (to see this, plot the nonlinear surface and note that a line with a slope of $\frac{4}{3}$ overbounds the nonlinearity). The Nyquist plot of G_cG is in the right half-plane so that there are no encirclements (i.e., $m = 0$). Also, G_cG is Hurwitz since it has no right half-plane poles (including none on the $j\omega$ axis). Using the second condition of the circle criterion, we can conclude that the system is absolutely stable. If you were to pick some initial conditions on the state and let the reference input be zero, you could show in simulation that the state trajectories asymptotically decrease to zero.

It is interesting to note, however, that if you let the reference input be

$$\tau_d = 20u(t)$$

where $u(t)$ is the unit step function, then you would find a large steady-state error. Hence, we see that the guarantee for stability holds only for the case where $\tau_d = 0$.

If you would like to get rid of this steady-state error, one way to proceed would be to add an integrator (using standard ideas from conventional control). Suppose that we choose $G_c(s) = \frac{3}{s}$. With this choice G_cG is no longer Hurwitz, so the second condition of the circle criterion cannot be used. Using the plot of the fuzzy controller nonlinearity, we see that we can choose $\alpha = \frac{3}{4}$ and $\beta = \frac{4}{3}$ and the sector condition holds on a region $[-80, 80]$. Now, we consider the disk $D(-4/3, -3/4)$ and note that there are no encirclements of this disk (i.e., $m = 0$). Hence, by the first condition of the circle criterion we get absolute stability on a finite domain.

From this, if you were to do a simulation where the initial conditions were started sufficiently close to the origin and the reference input were equal to zero, then the state trajectories would asymptotically decrease to zero. It is interesting to note that if we choose $\tau_d = 20u(t)$ (i.e., a nonzero reference input) and a simulation is done, we would find that there would be no steady-state error. The theory above does not guarantee this; however, we will study how to guarantee that we will get zero steady-state error in the next section.

4.5 Analysis of Steady-State Tracking Error

A terrain-following and terrain-avoidance aircraft control system uses an altimeter to provide a measurement of the distance of the aircraft from the ground to decide how to steer the aircraft to follow the earth at a pilot-specified height. If a fuzzy

controller is employed for such an application, and it consistently seeks to control the height of the plane to be lower than what the pilot specifies, there will be a steady-state tracking error (an error between the desired and actual heights) that could result in a crash. In this section we will show how to use the results in [178] for predicting and eliminating steady-state tracking errors for fuzzy control systems so that problems of this sort can be avoided.

4.5.1 Theory of Tracking Error for Nonlinear Systems

The system is assumed to be of the configuration shown in Figure 4.1 on page 190 where r, e, u, and y belong to $L_{\infty e}$ and $\Phi(e)$ is the SISO fuzzy controller described in Section 4.2.1. We will call $e_{ss} = \lim_{t\to\infty} e(t)$ the steady-state tracking error. $G(s)$ has the form

$$G(s) = \frac{p(s)}{s^{\rho}q(s)} \tag{4.21}$$

where ρ, a nonnegative integer, is the number of poles of $G(s)$ at $s = 0$, and $p(s)$ and $s^{\rho}q(s)$ are relatively prime polynomials (i.e., they have no common factors) such that $\deg(p(s)) < \deg(s^{\rho}q(s))$. For example, if

$$G(s) = \frac{s+1}{s(s+2)}$$

then $\rho = 1$. Furthermore, we assume that $\Phi(0) = 0$, and Φ is bounded by α and β according to

$$\alpha \leq \frac{\Phi(a) - \Phi(b)}{a - b} \leq \beta \tag{4.22}$$

for all $a \neq b$. Notice that this sector bound is different from the sector bound in Equation (4.14). This new sector bound is determined by the maximum and minimum slopes of Φ at any point and is sometimes not as easy to determine as the graphical sector bound described in the last section. Finally, we assume that one of the three circle criterion conditions listed on page 207 is satisfied.

To predict the value of e_{ss}, we must make several definitions. First, we define an "average gain" for Φ, c_0, as

$$c_0 = \frac{1}{2}(\alpha + \beta)$$

and we assume that $c_0 \neq 0$. In [178] the authors show that for this c_0, $1 + c_0 G(s) \neq 0$ for $Re(s) \geq 0$. Therefore, the rational function

$$H(s) = \frac{G(s)}{1 + c_0 G(s)}$$

is strictly proper, and has no poles in the closed right half-plane. Defined in this manner, $H(s)$ is the closed-loop equation for the system shown in Figure 4.1 with c_0 as an average gain of Φ. Finally, we define

$$\tilde{\Phi}(e) = \Phi(e) - c_0 e$$

for all e. That is, $\tilde{\Phi}$ is the difference between the actual value of Φ at some point e and a predicted value found by using the average gain c_0.

Suppose that the above assumptions are met. It is proven in [178] that for each given real number γ, there exists a unique real number ξ such that

$$\gamma = \xi + H(0)\tilde{\Phi}(\xi) \tag{4.23}$$

where to find the value of ξ we use

$$\xi = \lim_{k \to \infty} \xi_k \tag{4.24}$$

where

$$\xi_{k+1} = \gamma - H(0)\tilde{\Phi}(\xi_k) \tag{4.25}$$

and ξ_0 is an arbitrary real number and γ is given (Equation (4.25) is an iterative algorithm that will be used to find e_{ss}). Furthermore, if we define c as

$$c = \frac{1}{2}(\beta - \alpha)|H(0)| \tag{4.26}$$

and assume that $c < 1$, then the equation

$$|\xi - \xi_k| \leq \frac{c^k}{1-c}|\xi_0 - \gamma + H(0)\tilde{\eta}(\xi_0)|, \quad k \geq 1 \tag{4.27}$$

must be true for the iterative algorithm, Equation (4.25), to converge.

Finally, suppose that we define $\Theta(\gamma) = \xi$ to represent the algorithm in Equation (4.25). Hence, Θ is given a γ, an arbitrary ξ_0 is chosen, c_0 and $H(0)$ are specified, then with the given fuzzy controller Φ, we let $\tilde{\Phi}(e) = \Phi(e) - c_0 e$, and Equation (4.25) is computed until k is large enough that $\xi_{k+1} - \xi_k$ is very small. The resulting converged value of ξ is the value of $\Theta(\gamma)$.

Tracking Error Theorem: Assuming that all the described assumptions are satisfied, then

1. If $r(t)$ approaches a limit l as $t \to \infty$, then $e_{ss} \equiv \lim_{t\to\infty} e(t)$ exists. Moreover, $e_{ss} \neq 0$ if and only if $l \neq 0$ and $\rho = 0$, and then $e_{ss} = \Theta(\gamma)$ where $\gamma = \frac{l}{1+c_0 G(0)}$.

2. Assuming that

$$r(t) = \sum_{j=0}^{\nu} a_j t^j, \; t \geq 0 \tag{4.28}$$

in which the a_j are real, ν is a positive integer, and $a_\nu \neq 0$, the following holds:

(a) e is unbounded if $\nu > \rho$.

(b) if $\nu \leq \rho$, then e approaches a limit as $t \to \infty$. If $\nu = \rho$, this limit is $e_{ss} = \Theta(\gamma)$ where

$$\gamma = \frac{a_\nu \nu! q(0)}{c_0 p(0)} \tag{4.29}$$

If $\nu < \rho$, then the limit is zero.

Notice that for Equation (4.28) if we want $r(t)$ to be a unit step, then $\nu = 0$ so $r(t) = a_0$, $t \geq 0$ and we choose $a_0 = 1$. If we want $r(t)$ to be a ramp of unit slope, then we choose $\nu = 1$ so that $r(t) = a_0 + a_1 t$ and we choose $a_0 = 0$ and $a_1 = 1$.

An examination of the above theorem reveals that in actuality the proposed method for finding the steady-state error for fuzzy control systems is similar to the equations used in conventional linear control systems. The theorem performs the function of identifying an appropriate equation for e_{ss} based on the type of input and the "system type." Notice that the two equations for γ in the theorem are analogous to the equations for the "error constants" [54], $1/(1 + K_p)$, $1/K_v$, and $1/K_a$, and provide an initial estimate for e_{ss}.

4.5.2 Example: Hydrofoil Controller Design

The HS Denison is an 80-ton hydrofoil stabilized via flaps on the main foils and the incidence of the aft foil. The transfer function for a linearized model of the plant that includes the foil and vehicle is

$$\frac{\theta(s)}{D(s)} = \frac{10^4}{s^2 + 60s + 10^4}$$

where $\theta(s)$ is the pitch angle and $D(s)$ is the command input. We wish to design a fuzzy controller that will maintain a constant deflection of the pitch angle with less than 1% steady-state error from the desired angle.

We first determine that if $\alpha = 0$, then β must be less than 1.56 for the circle criterion conditions to be satisfied. Therefore, our preliminary design for the SISO parameterized fuzzy controller will have $A = B = 1$. For this controller $\beta = 1$, $\alpha = 0$, and $c_0 = 0.5$. The other relevant values are $H(0) = 0.6667$ and $G(0) = 1$. If our input $r(t)$ is a step with magnitude 5.0, then we will use the first condition of the theorem and $\gamma = 3.3333$. Using these values in the iterative equation, we find that our steady-state error will be 4.0. This is a very large error and is obviously

much larger than 1%. Even with $\beta = 1.559$ we cannot meet the error requirement. Therefore, the system requirements cannot be met with a simple fuzzy controller. However, if we combine a simple fuzzy controller with an integrator, the circle criterion is satisfied as long as $B/A < 50$. Furthermore, $\rho = 1$ for this system and part 1 of the theorem predicts that $e_{ss} = 0$. Simulations for this system with $A = B = 1$ show that in fact $e_{ss} = 0$ and we have met the design criteria.

4.6 Describing Function Analysis

Autopilots used for cargo ship steering seek to achieve a smooth response by appropriately actuating the rudder to steer the ship. The presence of unwanted oscillations in the ship heading results in loss of fuel efficiency and a less comfortable ride. While such oscillations, which are closed periodic orbits in the state plane, sometimes called "limit cycles," result from certain inherent nonlinearities in the control loop, it is sometimes possible to carefully construct a controller so that such undesirable behavior is avoided.

In this section we will investigate the use of the describing function method for the prediction of the existence, frequency, amplitude, and stability of limit cycles. We will first present describing function theory following the format in [189]. Next, we will use several examples to show how describing function analysis can be used in the design of SISO and MISO fuzzy controllers of the form described in Section 4.2. Finally, we will use describing function analysis to design fuzzy controllers for an underwater vehicle and a tape drive servo.

4.6.1 Predicting the Existence and Stability of Limit Cycles

Before explaining the describing function method, we will discuss several assumptions that we will use in applying the techniques of this section.

Basic Assumptions

There are several assumptions that need to be satisfied for our purposes for the describing function method. These assumptions are as follows:

1. There is only a single nonlinear component and the system can be rearranged into the form shown in Figure 4.1 on page 190.
2. The nonlinear component is time-invariant.
3. Corresponding to a sinusoidal input $e(t) = \sin(\omega t)$, only the fundamental component $u_1(t)$ in the output $u(t)$ must be considered.
4. The nonlinearity Φ (which will represent the fuzzy controller) is an odd function.

The first assumption requires that nonlinearities associated with the plant or output sensors be rearranged to appear in Φ as shown in Figure 4.1. The second assumption originates from the use in this method of the Nyquist criterion, which can only be

applied to linear time-invariant systems. The third assumption implies that the linear component following the nonlinearity has characteristics of a low-pass filter so that

$$|G(j\omega)| \gg |G(nj\omega)| \text{ for } n = 2, 3, ... \tag{4.30}$$

and therefore the higher-frequency harmonics, as compared to the fundamental component, can be neglected in the analysis. This is the *fundamental assumption* of describing function analysis and represents an approximation as there normally will be higher-frequency components in the signal. The fourth assumption simplifies the analysis of the system by allowing us to neglect the static term of the Fourier expansion of the output.

We emphasize that due to the lack of perfect satisfaction of the above assumptions the resulting analysis is only approximate. Next, we introduce the tools and methods of describing function analysis.

Defining and Computing the Describing Function

For an input $e(t) = C\sin(\omega t)$ to the nonlinearity, $\Phi(e)$, there will be an output $u(t)$. This output will often be periodic though generally nonsinusoidal. Expanding this $u(t)$ into a Fourier series results in

$$u(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}[a_n \cos(n\omega t) + b_n \sin(n\omega t)] \tag{4.31}$$

The Fourier coefficients (a_i's and b_i's) are generally functions of C and ω and are determined by

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} u(t)d(\omega t) \tag{4.32}$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} u(t)\cos(n\omega t)d(\omega t) \tag{4.33}$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} u(t)\sin(n\omega t)d(\omega t) \tag{4.34}$$

Because of our assumptions $a_0 = 0$, $n = 1$, and

$$u(t) \approx u_1(t) = a_1\cos(\omega t) + b_1\sin(\omega t) = M(C,\omega)\sin(\omega t + \phi(C,\omega)) \tag{4.35}$$

where

$$M(C,\omega) = \sqrt{a_1^2 + b_1^2} \tag{4.36}$$

and where

$$\phi(C,\omega) = \arctan\left(\frac{a_1}{b_1}\right) \tag{4.37}$$

From the above equations we can see that the fundamental component of the output, corresponding to a sinusoidal input, is a sinusoid of the same frequency that can be written in complex representation as

$$u_1 = M(C,\omega)e^{j(\omega t+\phi(C,\omega))} = (b_1 + ja_1)e^{j\omega t} \tag{4.38}$$

We will now define the *describing function* of the nonlinear element to be the complex ratio of the fundamental component of the nonlinear element by the input sinusoid

$$N(C,\omega) = \frac{u_1}{C\sin(\omega t)} = \frac{M(C,\omega)e^{j(\omega t+\phi(C,\omega))}}{Ce^{j\omega t}} = \frac{1}{C}(b_1 + ja_1) \tag{4.39}$$

By replacing the nonlinear element $\Phi(e)$ with its describing function $N(C,\omega)$, the nonlinear element can be treated as if it were a linear element with a parameterized frequency response function.

Generally, the describing function depends on the frequency and amplitude of the input signal. However, for some special cases it does not depend on frequency. For example, if the nonlinearity is time-invariant and memoryless, $N(C,\omega)$ is real and frequency-independent. For this case, $N(C,\omega)$ is real because evaluating Equation (4.33) gives $a_1 = 0$. Furthermore, in the same equations, the integration of the single-valued function $u(t)\sin(\omega t) = [C\sin(\omega t)]\sin(\omega t)$ is done for the variable ωt, implying that ω does not explicitly appear in the integration and that the function $N(C,\omega)$ is frequency-independent.

There are several ways to compute describing functions. The describing function can be computed analytically if $u = \Phi(e)$ is known and the integrations to find a_1 and b_1 can be easily carried out. If the input-output relationship of $\Phi(e)$ is given by graphs or tables, then numerical integration can be used. The third method, and the one that we will use, is "experimental evaluation." We will excite the input of the fuzzy controller with sinusoidal inputs, save the related outputs, and then use the input and output waveforms to determine the gain and phase shift at the frequency of the input sinusoid. By varying the amplitude and frequency (or just the amplitude if the fuzzy controller is SISO, time-invariant, and memoryless) of the input sinusoid, we can find u_1 at several points and plot the corresponding describing function.

Predicting Limit Cycles

In Figure 4.1 on page 190, if we replace $\Phi(e)$ with $N(C,\omega)$ and assume that a self-sustained oscillation of amplitude C and frequency ω exists in the system, then for

$r = 0$, $y \neq 0$, and

$$G(j\omega)N(C,\omega) + 1 = 0 \tag{4.40}$$

This equation, sometimes called the "harmonic balance equation," can be rewritten as

$$G(j\omega) = -\frac{1}{N(C,\omega)} \tag{4.41}$$

If any limit cycles exist in our system, and the four basic assumptions outlined above are satisfied, then the amplitude and frequency of the limit cycles can be predicted by solving the harmonic balance equation. If there are no solutions to the harmonic balance equation, then the system will have no limit cycles (under the above assumptions).

However, solving the harmonic balance equation is not trivial; for higher-order systems, the analytical solution is very complex. The usual method, therefore, is to plot $G(j\omega)$ and $-1/N(C,\omega)$ on the same graph and find the intersection points. For each intersection point, there will be a corresponding limit cycle. The amplitude and frequency of each limit cycle can then be determined by finding the particular C and ω that give the value of $-1/N(C,\omega)$ and $G(j\omega)$ at the intersection point.

Along with the amplitude and frequency of the limit cycles, we also would like to determine whether the limit cycles are stable or unstable. A limit cycle is considered stable if system trajectories move to the limit cycle when they start within a certain neighborhood of it. Therefore, once the system is in a limit cycle, the system will return to the limit cycle when perturbations move the system off of the limit cycle. For an unstable limit cycle, there is no neighborhood within which the system trajectory moves to the limit cycle when the system trajectory starts near it. Instead, the trajectory will move away from the limit cycle. Therefore, if a system is perturbed from an unstable limit cycle, the oscillations will either die out, increase until the system goes unstable, or move to a stable limit cycle. The stability of limit cycles can be determined from the same plot used to predict the existence of the limit cycles. A summary of the above conclusions is given by the following criterion from [189].

Limit Cycle Criterion: Each intersection point of the $G(j\omega)$ and $-1/N(C,\omega)$ curves corresponds to a limit cycle. In particular, if the curves intersect, we predict that there will be a limit cycle in the closed-loop system with amplitude C and frequency ω. If points near the intersection and along the increasing-C side of the curve $-1/N(C,\omega)$ are not encircled by the curve $G(j\omega)$, then the corresponding limit cycle is stable. Otherwise, the limit cycle is unstable.

In the next two subsections we will show how to use this criterion to test for the existence, amplitude, frequency, and stability of limit cycles. Also, we will show how it can be used in the redesign of the fuzzy controller to eliminate limit cycles.

4.6.2 SISO Example: Underwater Vehicle Control System

We wish to design a fuzzy controller for the direction control system of an underwater vehicle described in [45]. The electrically controlled rudder and an added compensator have transfer function

$$\frac{C(s)}{R(s)} = \frac{s+0.1}{s(s+5)^2(s+0.001)}$$

We must design the fuzzy controller such that there are no limit cycles possible within the closed-loop system.

Our fuzzy controller will be SISO, odd, time-invariant, and memoryless. Therefore, we know that the describing function will be real-valued and can only intersect the Nyquist plot of $G(j\omega)$ along the real axis. Examining a Nyquist plot of $G(j\omega)$ for this system, we find that it intersects the real axis at one point only, $-0.0042 + j0$ (an enlargement of this plot is shown in Figure 4.12, where $-1/N(C,\omega)$ is on top of the horizontal axis) and hence a limit cycle exists (you can simulate the closed-loop system to illustrate this). To avoid intersecting this point, we must construct the fuzzy controller so that $-1/N(C,\omega) < -0.0042$ or $N(C,\omega) < 240.0528$ for all values of C. For the type of fuzzy controller we are using, this criterion can be achieved if $B/A < 240.0528$. We will choose $A = 2$ and $B = 200$. The resulting describing function is shown in Figure 4.13. Since the largest value of $-1/N(C,\omega) = -2/200 = -0.01$ is less than -0.0042, there is no solution to the harmonic balance equation, and the approximate analysis indicates that the existence of a limit cycle is unlikely. A simulation of this system for $r = 5$ is shown in Figure 4.14. No limit cycles exist, and our design was successful.

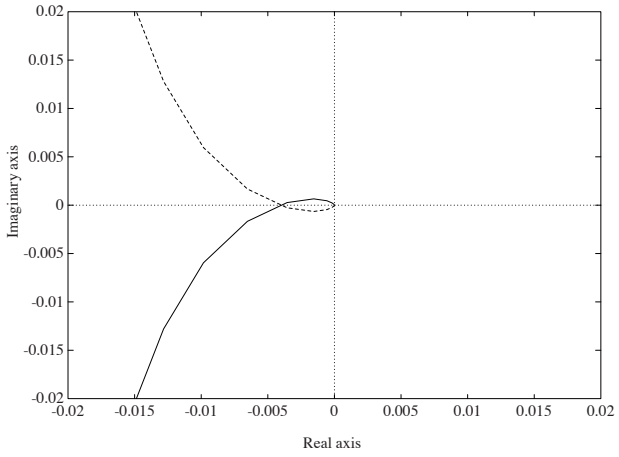


FIGURE 4.12 Plot of $G(j\omega)$ for the underwater vehicle (figure taken from [83], © John Wiley and Sons).

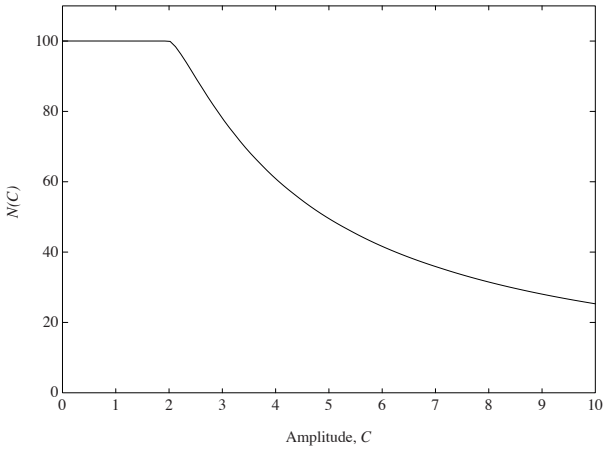


FIGURE 4.13 Describing function for fuzzy controller with $A = 2$ and $B = 200$ (figure taken from [83], © John Wiley and Sons).

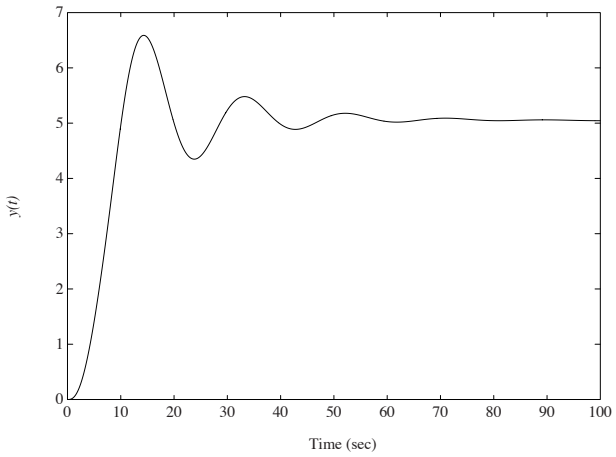


FIGURE 4.14 Simulation of the underwater vehicle (figure taken from [83], © John Wiley and Sons).

4.6.3 MISO Example: Tape Drive Servo

The describing function analysis of the previous design example was for SISO fuzzy controllers whose describing functions are not dependent on ω. However, it is impor-

tant that we also examine how this type of analysis can be applied to MISO fuzzy controllers. While for a MISO fuzzy controller the basic theory is still the same, there are several differences in determining and using the describing function. First, the describing function will be dependent on both C and ω. Because of this, when we experimentally determine $N(C, \omega)$, we have to find not only the amplitude of the fundamental frequency of the output waveform but also the phase of the fundamental frequency for inputs of different amplitude and frequency. Methods for doing this can be found in [13]. This also means that there will be more lines to plot as we will have to plot $-1/N(C, \omega)$ as C changes for each value of ω so that there will be a curve for each value of ω for which $N(C, \omega)$ is calculated. Second, not all intersections of $G(j\omega)$ and $-1/N(C, \omega)$ will be limit cycles. For an intersection to predict a limit cycle, the values of ω for $G(j\omega)$ and $-1/N(C, \omega)$ at the intersection must be the same. We can see that, as would be expected, the limit cycle prediction procedure using describing functions is slightly more complex for MISO systems. However, with the adjustments mentioned above, the procedure follows the same format as before. This will be shown in the following design example.

We will design a fuzzy controller for a tape drive servo described in [54] with transfer function

$$G(s) = \frac{15s^2 + 13.5s + 12}{(s+1)(s^2 + 1.1s + 1)}$$

Also included in the system is a precompensator of the form $C(s) = (s + 20)/s$. It is desired that a step input of current to the drive mechanism will cause the tape to have a stable velocity. To analyze the system for limit cycles, we will choose a fuzzy controller, empirically find the describing function, search for solutions to the harmonic balance equation, and then redesign the fuzzy controller if necessary.

We will begin by choosing $A = 100$, $B = 600$, and $D = 50$. Next, we will find $N(C, \omega)$ for $0 \leq C \leq 100$ and $\omega = 0.5$, 1, 10, 50, 100, and 500. The resulting plot of $G(j\omega)$ and $-1/N(C, \omega)$ is shown in Figure 4.15. There are no intersection points and therefore no predicted limit cycles. By simulating the system with the chosen values of A, B, and D and $r = 12$, we verify that no limit cycles exist. This simulation is shown in Figure 4.16.

4.7 Limitations of the Theory

It is important to note that there are limitations to the approaches that we covered in this chapter in addition to the general ones outlined in Section 4.1 on page 187, which included the following:

- The model of a physical process is never perfectly accurate, and since the mathematical analysis is based on the model, the analysis is of limited accuracy for the physical system. The more accurate the model, the more accurate the conclusions from the mathematical analysis as they pertain to the real physical system.
- Fuzzy control tends to show its greatest advantages for processes that are very

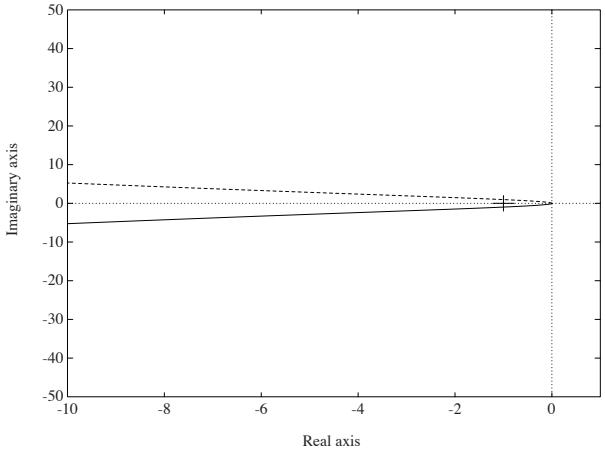


FIGURE 4.15 Plot of $G(j\omega)$ and $-1/N(C,\omega)$ for $A = 100$, $B = 600$, and $D = 50$ (figure taken from [83], © John Wiley and Sons).

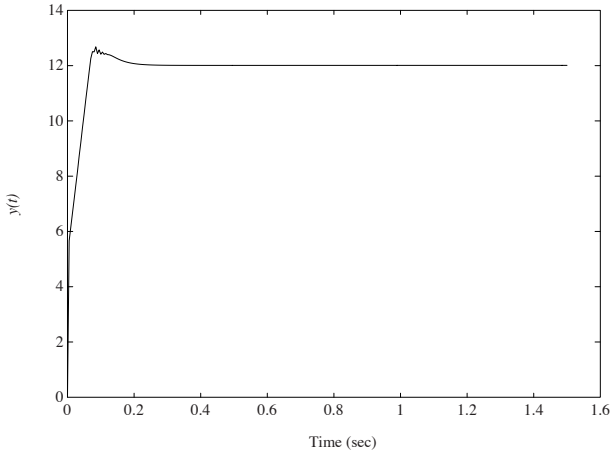


FIGURE 4.16 Simulation of the tape drive servo and fuzzy controller (figure taken from [83], © John Wiley and Sons).

complex in terms of nonlinearities, stochastic influences, process uncertainties, and so on. The mathematical analysis tools that are available often do not apply to very complex processes as the needed assumptions are often not satisfied. There is then an inherent limitation of the mathematical analysis tools due to the need

for such tools for any nonlinear control systems, let alone fuzzy control systems.

Next, we provide a more detailed overview of some additional limitations to the approaches covered in this chapter. In general, except for Lyapunov's methods, discussed in Section 4.3, we have examined only linear plant models or nonlinear plants that can be manipulated to be in the form of Figure 4.5. In Section 4.3 some of the stability conditions are often conservative, which means that if the conditions for stability are not met, the system could still be stable. Indeed, the results for the circle criterion have often been found to be too conservative. While the results for absolute stability, steady-state tracking error, and describing functions can certainly be applied to models linearized about operating points in a nonlinear system, such results are only local in nature. Furthermore, we have limited ourselves throughout the entire chapter (except Section 4.3) to SISO and MISO fuzzy controllers. In addition to these general limitations, there are also limitations specific to each section. In the section on absolute stability, we have only examined the SISO fuzzy controller and not the MISO case (of course, extension to the multivariable case is not difficult using, for example, the development in [90]). Furthermore, although the circle criterion conditions are sufficient and necessary, the necessary conditions are for a class of nonlinearities and do not identify which of the nonlinearities (i.e., which fuzzy controller) within the class will cause the system to become unstable. There is currently no theory for the tracking error analysis of multivariable nonlinear systems. Our describing function technique, even though it can be applied to SISO and MISO fuzzy controllers and certain nonlinear plant models, is limited by the fact that the use of the approach for more than three inputs to the fuzzy controller becomes prohibitive. There has been some work on the expansion of the theory of nonlinear analysis to a wider class of nonlinear plants where a mathematical characterization of the fuzzy controller is used (see, for example, [106, 105, 47, 154]).

In this chapter we often utilize a *graphical approach* to nonlinear analysis where, for example, we plot the input-output map of the fuzzy controller and read off pertinent information such as the sector bounds, or use a graphical technique for describing function analysis. We believe that the incorporation of graphical techniques for the nonlinear analysis of fuzzy control systems offers (1) an intuitive approach that ties in better with the fuzzy control design procedure, and (2) some of the same advantages as have been realized in classical control via the use of graphical techniques (such as the Nyquist plot). On the other hand, our approach has its own limitations (listed above). We emphasize that there are many approaches to analyzing fuzzy control systems, and we highly recommend that the reader study Section 4.9, For Further Study, and the references provided there.

4.8 Summary

In this chapter we have provided an introduction to nonlinear analysis of (non-adaptive) fuzzy control systems. We showed how to perform Lyapunov stability analysis of fuzzy control systems and showed how the circle criterion could be used to analyze and redesign a fuzzy control system. We introduced the theory of

steady-state tracking error for fuzzy control systems and showed how to predict and eliminate tracking error. We outlined the theory of describing functions and showed how to predict the amplitude, frequency, and stability of limit cycles. We performed analysis and design examples for an inverted pendulum, a temperature control problem, a hydrofoil, an underwater vehicle, and a tape drive servo.

Upon completing this chapter, the reader should understand the following:

- Lyapunov's direct and indirect methods.
- How to use the direct and indirect methods, coupled with a plot of the nonlinear surface of the fuzzy controller, to establish conditions for stability.
- How to use Lyapunov's direct method to provide sufficient conditions for stability for Takagi-Sugeno fuzzy systems.
- The concept of absolute stability.
- The circle criterion in two forms.
- The procedure for the application of the circle criterion to fuzzy control systems, both to predict instability and its use in design to avoid instability.
- The concepts and theory of steady-state tracking error for nonlinear systems.
- The procedure for applying the theory of analysis of tracking error to fuzzy control systems.
- The assumptions and theory of describing functions.
- How to construct a describing function for a fuzzy controller that has one or two inputs.
- The conditions for the existence of limit cycles and how to determine their amplitude and frequency, and whether or not they are stable.
- The procedure to use describing function analysis for both SISO and MISO fuzzy control systems, both for limit cycle prediction and in redesigning for limit cycle elimination.

Essentially, this is a checklist of the major topics of this chapter. With the completion of Chapters 1–4 you have now finished the first part of this book, where our primary focus has been on direct fuzzy controllers. The second part of the book, Chapters 5–7, focuses on adaptive fuzzy systems in estimation and control.

4.9 For Further Study

An earlier version of this chapter appears in [83]. Several of the design problems at the end of the chapter also came from [83]. For a detailed comparative analysis of

fuzzy controllers and linear controllers and for more details on the nonlinear characteristics of fuzzy controllers, see [29, 28, 241] and the more recent work in [124]. The work in [30] and [34] presents Lyapunov methods for analyzing the stability of fuzzy control systems. The authors in [106, 105] also use Lyapunov's direct method and the generalized theorem of Popov [148, 90] to provide sufficient conditions for fuzzy control system stability.

An area that is receiving an increasing amount of attention is stability analysis of fuzzy control systems where the fuzzy control system is developed using ideas from sliding-mode control or where Takagi-Sugeno fuzzy systems are used in a gain-scheduling type of control [153, 47, 154]. Here, our treatment of the stability of Takagi-Sugeno fuzzy systems is based on the work in [213, 209]. Extensions to this work that focus on robustness can be found in [212, 210, 84], and work focusing on the use of linear matrix inequality (LMI) methods for analysis and controller construction is provided in [210, 226, 225, 248, 247].

In [7], stability indices for fuzzy control systems are established using phase portraits (of course, standard phase plane analysis [90] can be useful in characterizing and understanding the dynamic behavior of low-order fuzzy control systems [65]). Related work is given in [49]. The circle criterion [148] is used in [171] and [172] to provide sufficient conditions for fuzzy control system stability. Related work on stability analysis of fuzzy control systems is provided in [211]. While we use the circle criterion theory found in [90] and [223], there are other frequency domain–based criteria for stability that can be utilized for fuzzy control system analysis (e.g., Popov's criterion and the multivariable circle criterion [148, 90]). Describing function analysis has already been examined in [92] and [14]. Our coverage here differs from that in [92] in that we use experimentally determined describing functions, whereas in [92] the describing function is determined for a "multilevel relay" model of a specific class of fuzzy controllers. A collection of papers on theoretical aspects of fuzzy control is in [151].

The characterization and analysis of the stability of fuzzy dynamic systems is studied in [93]. Furthermore, approximate analysis of fuzzy systems is studied by the authors in [33, 32, 52] using the "cell-to-cell mapping approach" from [71, 72].

One graphical technique that we have found to be useful on occasion, which we did not cover here, is called the "method of equivalent gains" (see [55, 54]), where we view the fuzzy controller as an input-dependent time-varying gain and then use conventional root-locus methods to design fuzzy control systems (the gain moves the poles along the root-locus). This method is, however, limited to the case of linear plants. For an idea of how this approach is used, see Exercise 4.3 at the end of the chapter. Another topic that we did not cover is that of phase plane analysis for differential equations and what has been called "fuzzy phase plane analysis." To get an idea of how such analysis is done, see Exercise 4.2 at the end of this chapter or [47].

For a more detailed discussion on the general relationships between conventional and intelligent control and mathematical modeling and nonlinear analysis of more general intelligent control systems (including expert control systems), see [6, 160, 156, 157, 163].

4.10 Exercises

Exercise 4.1 (The Nonlinear Fuzzy Control Surface): In this problem you will study the nonlinear control surface that is induced by the fuzzy controller.

(a) Plot u versus e for the parameterized SISO fuzzy controller of Section 4.2.1 for the case where $A = 5$ and $B = 2$.

(b) Plot u versus e for the parameterized SISO fuzzy controller of Section 4.2.1 for the case where $A = 3$ and $B = 6$. Compare the result to that obtained in (a).

(c) Plot the three-dimensional plot of the PD fuzzy controller surface for the case where there is a proportional and derivative input, as described in Section 4.2. Choose $A = B = D = 1$.

(d) Choose $A = 5$, $B = 2$, and $D = 1$ and repeat (c). Compare the result to that obtained in (c).

Exercise 4.2 (Phase Plane Analysis: Conventional and Fuzzy): The phase plane is a graph used for the analysis of low-order (typically second-order) nonlinear differential equations (i.e., $n = 2$ for Equation (4.1)). The phase plane is simply a plot of $x_1(t)$ versus $x_2(t)$, where $x = [x_1, x_2]^\top$ is the state of Equation (4.1), for a number of initial conditions $x(0)$.

(a) Write down second-order differential equations that are unstable, marginally stable, and asymptotically stable, and use a computer program to generate their phase planes (the choice of the initial conditions should be done so that they are within a ball of size h where $h = 10$ and there are at least 50 initial conditions spread out uniformly in the ball).

(b) Learn at least one technique for the construction of phase planes (by hand) and apply it to the differential equations you developed for (a). Refer to [90] to learn a method for constructing the phase plane.

(c) When the inputs to a fuzzy controller are $e(t) = r(t) - y(t)$ (where $r(t)$ is the reference input and $y(t)$ is the plant output) and $\frac{d}{dt}e(t)$, sometimes the plot of $e(t)$ versus $\frac{d}{dt}e(t)$ is thought of as a type of phase plane if $r(t) = 0$. Moreover, some have introduced the notion of a "fuzzy phase plane" that is best thought of as a rule-base table for the two-input fuzzy controller. Motion in the fuzzy phase plane is given by which membership functions have values greater than zero and as the control system operates, different cells in the rule-base table become "active" (i.e., the rules associated with them are on). Following Design Problem 2.1(a) on page 110, begin the pendulum out of the balanced position but with zero initial velocity and show on the corresponding rule-base table the trajectory of active regions as the fuzzy controller balances the pendulum.

Exercise 4.3 (Method of Equivalent Gains): In the "method of equivalent gains" (see [55, 54]), we view the fuzzy controller as an input-dependent time-varying gain and use conventional root-locus methods to design fuzzy control systems (the gain moves the poles along the root-locus).

(a) To understand why the fuzzy controller is an input-dependent gain, choose $A = B = 1$ for the parameterized SISO fuzzy controller of Section 4.2.1, and plot the output of the fuzzy controller u divided by its input e (i.e., the "gain of the fuzzy controller"—notice that it is closely related to the describing function of the fuzzy controller) versus its input e for both positive and negative values of e.

(b) Suppose that you are given a plant

$$G(s) = \frac{1}{s(s+1)}$$

that is in a unity feedback configuration with a fuzzy controller. Suppose that you know that the reference input will never be larger, in magnitude, than one. View the fuzzy controller as implementing a gain in the control loop where the value of the gain is given at any one time by the plot you produced in (a). Use this gain, coupled with the conventional root-locus approach [54], to design a fuzzy controller so that you get as short a rise-time due to a unit-step input as possible but with no more than 5% overshoot. This approach to design is called the method of equivalent gains. Note that this approach is heuristic and that there are no guarantees of achieving the performance sought or that the resulting closed-loop system will be stable.

Exercise 4.4 (Lyapunov's Direct Method): Suppose that you are given the plant

$$\dot{x} = ax + bu$$

where $b > 0$ and $a < 0$ (so the system is stable). Suppose that you design a fuzzy controller Φ that generates the input to the plant given the state of the plant (i.e., $u = \Phi(x)$). Assume that you design the fuzzy controller so that $\Phi(0) = 0$ and so that $\Phi(x)$ is continuous. Choose the Lyapunov function $V(x) = \frac{1}{2}x^2$.

(a) Show that if x and $\Phi(x)$ always have opposite signs, then $x = 0$ is stable.

(b) What types of stability does $x = 0$ of the fuzzy control system possess for part (a)?

(c) Why do we assume that $\Phi(0) = 0$ for (a)?

(d) Design a fuzzy controller that satisfies the condition stated in (a) and simulate the closed-loop system to help illustrate the stability of the fuzzy control system (of course, the simulation does not *prove* that the closed-loop system

is stable—it only shows that for one initial condition the state appears to converge but cannot prove that it converges since the simulation is only for a finite amount of time). Choose the initial condition $x(0) = 1$, $a = -2$, and $b = 2$.

Exercise 4.5 (Stability of Takagi-Sugeno Fuzzy Systems): Suppose that you have the same plant as described in the Section 4.3.5 example but with $A_1 = -3$, $B_1 = 6$, $A_2 = -5$, and $B_2 = 2$. Construct the Takagi-Sugeno fuzzy controller gains K_1 and K_2 so that $x(0) = 0$ of the closed-loop system is globally asymptotically stable.

Exercise 4.6 (Stability of Discrete-Time Takagi-Sugeno Fuzzy Systems): Suppose that you are given a discrete-time Takagi-Sugeno fuzzy system model of a nonlinear system that arises from R Takagi-Sugeno rules and results in

$$x(k+1) = \sum_{i=1}^{R} \Phi_i \xi_i(x(k))x(k) + \sum_{i=1}^{R} \Gamma_i \xi_i(x(k))u(k) \tag{4.42}$$

where

$$\xi_i(x(k)) = \frac{\mu_i(x(k))}{\sum_{i=1}^{R} \mu_i(x(k))}$$

In Equation (4.42), Φ_i is an $n \times n$ matrix, and Γ_i is the n input matrix.

Stability conditions for the discrete-time direct method of Lyapunov are slightly different from the continuous-time case so we discuss these first. The equilibrium $x(0) = 0$ of the system in Equation (4.42) is globally asymptotically stable if there exists a function $V(x)$ such that $V(x) \geq 0$ except at $x = 0$ where $V(x) = 0$, $V(x) \to \infty$ if $|x| \to \infty$, and

$$V(x(k+1)) - V(x(k)) < 0$$

(a) Let $u(k) = 0$ for $k \geq 0$. Choose $V(x) = x^\top P x$ where P is a positive definite symmetric matrix. Show that if there exists a single $n \times n$ matrix, $P > 0$ such that for all $i = 1, 2, \ldots, R$ and $j = 1, 2, \ldots, R$

$$\Phi_i^\top P \Phi_j - P < 0 \tag{4.43}$$

then the equilibrium $x = 0$ of Equation (4.42) is globally asymptotically stable.

(b) Suppose that you use a Takagi-Sugeno fuzzy controller to choose the input $u(k)$ so that

$$u(k) = \sum_{i=1}^{R} K_i \xi_i(x(k))x(k)$$

Using the result from (a), find a stability condition similar to Equation (4.43) for the closed-loop system.

This problem is based on the work in [213] where the authors also show how to further simplify the condition in Equation (4.43).

4.11 Design Problems

Design Problem 4.1 (Stable Fuzzy Controller for an Inverted Pendulum): In this problem you will verify the stability analysis for the design of the fuzzy controller for the inverted pendulum described in Section 4.3.

(a) Design a fuzzy controller that will result in the inverted pendulum of Section 4.3 being locally stable, and demonstrate this via Lyapunov's indirect method.

(b) Repeat (a) except use minimum to represent the premise and implication and COG for defuzzification.

(c) Using Lyapunov's direct method, design a fuzzy controller for the inverted pendulum that you can guarantee is stable in the inverted position. Provide a simulation to help verify the stability of the closed-loop system.

Design Problem 4.2 (Stable Fuzzy Controller for the Magnetic Ball Suspension System): In this problem you study the stability properties of a fuzzy controller for the magnetic ball suspension system.

(a) Design a fuzzy controller for the ball suspension system studied in Exercise 2.5 on page 116, and demonstrate in simulation that it *appears* to be stable (at least locally—i.e., for initial conditions very near the operating point at which you perform the linearization to test stability). Seek to balance the ball half way between the coil and the "ground."

(b) Prove, using the methods of Section 4.3, that the fuzzy control system is locally stable at the operating point studied in (a).

Design Problem 4.3 (Designing Stable Fuzzy Control Systems): Suppose that you are given a plant with transfer function

$$G(s) = \frac{1}{s^3 + 7s^2 + 7s + 15}$$

This plant is chosen because it illustrates the problems with stability that can arise when designing fuzzy controllers.

(a) A controller that some expert could construct is one with $A = 0.5$ and $B = 16.6667$ (using the parameterized fuzzy controller of Section 4.2.1). Simulate this system with initial conditions $x(0) = [0, 0, 2]^\top$ to show that the system has sustained oscillations.

(b) If we consider the fuzzy controller as a nonlinearity Φ, we can find a sector (α, β) in which Φ lies and use the circle criterion to determine why the instability is occurring and perhaps determine how to tune the fuzzy controller so that it does not cause sustained oscillations. Plot the nonlinearity of the fuzzy controller from (a). Plot the Nyquist plot of G. Show that the circle criterion/SNC predicts that not all of the nonlinearities within this sector will be stable. Hence, the fuzzy controller in (a) verifies this statement by producing sustained oscillations in the closed-loop system.

(c) Next we use condition (b) of the circle criterion/SNC to provide ideas on how to tune the fuzzy controller. To do this, we will have to adjust β so that $-\frac{1}{\beta} < -0.0733$, (i.e., so that $\beta < 13.64$). Why? As there are many different choices for A and B so that the fuzzy controller will fit inside the sector, more about the system would have to be known (e.g., what the saturation limits at the input of the plant are) to know whether to tune A or B. Suppose you choose $B = 16.6667$ and make $A > 1.222$ so that $\frac{B}{A} < 13.64$. As an example, choose $A = 1.3$. Produce a simulation of the resulting fuzzy control system with $x(0) = [0, 0, 2]^\top$ to show that there are no sustained oscillations so that the fuzzy controller has been successfully redesigned to avoid the instability.

(d) Repeat (c) but choose $A = 0.5$ and find a value of B that will result in a stable closed-loop system. Justify your choice of B theoretically and by providing a simulation that shows the choice was good.

Design Problem 4.4 (Stable Temperature Control): In this problem you will verify the results of Section 4.4.2 on page 208 where the problem of designing a stable fuzzy control system for a temperature control problem was addressed. Suppose that the control system that we use is shown in Figure 4.9. The controller $G_c(s)$ is a post compensator for the fuzzy controller.

(a) Suppose that we begin by choosing $G_c(s) = K = 2$. Provide a plot of q' versus e for the 11-rule fuzzy controller that is specified in Section 4.4.2.

(b) Show that $\alpha = 0$ and $\beta = \frac{4}{3}$. Plot the Nyquist plot of G_cG and determine the number of encirclements. What conclusion can be reached from the circle criterion?

(c) Choose a value for the initial state of the system, let the reference input be zero, and show that the state trajectories converge asymptotically to zero.

(d) Let $\tau_d = 20u(t)$ where $u(t)$ is a unit step, and determine the value of the steady-state error using a simulation of the closed-loop system.

(e) Suppose that we choose $G_c(s) = \frac{3}{s}$ (chosen to try to eliminate the steady-state error). Using the plot of the fuzzy controller nonlinearity, show that

we can choose $\alpha = \frac{3}{4}$ and $\beta = \frac{4}{3}$ and the sector condition holds on a region $[-80, 80]$.

(f) Show that the Nyquist plot of $G_c G$ does not encircle the disk $D(-4/3, -3/4)$. What is concluded from the circle criterion?

(g) Do a simulation where the initial conditions are started sufficiently close to the origin (and the reference input is equal to zero) to show that the state trajectories asymptotically decrease to zero.

(h) Next, choose $\tau_d = 20u(t)$ where $u(t)$ is a unit step, and do a simulation to show that there is no steady-state error.

Design Problem 4.5 (Designing for Zero Steady State Tracking Error): Consider a plant of the form

$$G(s) = \frac{1}{s^2 + 4s + 3}$$

(a) Choose a SISO proportional fuzzy controller and determine the α and β describing the sector in which it lies where the type of sector is the one used for the theory of steady-state tracking error. Note that you can find α and β numerically by inserting values of a and b into the equation

$$\alpha \leq \frac{\Phi(a) - \Phi(b)}{a - b} \leq \beta$$

and determining the maximum and minimum values. An alternative approach is to plot the fuzzy controller nonlinearity and read the values off the plot by inspection.

(b) Which condition of the circle criterion holds? Show a Nyquist plot to support your conclusion.

(c) For your choice of α and β, find c_0 and $H(0)$. Find γ, and then solve the recursive equation from Equation (4.25). Suppose that we choose a step input of magnitude 3. What is the value of e_{ss}?

(d) Suppose that we consider the steady-state error to be excessive, and that we would like to redesign our fuzzy controller using the steady-state error prediction procedure as part of the design process. Intuitively, we would expect that if we increased the "gain of the fuzzy controller," the steady-state error would decrease. In terms of the e_{ss} prediction procedure, this would mean changing α and β. Because of the inherent saturation of the fuzzy controller, α will always equal 0. Therefore, we will have to adjust by changing β only. Find a value of β so that $e_{ss} < 0.4$.

(e) Consider the response of the system from (d) to a ramp input. What is the value of $e(t)$ as t goes to infinity? Will changing the scaling gains of your fuzzy controller improve tracking error?

Design Problem 4.6 (Design of Hydrofoil Controller to Get Zero Tracking Error): In this problem you will verify the results of Section 4.5.2 on page 213. Suppose that we use a proportional fuzzy controller of the form described in Section 4.2.1.

(a) Show that β must be less than 1.56 for the circle criterion conditions to be satisfied.

(b) Choose $A = B = 1$. Show that $c_0 = 0.5$, $H(0) = 0.6667$, and $G(0) = 1$. Let the input be a step with magnitude 5.0, and show that $\gamma = 3.3333$. Find the value of the steady-state error.

(c) Add an integrator and show that if $B/A < 50$ we can meet the conditions to get $e_{ss} = 0$. Perform a simulation for this system with $A = B = 1$ and show that $e_{ss} = 0$.

Design Problem 4.7 (Prediction and Elimination of Limit Cycles: SISO Case): Suppose that a fuzzy controller of the form described in Section 4.2.1 has $A = 0.2$ and $B = 0.1$ and a plant with transfer function

$$G(s) = \frac{1}{s(s^2 + 0.2s + 1)}$$

configured in the form used in Section 4.6.1.

(a) Plot the describing function for the fuzzy controller.

(b) Plot $G(j\omega)$ and $-\frac{1}{N(C,\omega)}$ on the same plot and find the intersection point(s). What are the magnitude and frequency of the predicted limit cycle? Is the limit cycle stable? Why?

(c) The last step of this process is to verify by simulation that the limit cycle does exist. Choose $r(t) = 1$ and simulate the closed-loop system. What are the frequency and amplitude of the limit cycle in the simulation? Compare your results to the predicted values in (b).

(d) Now that we have predicted the existence of a limit cycle for our system, we desire to redesign the fuzzy controller so that there are no limit cycles. What value must $-1/N(C, \omega)$ be less than so that there would be no intersection point and no limit cycle? What values of A and B should you choose so that there will be no limit cycles? Why? Choose $r(t) = 1$ and simulate the closed-loop system to verify that there are now no limit cycles for your choice of A and B.

Design Problem 4.8 (Prediction and Elimination of Limit Cycles: SISO Case, Unstable Limit Cycle): Consider a plant with transfer function

$$G(s) = \frac{s^2 + 0.4s + 2.29}{s(s^2 + 0.4s + 1.04)}$$

(a) Our first design for the fuzzy controller will have $A = 0.1$ and $B = 0.3$. To predict the limit cycles of this system, find $N(C, \omega)$ then plot $-1/N(C, \omega)$ and $G(j\omega)$ on the same plot and identify the intersection points. What amplitudes and frequencies will the limit cycles have? Are they stable?

(b) To confirm that these limit cycles exist, simulate the system with $r = 0.761$ (this value was chosen to best show the existence of both limit cycles). What are the values of the amplitudes and frequencies of the limit cycles? How do these compare with the predicted values? What happens if $r < 0.761$? Simulate the system for this case to illustrate the behavior.

(c) Redesign the fuzzy controller so that no limit cycles exist. To demonstrate that no limit cycles exist for your design, use the theory and a simulation with $r = 0.761$.

Design Problem 4.9 (Prediction and Elimination of Limit Cycles: MISO Case): Suppose that the plant has the transfer function

$$G(s) = \frac{(s+1)^2}{s^3}$$

Our fuzzy controller is the two-input fuzzy controller with inputs e and $\dot{e}$ described in Section 4.2 on page 189, and with parameters A, B, and D.

(a) Show that choosing $A = B = D = 1$ is not a good choice.

(b) Use describing function analysis to choose the parameters A, B, and D for the fuzzy controller so that no limit cycles occur, and demonstrate in simulation that they do not occur. Note that when you experimentally determine the describing function you must consider a range of values of both C and ω to find different $-1/N(C, \omega)$ curves to find the intersection points. You can assume that the reference input is a positive step with a magnitude no larger than five. What happens if the amplitude of the step input is greater than 30? Simulate the system for this case to illustrate the behavior.

CHAPTER 5

Fuzzy Identification and Estimation

For the things we have to learn before we can do them,
we learn by doing them.
–Aristotle

5.1 Overview

While up to this point we have focused on control, in this chapter we will examine how to use fuzzy systems for estimation and identification. The basic problem to be studied here is how to construct a fuzzy system from numerical data. This is in contrast to our discussion in Chapters 2 and 3, where we used linguistics as the starting point to specify a fuzzy system. If the numerical data is plant input-output data obtained from an experiment, we may identify a fuzzy system model of the plant. This may be useful for simulation purposes and sometimes for use in a controller. On the other hand, the data may come from other sources, and a fuzzy system may be used to provide for a parameterized nonlinear function that fits the data by using its basic interpolation capabilities. For instance, suppose that we have a human expert who controls some process and we observe how she or he does this by observing what *numerical* plant input the expert picks for the given *numerical* data that she or he observes. Suppose further that we have many such associations between "decision-making data." The methods in this chapter will show how to construct rules for a fuzzy controller from this data (i.e., identify a controller from the human-generated decision-making data), and in this sense they provide another method to design controllers.

Yet another problem that can be solved with the methods in this chapter is that of how to construct a fuzzy system that will serve as a parameter estimator.

To do this, we need data that shows roughly how the input-output mapping of the estimator should behave (i.e., how it should estimate). One way to generate this data is to begin by establishing a simulation test bed for the plant for which parameter estimation must be performed. Then a set of simulations can be conducted, each with a different value for the parameter to be estimated. By coupling the test conditions and simulation-generated data with the parameter values, you can gather appropriate data pairs that allow for the construction of a fuzzy estimator. For some plants it may be possible to perform this procedure with actual experimental data (by physically adjusting the parameter to be estimated). In a similar way, you could construct fuzzy predictors using the approaches developed in this chapter.

We begin this chapter by setting up the basic function approximation problem in Section 5.2, where we provide an overview of some of the fundamental issues in how to fit a function to input-output data, including how to incorporate linguistic information into the function that we are trying to force to match the data. We explain how to measure how well a function fits data and provide an example of how to choose a data set for an engine failure estimation problem (a type of parameter estimation problem in which when estimates of the parameters take on certain values, we say that a failure has occurred).

In Section 5.3 we introduce conventional least squares methods for identification, explain how they can be used to tune fuzzy systems, provide a simple example, and offer examples of how they can be used to train fuzzy systems. Next, in Section 5.4 we show how gradient methods can be used to train a standard and Takagi-Sugeno fuzzy system. These methods are quite similar to the ones used to train neural networks (e.g., the "back-propagation technique"). We provide examples for standard and Takagi-Sugeno fuzzy systems. We highlight the fact that via either the recursive least squares method for fuzzy systems or the gradient method we can perform on-line parameter estimation. We will see in Chapter 6 that these methods can be combined with a controller construction procedure to provide a method for adaptive fuzzy control.

In Section 5.5 we introduce two techniques for training fuzzy systems based on clustering. The first uses "c-means clustering" and least squares to train the premises and consequents, respectively, of the Takagi-Sugeno fuzzy system; while the second uses a nearest neighborhood technique to train standard fuzzy systems. In Section 5.6 we present two "learning from examples" (LFE) methods for constructing rules for fuzzy systems from input-output data. Compared to the previous methods, these do not use optimization to construct the fuzzy system parameters. Instead, the LFE methods are based on simple procedures to extract rules directly from the data.

In Section 5.7 we show how hybrid methods for training fuzzy systems can be developed by combining the methods described in this chapter. Finally, in Section 5.8, we provide a design and implementation case study for parameter estimation in an internal combustion engine.

Overall, the objective of this chapter is to show how to construct fuzzy systems from numerical data. This will provide the reader with another general approach for fuzzy system design that may augment or extend the approach described in

Chapters 2 and 3, where we start from linguistic information. With a good understanding of Chapter 2, the reader can complete this chapter without having read Chapters 3 and 4. The section on indirect adaptive control in Chapter 6 relies on the gradient and least squares methods discussed in this chapter, and a portion of the section on gain schedule construction in Chapter 7 relies on the reader knowing at least one method from this chapter. In other words, this chapter is important since many adaptive control techniques depend on the use of an estimator. Moreover, the sections on neural networks and genetic algorithms in Chapter 8 depend on this chapter in the sense that if you understand this chapter and those sections, you will see how those techniques relate to the ones discussed here. Otherwise, the remainder of the book can be completed without this chapter; however, this chapter will provide for a deeper understanding of many of the concepts to be presented in Chapters 6 and 7. For example, the learning mechanism for the fuzzy model reference learning controller (FMRLC) described in Chapter 6 can be viewed as an identification algorithm that is used to tune a fuzzy controller.

5.2 Fitting Functions to Data

We begin this section by precisely defining the function approximation problem, in which you seek to synthesize a function to approximate another function that is inherently represented via a finite number of input-output associations (i.e., we only know how the function maps a finite number of points in its domain to its range). Next, we show how the problem of how to construct nonlinear system identifiers and nonlinear estimators is a special case of the problem of how to perform function approximation. Finally, we discuss issues in the choice of the data that we use to construct the approximators, discuss the incorporation of linguistic information, and provide an example of how to construct a data set for a parameter estimation problem.

5.2.1 The Function Approximation Problem

Given some function

$$g : \bar{\mathcal{X}} \rightarrow \bar{\mathcal{Y}}$$

where $\bar{\mathcal{X}} \subset \Re^n$ and $\bar{\mathcal{Y}} \subset \Re$, we wish to construct a fuzzy system

$$f : X \rightarrow Y$$

where $X \subset \bar{\mathcal{X}}$ and $Y \subset \bar{\mathcal{Y}}$ are some domain and range of interest, by choosing a parameter vector θ (which may include membership function centers, widths, etc.) so that

$$g(x) = f(x|\theta) + e(x) \tag{5.1}$$

for all $x = [x_1, x_2, \ldots, x_n]^\top \in X$ where the approximation error $e(x)$ is as small as possible. If we want to refer to the input at time k, we will use $x(k)$ for the vector and $x_j(k)$ for its j^{th} component.

Assume that all that is available to choose the parameters θ of the fuzzy system $f(x|\theta)$ is some part of the function g in the form of a finite set of input-output data pairs (i.e., the functional mapping implemented by g is largely unknown). The i^{th} input-output data pair from the system g is denoted by (x^i, y^i) where $x^i \in X$, $y^i \in Y$, and $y^i = g(x^i)$. We let $x^i = [x_1^i, x_2^i, ..., x_n^i]^\top$ represent the input vector for the i^{th} data pair. Hence, x_j^i is the j^{th} element of the i^{th} data vector (it has a specific value and is not a variable). We call the set of input-output data pairs the *training data set* and denote it by

$$G = \{(x^1, y^1), \ldots, (x^M, y^M)\} \subset X \times Y \tag{5.2}$$

where M denotes the number of input-output data pairs contained in G. For convenience, we will sometimes use the notation $d^{(i)}$ for data pair (x^i, y^i).

To get a graphical picture of the function approximation problem, see Figure 5.1. This clearly shows the challenge; it can certainly be hard to come up with a good function f to match the mapping g when we know only a little bit about the association between X and Y in the form of data pairs G. Moreover, it may be hard to know when we have a good approximation—that is, when f approximates g over the *whole* space of inputs X.

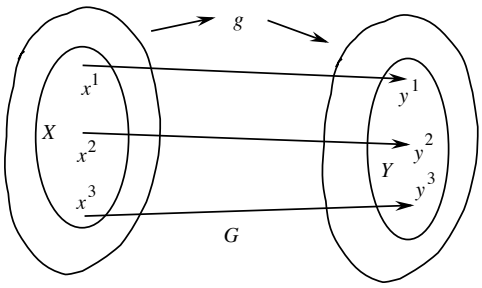


FIGURE 5.1 Function mapping with three known input-output data pairs.

To make the function approximation problem even more concrete, consider a simple example. Suppose that $n = 2$, $X \subset \Re^2$, $Y = [0, 10]$, and $g : X \to Y$. Let $M = 3$ and the training data set

$$G = \left\{ \left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, 1 \right), \left(\begin{bmatrix} 2 \\ 4 \end{bmatrix}, 5 \right), \left(\begin{bmatrix} 3 \\ 6 \end{bmatrix}, 6 \right) \right\} \tag{5.3}$$

which partially specifies g as shown in Figure 5.2. The function approximation problem amounts to finding a function $f(x|\theta)$ by manipulating θ so that $f(x|\theta)$

approximates g as closely as possible. We will use this simple data set to illustrate several of the methods we develop in this chapter.

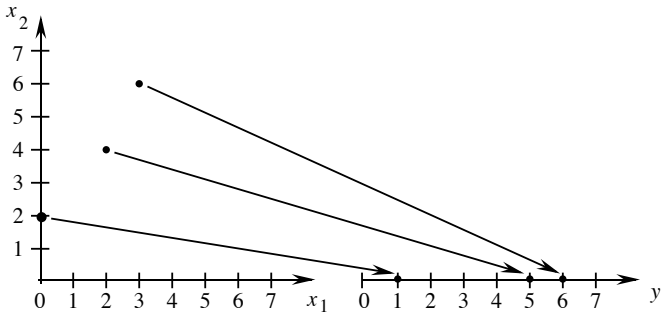


FIGURE 5.2 The training data G generated from the function g.

How do we evaluate how closely a fuzzy system $f(x|\theta)$ approximates the function $g(x)$ for all $x \in X$ for a given θ? Notice that

$$\sup_{x \in X} \{|g(x) - f(x|\theta)|\} \tag{5.4}$$

is a bound on the approximation error (if it exists). However, specification of such a bound requires that the function g be completely known; however, as stated above, we know only a part of g given by the finite set G. Therefore, we are only able to evaluate the accuracy of approximation by evaluating the error between $g(x)$ and $f(x|\theta)$ at certain points $x \in X$ given by *available* input-output data. We call this set of input-output data the *test set* and denote it as Γ, where

$$\Gamma = \{(x^1, y^1), \ldots, (x^{M_\Gamma}, y^{M_\Gamma})\} \subset X \times Y \tag{5.5}$$

Here, M_Γ denotes the number of known input-output data pairs contained within the test set. It is important to note that the input-output data pairs (x^i, y^i) contained in Γ may not be contained in G, or vice versa. It also might be the case that the test set is equal to the training set ($G = \Gamma$); however, this choice is not always a good one. Most often you will want to test the system with at least some data that were not used to construct $f(x|\theta)$ since this will often provide a more realistic assessment of the quality of the approximation.

We see that evaluation of the error in approximation between g and a fuzzy system $f(x|\theta)$ based on a test set Γ may or may not be a true measure of the error between g and f for every $x \in X$, but it is the only evaluation we can make based

on known information. Hence, you can use measures like

$$\sum_{(x^i,y^i)\in\Gamma} (g(x^i) - f(x^i\theta))^2 \tag{5.6}$$

or

$$\sup_{(x^i,y^i)\in\Gamma} \{|g(x^i) - f(x^i|\theta)|\} \tag{5.7}$$

to measure the approximation error. Accurate function approximation requires that some expression of this nature be small; however, this clearly does not guarantee perfect representation of g with f since most often we cannot test that f matches g over all possible input points.

We would like to emphasize that the type of function that you choose to adjust (i.e., $f(x|\theta)$) can have a significant impact on the ultimate accuracy of the approximator. For instance, it may be that a Takagi-Sugeno (or functional) fuzzy system will provide a better approximator than a standard fuzzy system for a particular application. We think of $f(x|\theta)$ as a structure for an approximator that is parameterized by θ. In this chapter we will study the use of fuzzy systems as approximators, and use a fuzzy system as the structure for the approximator. The choice of the parameter vector θ depends on, for example, how many membership functions and rules you use. Generally, you want enough membership functions and rules to be able to get good accuracy, but not too many since if your function is "overparameterized" this can actually degrade approximation accuracy. Often, it is best if the structure of the approximator is based on some physical knowledge of the system, as we explain how to do in Section 5.2.4 on page 241.

Finally, while in this book we focus primarily on fuzzy systems (or, if you understand neural networks you will see that several of the methods of this chapter directly apply to those also), at times it may be beneficial to use other approximation structures such as neural networks, polynomials, wavelets, or splines (see Section 5.10 "For Further Study," on page 302).

5.2.2 Relation to Identification, Estimation, and Prediction

Many applications exist in the control and signal processing areas that may utilize nonlinear function approximation. One such application is system identification, which is the process of constructing a mathematical model of a dynamic system using experimental data from that system. Let g denote the physical system that we wish to identify. The training set G is defined by the experimental input-output data.

In linear system identification, a model is often used where

$$y(k) = \sum_{i=1}^{\bar{q}} \theta_{a_i} y(k-i) + \sum_{i=0}^{\bar{p}} \theta_{b_i} u(k-i) \tag{5.8}$$

and $u(k)$ and $y(k)$ are the system input and output at time $k \geq 0$. Notice that you will need to specify appropriate initial conditions. In this case $f(x|\theta)$, which is not a fuzzy system, is defined by

$$f(x|\theta) = \theta^\top x$$

where

$$x(k) = [y(k-1), \cdots, y(k-\bar{q}), u(k), \cdots, u(k-\bar{p})]^\top \tag{5.9}$$
$$\theta = [\theta_{a_1}, \cdots, \theta_{a_{\bar{q}}}, \theta_{b_0}, \cdots, \theta_{b_{\bar{p}}}]^\top \tag{5.10}$$

Let $N = \bar{q} + \bar{p} + 1$ so that $x(k)$ and θ are $N \times 1$ vectors. Linear system identification amounts to adjusting θ using information from G so that $g(x) = f(x|\theta) + e(x)$ where $e(x)$ is small for all $x \in X$.

Similar to conventional linear system identification, for fuzzy identification we will utilize an appropriately defined "regression vector" x as specified in Equation (5.9), and we will tune a fuzzy system $f(x|\theta)$ so that $e(x)$ is small. Our hope is that since the fuzzy system $f(x|\theta)$ has more functional capabilities (as characterized by the universal approximation property described in Section 2.3.8 on page 77) than the linear map defined in Equation (5.8), we will be able to achieve more accurate identification for nonlinear systems by appropriate adjustment of its parameters θ of the fuzzy system.

Next, consider how to view the construction of a parameter (or state) estimator as a function approximation problem. To do this, suppose for the sake of illustration that we seek to construct an estimator for a single parameter in a system g. Suppose further that we conduct a set of experiments with the system g in which we vary a parameter in the system—say, α. For instance, suppose we know that the parameter α lies in the range $[\alpha_{min}, \alpha_{max}]$ but we do not know where it lies and hence we would like to estimate it. Generate a data set G with data pairs $(x^i, \alpha^i) \in G$ where the α^i are a range of values over the interval $[\alpha_{min}, \alpha_{max}]$ and the x^i corresponding to each α^i is a set of input-output data from the system g in the form of Equation (5.9) that results from using α^i as the parameter value in g. Let $\hat{\alpha}$ denote the fuzzy system estimate of α. Now, if we construct a function $\hat{\alpha} = f(x|\theta)$ from the data in G, it will serve as an estimator for the parameter α. Each time a new x vector is encountered, the estimator f will interpolate between the known associations $(x^i, \alpha^i) \in G$ to produce the estimate $\hat{\alpha}$. Clearly, if the data set G is "rich" enough, it will have enough (x^i, α^i) pairs so that when the estimator is presented with an $x \neq x^i$, it will have a good idea of what $\hat{\alpha}$ to specify because it will have many x^i that are close to x that it *does* know how to specify α for. We will study several applications of parameter estimation in this chapter and in the problems at the end of the chapter.

To apply function approximation to the problem of how to construct a predictor for a parameter (or state variable) in a system, we can proceed in a similar manner to how we did for the parameter estimation case above. The only significant difference lies in how to specify the data set G. In the case of prediction, suppose that we

wish to estimate a parameter $\alpha(k+D)$, D time steps into the future. In this case we will need to have available training data pairs $(x^i, \alpha^i(k+D)) \in G$ that associate known future values of α with available data x^i. A fuzzy system constructed from such data will provide a predicted value $\hat{\alpha}(k+D) = f(x|\theta)$ for given values of x.

Overall, notice that in each case—identification, estimation, and prediction—we rely on the existence of the data set G from which to construct the fuzzy system. Next, we discuss issues in how to choose the data set G.

5.2.3 Choosing the Data Set

While the method for adjusting the parameters θ of $f(x|\theta)$ is critical to the overall success of the approximation method, there is virtually no way that you can succeed at having f approximate g if there is not appropriate information present in the training data set G. Basically, we would like G to contain as much information as possible about g. Unfortunately, most often the number of training data pairs is relatively small, or it is difficult to use too much data since this affects the computational complexity of the algorithms that are used to adjust θ. The key question is then, How would we like the limited amount of data in G structured so that we can adjust θ so that f matches g very closely?

There are several issues involved in answering this question. Intuitively, if we can manage to spread the data over the input space uniformly (i.e., so that there is a regular spacing between points and not too many more points in one region than another) and so that we get coverage of the whole input space, we would often expect that we may be able to adjust θ properly, provided that the space between the points is not too large [108]. This is because we would then expect to have information about how the mapping g is shaped in all regions so we should be able to approximate it well in all regions. The accuracy will generally depend on the slope of g in various regions. In regions where the slope is high, we may need more data points to get more information so that we can do good approximation. In regions with lower slopes, we may not need as many points. This intuition, though, may not hold for all methods of adjusting θ. For some methods, you may need just as many points in "flat" regions as for those with ones that have high slopes. It is for this reason that we seek data sets that have uniform coverage of the X space. If you feel that more data points are needed, you may want to simply add them more uniformly over the entire space to try to improve accuracy.

While the above intuitive ideas do help give directions on how to choose G for many applications, they cannot always be put directly into use. The reason for this is that for many applications (e.g., system identification) we cannot directly pick the data pairs in G. Notice that since our input portion of the input-output training data pairs (i.e., x) is typically of the form shown in Equation (5.9), x actually contains *both* the inputs and the outputs of the system. It is for this reason that it is not easy to pick an input to the system u that will ensure that the outputs y will have appropriate values so that we get x values that uniformly cover the space X. Similar problems may exist for other applications (e.g., parameter estimation), but for some applications this may not be a problem. For instance, in constructing a

fuzzy controller from human decision-making data, we may be able to ensure that we have the human provide data on how to respond to a whole range of input data (i.e., we may have full control over what the input portion of the training data in G is).

It is interesting to note that there are fundamental relationships between a data set that has uniform coverage of X and the idea of "sufficiently rich" signals in system identification (i.e., "persistency of excitation" in adaptive systems). Intuitively, for system identification we must choose a signal u to "excite" the dynamics of the system so that we can "see," via the plant input-output data, what the dynamics are that generated the output data. Normally, constraints from conventional linear system identification will require that, for example, a certain number of sinusoids be present in the signal u to be able to estimate a certain number of parameters. The idea is that if we excite more modes of the system, we will be able to identify these modes. Following this line of reasoning, if we use white noise for the input u, then we should excite all frequencies of the system—and therefore we should be able to better identify the dynamics of the plant.

Excitation with a noise signal will have a tendency to place points in X over a whole range of locations; however, there is no guarantee that uniform coverage will be achieved for nonlinear identification problems with standard ideas from conventional linear identification. Hence, it is a difficult problem to know how to pick u so that G is a good data set for solving a function approximation problem. Sometimes we will be able to make a choice for u that makes sense for a particular application. For other applications, excitation with noise may be the best choice that you can make since it can be difficult to pick the input u that results in a better data set G; however, sometimes putting noise into the system is not really a viable option due to practical considerations.

5.2.4 Incorporating Linguistic Information

While we have focused above on how best to construct the numerical data set G so that it provides us with good information on how to construct f, it is important not to ignore the basic idea from the earlier chapters that linguistic information has a valuable role to play in the construction of a fuzzy system. In this section we explain how all the methods treated in this chapter can be easily modified so that linguistic information can be used together with the numerical data in G to construct the fuzzy system.

Suppose that we call f the fuzzy system that is constructed with one of the techniques described in this chapter—that is, from numerical data. Now, suppose that we have some linguistic information and with it we construct another fuzzy system that we denote with f_L. If we are studying a system identification problem, then f_L may contain heuristic knowledge about how the plant outputs will respond to its inputs. For specific applications, it is often easy to specify such information, especially if it just characterizes the gross behavior of the plant. If we are studying how to construct a controller, then just as we did in Chapters 2 and 3, we may know something about how to construct the controller in addition to the numerical

data about the decision-making process. If so, then this can be loaded into f_L. If we are studying an estimation or prediction problem, then we can provide similar heuristic information about guesses at what the estimate or prediction should be given certain system input-output data.

Suppose that the fuzzy system f_L is in the same basic form (in terms of its inference strategy, fuzzification, and defuzzification techniques) as f, the one constructed with numerical data. Then to combine the linguistic information in f_L with the fuzzy system f that we constructed from numerical data, we simply need to combine the two fuzzy systems. There are many ways to do this. You could merge the two rule-bases then treat the combined rule-base as a single rule-base. Alternatively, you could interpolate between the outputs of the two fuzzy systems, perhaps with another fuzzy system. Here, we will explain how to merge the two fuzzy systems using one rule-base merging method. It will then be apparent how to incorporate linguistic information by combining fuzzy systems for the variety of other possible cases (e.g., merging information from two different types of fuzzy systems such as the standard fuzzy system and the Takagi-Sugeno fuzzy system).

Suppose that the fuzzy system we constructed from numerical data is given by

$$f(x) = \frac{\sum_{i=1}^{R} b_i \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

where

$$\mu_i(x) = \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)$$

It uses singleton fuzzification, Gaussian membership functions, product for the premise and implication, and center-average defuzzification. It has R rules, output membership function centers at b_i, input membership function centers at c_j^i, and input membership function spreads σ_j^i. Suppose that the additional linguistic information is described with a fuzzy system

$$f_L(x) = \frac{\sum_{i=1}^{R_L} b_i^L \mu_i^L(x)}{\sum_{i=1}^{R_L} \mu_i^L(x)}$$

where

$$\mu_i^L(x) = \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - \overline{c}_j^i}{\overline{\sigma}_j^i}\right)^2\right)$$

This fuzzy system has R_L rules, output membership function centers at b_i^L, input membership function centers at $\overline{c}_j^i$, and input membership function widths $\overline{\sigma}_j^i$.

The combined fuzzy system f_C can be defined by

$$f_C(x) = \frac{\sum_{i=1}^{R} b_i \mu_i(x) + \sum_{i=1}^{R_L} b_i^L \mu_i^L(x)}{\sum_{i=1}^{R} \mu_i(x) + \sum_{i=1}^{R_L} \mu_i^L(x)}$$

This fuzzy system is obtained by concatenating the rule-bases for the two fuzzy systems, and this equation provides a mathematical description of how this is done. This combination approach results in a fuzzy system that has the same basic form as the fuzzy systems that it is made of.

Overall, we would like to emphasize that at times it can be very beneficial to include heuristic information via the judicious choice of f_L. Indeed, at times it can make the difference between the success or failure of the methods of this chapter. Also, some would say that our ability to easily incorporate heuristic knowledge via f_L is one of the advantages of fuzzy over neural or conventional identification and estimation methods.

5.2.5 Case Study: Engine Failure Data Sets

In this section we will show how to choose the training data for a case study that we will use in the homework problems of this chapter. In particular, we will establish an engine failure simulator for the generation of data to train a failure estimator (a type of parameter estimator) for an internal combustion engine.

Engine Failure Simulator

An engine failure simulator takes engine inputs and and uses an engine model with specified parameters to produce engine outputs. When the engine parameters are varied, the failure simulator produces an output corresponding to the varied parameters. In this case study we use the engine model shown in Figure 5.3, with parameters defined in Table 5.1, which was developed in [174]. This particular model is a crude representation of a fuel-injected internal combustion engine. It describes the throttle to engine speed dynamics, taking into account some of the dynamics from other engine subsystems. The engine model includes a throttle position sensor for Θ, manifold absolute pressure (MAP) sensor for P_m, and a sensor for the engine speed N. The model describes the intake manifold dynamics, the pressure to torque map, the rotating dynamics of the engine, including the inherent frictional losses, and the load torque due to outside disturbances, T_L. Under normal vehicle operation, the system contains nonlinearities that make modeling the engine quite complex. Some of these nonlinearities are determined by the speed and throttle and can be linearized about an idle speed operating point, as was done with this model. While such a simple model does not represent the complete engine dynamics, it proves adequate for our failure estimation example, as it has the ability to roughly model the failure modes that we are interested in.

There are several inputs to the failure simulator: the throttle position Θ; the parameters k_1–k_7, whose variations represent failures; and the load torque disturbance T_L. Recall that we will use different conditions for training and testing the

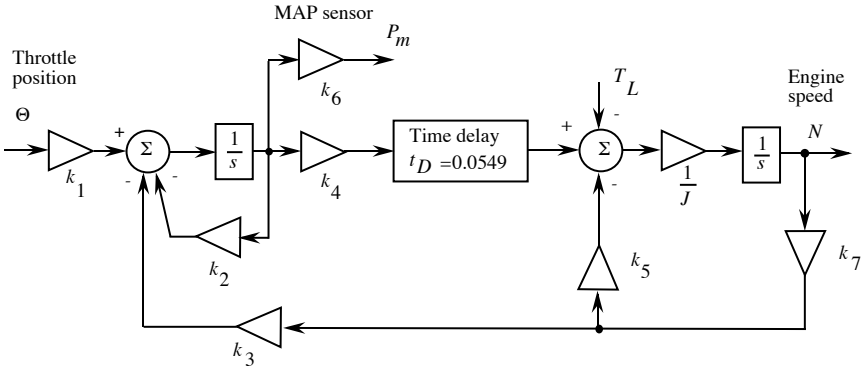


FIGURE 5.3 Linearized engine model (figure drawn by Sashonda Morris).

TABLE 5.1 Parameter Values for an Operating Condition

Engine speed	N	1092.6 rpm
Throttle position	Θ	10.22% of full throttle
Change in input throttle angle	k_1	949.23 kPa/second-volts
Change in intake manifold	k_2	6.9490 kPa/second-kPa
Change in engine pumping	k_3	0.3787 kPa/second-rpm
Change in combustion characteristic	k_4	0.8045 (Nt-m)/kPa
Change in the engine friction	k_5	0.0246 (Nt-m)/rpm
Change in air pressure intake manifold	k_6	1.0000 kPa/second-kPa
Change in speedometer sensor	k_7	1.0000 (Nt-m)/rpm
Time delay	t_D	0.0549 second
Inertia	J	0.00332 (Nt-m second)/rpm

accuracy of our estimator. For training, the input Θ is a "staircase step," with amplitude ranging from 0.1 to 0.025, as shown in Figure 5.4. For testing, the input Θ is a constant step of 0.1. For training, we set $T_L = 0$. For testing, the load torque disturbance T_L is shown in Figure 5.4 where we use a height of 5 Nm, a start time of 0.1 sec, a period of 3 sec, and a width of 0.2 sec. This type of disturbance corresponds to the load placed on the engine due to the on/off cycling of the air conditioner compressor. Since we are using a linearized model, the values of the step correspond to the change in the input throttle angle and load torque around the idle speed. Note that we use $T_L = 0$ for training since this represents that we do not know how the load torque will influence the system a priori. Then, in testing we can evaluate the ability of our estimator to perform under conditions that it was not originally designed for.

To modify the gains k_1–k_7 in the engine model in Figure 5.3 to represent failures, we use

$$k_i(\text{failure}) = k_i(\text{nominal}) + \Delta k_i \times k_i(\text{nominal}) \tag{5.11}$$

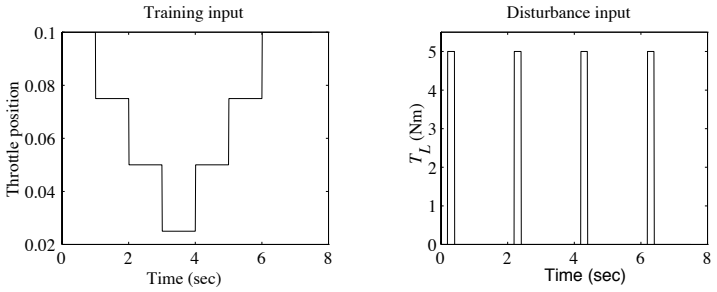


FIGURE 5.4 Throttle position Θ and load torque T_L (plots created by Sashonda Morris).

where

$$\Delta k_i = \frac{\%\text{ of failure}}{100} \tag{5.12}$$

$\Delta k_i \in [-1.0, +1.0\,]$, $i \in \{1, 2, \ldots, 7\}$, and the k_i(nominal), are the values of the parameters k_i given in Table 5.1. The percentage of failure can be any value between $\pm 100\%$. If $\Delta k_i = 0$, then k_i(failure) $= k_i$(nominal), and no failure occurred for that particular parameter. If $\Delta k_i \neq 0$, then the value of the nominal gain is increased for $\Delta k_i > 0$ and decreased for $\Delta k_i < 0$.

Engine Failure Scenarios

The failure simulator is capable of simulating throttle position, manifold absolute pressure, and vehicle speed sensor failures. It also has the ability to simulate various plant and actuator failures, such as the change in engine pumping and change in combustion characteristics. In this case study, the intake manifold coefficient, k_2, and the frictional coefficient, k_5, were varied for failure estimation purposes. A decrease in k_2 represents a vacuum or gasket leak, which results from a cracked or loose gasket. Under these conditions, the engine may idle rough, stall, or yield poor fuel economy. An increase in k_5 indicates excessive engine friction resulting from an excessive loss in torque. This condition may result in engine knocking, backfiring, surging at steady speed, or a lack of engine power. Our objective is to develop a parameter estimator for these parameters so that we can provide an indication if there has been a failure in the engine.

The two failure scenarios are shown in Table 5.2. The scenarios represent single parameter engine faults. Figure 5.5 shows the output responses for the specified failure scenarios. These will be used to test the fuzzy parameter estimators for k_2 and k_5 after they are constructed. The first plot in the figure indicates normal operation of the engine when Θ is a step input of amplitude 0.1. The last two plots illustrate the output responses for the failure scenarios specified in Table 5.2. The failures were induced at the beginning of the simulation. Notice that a k_2 failure

results in an increase in overshoot and some steady-state error, while a k_5 failure results in a significant steady-state error.

TABLE 5.2 Failure Scenarios for Automotive Engine

Failure Scenarios	Gain	Original Value	Failure Setting
Leakage in gasket	k_2	6.9490 kPa/second-kPa	-50%
Excessive engine friction	k_5	0.0246 (Nt-m)/rpm	$+100\%$

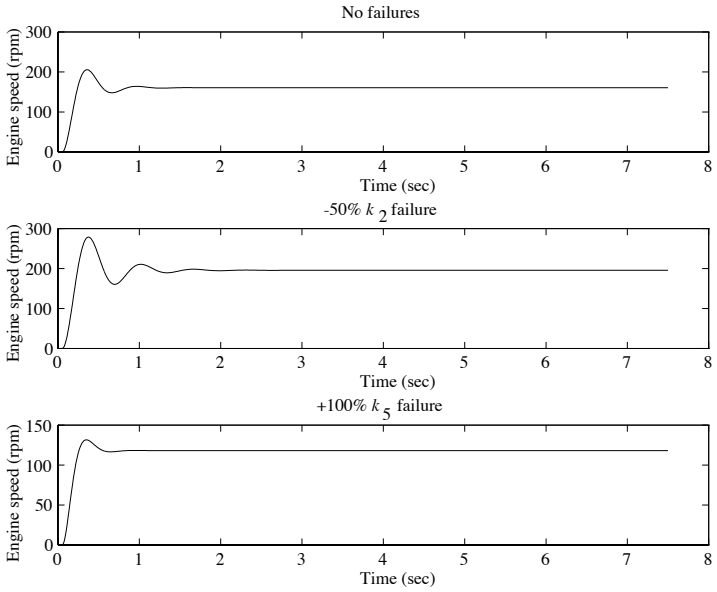


FIGURE 5.5 Output responses for the automotive engine failure scenarios (plots created by Sashonda Morris).

The Training Data Set

To train the fuzzy parameter estimator, the training input for the throttle position Θ shown in Figure 5.4 is used as the input to the engine. Using this input, the engine failure simulator produces output responses corresponding to the system parameters. Varying a single parameter over a range of values yields different responses, with each one corresponding to the value of the parameter. For our purposes, the parameters k_2 and k_5 were varied individually over a specified range of values to account for the possible failure scenarios the system might encounter. The parameter k_2 was varied between -50% and $+50\%$ of its nominal value (i.e.,

$\Delta k_2 \in [-0.5, +0.5]$), and k_5 was varied between +100% and +200% of its nominal value (i.e., $\Delta k_5 \in [+1, +2]$). The parameters k_2 and k_5 were varied at 5% and 10% increments, yielding $M_{k_2} = 21$ and $M_{k_5} = 11$ responses, which are shown in Figures 5.6 and 5.7. These plots represent how the engine will behave over a variety of failure conditions.

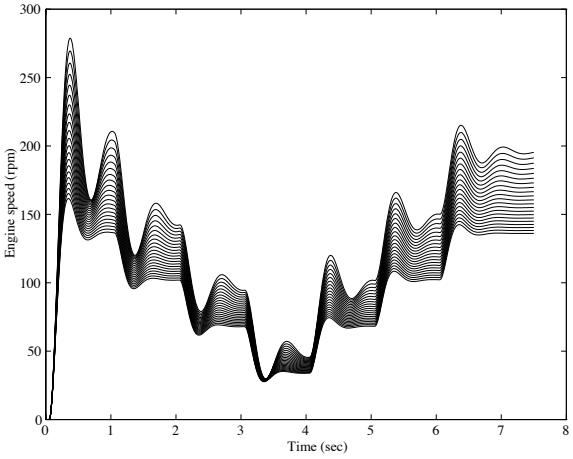


FIGURE 5.6 Automotive engine rpm for $\Delta k_2 \in [-0.5, 0.5]$ (plots created by Sashonda Morris).

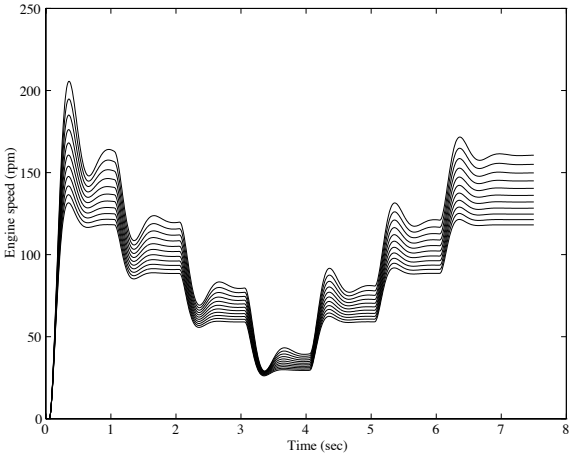


FIGURE 5.7 Automotive engine rpm for $\Delta k_5 \in [+1, +2]$ (plots created by Sashonda Morris).

The output responses were sampled with a sampling period of $T = 0.25$ sec to form the engine failure data sets. In particular, the full set of engine failure data is given by

$$G_{k_i} = \{([\Theta_i^j(kT), N_i^j(kT), N_i^j(kT-T)]^\top, k_i^j) : k \in \{1, 2, \ldots, 30\}, \\ \text{if } i = 2, 1 \le j \le M_{k_2}, \text{ and if } i = 5, 1 \le j \le M_{k_5}\} \quad (5.13)$$

where k_i^j denotes the j^{th} value ($1 \le j \le M_{k_i}$) of k_i and $\Theta_i^j(kT)$, $N_i^j(kT)$, and $N_i^j(kT-T)$ represent the corresponding values of $\Theta(kT)$, $N(kT)$, and $N(kT-T)$ that were generated using this k_i^j (note that "k" denotes a time index while k_i and k_i^j denote parameter values). Hence, G_{k_2} (G_{k_5}) is the set of data that we will use in the problems at the end of the chapter to train fuzzy systems to estimate the value of k_2 (k_5). Notice that the number of training data points in G_{k_i} is $30M_{k_i}$, $i = 2, 5$.

We choose $x = [\Theta_i^j(kT), N_i^j(kT), N_i^j(kT-T)]^\top$ since the value of k_i depends on the size of Θ, the size of $N(kT)$, and the rate of change of $N(kT)$. Also, we chose to have more training points in the k_2 data set since we found it somewhat more difficult to estimate. Notice also that we choose the G_{k_i} to represent a range of failures, and for illustration purposes we will test the performance of our estimators near the end of these ranges (i.e., a -50% failure on k_2 and a $+100\%$ failure on k_5). Generally, it is often better to train for a whole range of failures *around* where you expect the failed parameter values to be. For this reason, estimators developed based on these training data will tend to be worse than what is possible to obtain (we made this choice for testing our fuzzy parameter estimation systems for illustrative purposes to show that even at the limits of the training data it is possible for you get reasonably good estimation results).

5.3 Least Squares Methods

In this section we will introduce batch and recursive least squares methods for constructing a linear system to match some input-output data. Following this, we explain how these methods can be directly used for training fuzzy systems. We begin by discussing least squares methods as they are simple to understand and have clear connections to conventional estimation methods. We also present them first since they provide for the training of only certain parameters of a fuzzy system (e.g., the output membership function centers). Later, we will provide methods that can be used to tune all the fuzzy system's parameters.

5.3.1 Batch Least Squares

We will introduce the batch least squares method to train fuzzy systems by first discussing the solution of the linear system identification problem. Let g denote the physical system that we wish to identify. The training set G is defined by the experimental input-output data that is generated from this system. In linear system

identification, we can use a model

$$y(k) = \sum_{i=1}^{\bar{q}} \theta_{a_i} y(k-i) + \sum_{i=0}^{\bar{p}} \theta_{b_i} u(k-i)$$

where $u(k)$ and $y(k)$ are the system input and output at time k. In this case $f(x|\theta)$, which is not a fuzzy system, is defined by

$$f(x|\theta) = \theta^\top x(k) \tag{5.14}$$

where we recall that

$$x(k) = [y(k-1), \cdots, y(k-\bar{q}), u(k), \cdots, u(k-\bar{p})]^\top$$

and

$$\theta = [\theta_{a_1}, \cdots, \theta_{a_{\bar{q}}}, \theta_{b_0}, \cdots, \theta_{b_{\bar{p}}}]^\top$$

We have $N = \bar{q} + \bar{p} + 1$ so that $x(k)$ and θ are $N \times 1$ vectors, and often $x(k)$ is called the "regression vector."

Recall that system identification amounts to adjusting θ using information from G so that $f(x|\theta) \approx g(x)$ for all $x \in X$. Often, to form G for linear system identification we choose $x^i = x(i)$, $y^i = y(i)$, and let $G = \{(x^i, y^i) : i = 1, 2, \ldots, M\}$. To do this you will need appropriate initial conditions.

Batch Least Squares Derivation

In the batch least squares method we define

$$Y(M) = \left[y^1, y^2, \ldots, y^M\right]^\top$$

to be an $M \times 1$ vector of output data where the y^i, $i = 1, 2, \ldots, M$ come from G (i.e., y^i such that $(x^i, y^i) \in G$). We let

$$\Phi(M) = \begin{bmatrix} (x^1)^\top \\ (x^2)^\top \\ \vdots \\ (x^M)^\top \end{bmatrix}$$

be an $M \times N$ matrix that consists of the x^i data vectors stacked into a matrix (i.e., the x^i such that $(x^i, y^i) \in G$). Let

$$\epsilon_i = y^i - (x^i)^\top \theta$$

be the error in approximating the data pair $(x^i, y^i) \in G$ using θ. Define

$$E(M) = [\epsilon_1, \epsilon_2, \ldots, \epsilon_M]^\top$$

so that

$$E = Y - \Phi\theta$$

Choose

$$V(\theta) = \frac{1}{2} E^\top E$$

to be a measure of how good the approximation is for all the data for a given θ. We want to pick θ to minimize $V(\theta)$. Notice that $V(\theta)$ is convex in θ so that a local minimum is a global minimum.

Now, using basic ideas from calculus, if we take the partial of V with respect to θ and set it equal to zero, we get an equation for $\hat{\theta}$, the best estimate (in the least squares sense) of the unknown θ. Another approach to deriving this is to notice that

$$2V = E^\top E = Y^\top Y - Y^\top \Phi\theta - \theta^\top \Phi^\top Y + \theta^\top \Phi^\top \Phi\theta$$

Then, we "complete the square" by assuming that $\Phi^\top \Phi$ is invertible and letting

$$\begin{aligned} 2V = {} & Y^\top Y - Y^\top \Phi\theta - \theta^\top \Phi^\top Y + \theta^\top \Phi^\top \Phi\theta \\ & + Y^\top \Phi(\Phi^\top \Phi)^{-1} \Phi^\top Y - Y^\top \Phi(\Phi^\top \Phi)^{-1} \Phi^\top Y \end{aligned}$$

(where we are simply adding and subtracting the same terms at the end of the equation). Hence,

$$2V = Y^\top (I - \Phi(\Phi^\top \Phi)^{-1} \Phi^\top) Y + (\theta - (\Phi^\top \Phi)^{-1} \Phi^\top Y)^\top \Phi^\top \Phi (\theta - (\Phi^\top \Phi)^{-1} \Phi^\top Y)$$

The first term in this equation is independent of θ, so we cannot reduce V via this term, so it can be ignored. Hence, to get the smallest value of V, we choose θ so that the second term is zero. We will denote the value of θ that achieves the minimization of V by $\hat{\theta}$, and we notice that

$$\hat{\theta} = (\Phi^\top \Phi)^{-1} \Phi^\top Y \tag{5.15}$$

since the smallest we can make the last term in the above equation is zero. This is the equation for batch least squares that shows we can directly compute the least squares estimate $\hat{\theta}$ from the "batch" of data that is loaded into Φ and Y. If we pick the inputs to the system so that it is "sufficiently excited" [127], then we will be guaranteed that $\Phi^\top \Phi$ is invertible; if the data come from a linear plant with known $\bar{q}$ and $\bar{p}$, then for sufficiently large M we will achieve perfect estimation of the plant parameters.

In "weighted" batch least squares we use

$$V(\theta) = \frac{1}{2} E^\top W E \tag{5.16}$$

where, for example, W is an $M \times M$ diagonal matrix with its diagonal elements $w_i > 0$ for $i = 1, 2, \ldots, M$ and its off-diagonal elements equal to zero. These w_i can be used to weight the importance of certain elements of G more than others. For example, we may choose to have it put less emphasis on older data by choosing $w_1 < w_2 < \cdots < w_M$ when x^2 is collected after x^1, x^3 is collected after x^2, and so on. The resulting parameter estimates can be shown to be given by

$$\hat{\theta}_{wbls} = (\Phi^\top W \Phi)^{-1} \Phi^\top W Y \tag{5.17}$$

To show this, simply use Equation (5.16) and proceed with the derivation in the same manner as above.

Example: Fitting a Line to Data

As an example of how batch least squares can be used, suppose that we would like to use this method to fit a line to a set of data. In this case our parameterized model is

$$y = x_1 \theta_1 + x_2 \theta_2 \tag{5.18}$$

Notice that if we choose $x_2 = 1$, y represents the equation for a line. Suppose that the data that we would like to fit the line to is given by

$$\left\{ \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, 1 \right), \left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}, 1 \right), \left(\begin{bmatrix} 3 \\ 1 \end{bmatrix}, 3 \right) \right\}$$

Notice that to train the parameterized model in Equation (5.18) we have chosen $x_2^i = 1$ for $i = 1, 2, 3 = M$. We will use Equation (5.15) to compute the parameters for the line that best fits the data (in the sense that it will minimize the sum of the squared distances between the line and the data). To do this we let

$$\Phi = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}$$

and

$$Y = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$$

Hence,

$$\hat{\theta} = (\Phi^\top \Phi)^{-1} \Phi^\top Y = \left(\begin{bmatrix} 14 & 6 \\ 6 & 3 \end{bmatrix} \right)^{-1} \begin{bmatrix} 12 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{1}{3} \end{bmatrix}$$

Hence, the line

$$y = x_1 - \frac{1}{3}$$

best fits the data in the least squares sense. We leave it to the reader to plot the data points and this line on the same graph to see pictorially that it is indeed a good fit to the data.

The same general approach works for larger data sets. The reader may want to experiment with weighted batch least squares to see how the weights w_i affect the way that the line will fit the data (making it more or less important that the data fit at certain points).

5.3.2 Recursive Least Squares

While the batch least squares approach has proven to be very successful for a variety of applications, it is by its very nature a "batch" approach (i.e., all the data are gathered, then processing is done). For small M we could clearly repeat the batch calculation for increasingly more data as they are gathered, but the computations become prohibitive due to the computation of the inverse of $\Phi^\top \Phi$ and due to the fact that the dimensions of Φ and Y depend on M. Next, we derive a recursive version of the batch least squares method that will allow us to update our $\hat{\theta}$ estimate each time we get a new data pair, without using all the old data in the computation and without having to compute the inverse of $\Phi^\top \Phi$.

Since we will be considering successively increasing the size of G, and we will assume that we increase the size by one each time step, we let a time index $k = M$ and i be such that $0 \leq i \leq k$. Let the $N \times N$ matrix

$$P(k) = (\Phi^\top \Phi)^{-1} = \left(\sum_{i=1}^{k} x^i (x^i)^\top \right)^{-1} \tag{5.19}$$

and let $\hat{\theta}(k-1)$ denote the least squares estimate based on $k-1$ data pairs ($P(k)$ is called the "covariance matrix"). Assume that $\Phi^\top \Phi$ is nonsingular for all k. We have $P^{-1}(k) = \Phi^\top \Phi = \sum_{i=1}^{k} x^i (x^i)^\top$ so we can pull the last term from the summation to get

$$P^{-1}(k) = \sum_{i=1}^{k-1} x^i (x^i)^\top + x^k (x^k)^\top$$

and hence

$$P^{-1}(k) = P^{-1}(k-1) + x^k (x^k)^\top \tag{5.20}$$

Now, using Equation (5.15) we have

$$\begin{aligned}
\hat{\theta}(k) &= (\Phi^\top \Phi)^{-1} \Phi^\top Y \\
&= \left(\sum_{i=1}^{k} x^i (x^i)^\top \right)^{-1} \left(\sum_{i=1}^{k} x^i y^i \right) \\
&= P(k) \left(\sum_{i=1}^{k} x^i y^i \right) \\
&= P(k) \left(\sum_{i=1}^{k-1} x^i y^i + x^k y^k \right)
\end{aligned} \tag{5.21}$$

Hence,

$$\hat{\theta}(k-1) = P(k-1) \sum_{i=1}^{k-1} x^i y^i$$

and so

$$P^{-1}(k-1)\hat{\theta}(k-1) = \sum_{i=1}^{k-1} x^i y^i$$

Now, replacing $P^{-1}(k-1)$ in this equation with the result in Equation (5.20), we get

$$(P^{-1}(k) - x^k (x^k)^\top)\hat{\theta}(k-1) = \sum_{i=1}^{k-1} x^i y^i$$

Using the result from Equation (5.21), this gives us

$$\begin{aligned}
\hat{\theta}(k) &= P(k)(P^{-1}(k) - x^k (x^k)^\top)\hat{\theta}(k-1) + P(k) x^k y^k \\
&= \hat{\theta}(k-1) - P(k) x^k (x^k)^\top \hat{\theta}(k-1) + P(k) x^k y^k \\
&= \hat{\theta}(k-1) + P(k) x^k (y^k - (x^k)^\top \hat{\theta}(k-1)).
\end{aligned} \tag{5.22}$$

This provides a method to compute an estimate of the parameters $\hat{\theta}(k)$ at each time step k from the past estimate $\hat{\theta}(k-1)$ and the latest data pair that we received, (x^k, y^k). Notice that $(y^k - (x^k)^\top \hat{\theta}(k-1))$ is the error in predicting y^k using $\hat{\theta}(k-1)$.

To update $\hat{\theta}$ in Equation (5.22) we need $P(k)$, so we could use

$$P^{-1}(k) = P^{-1}(k-1) + x^k (x^k)^\top \tag{5.23}$$

But then we will have to compute an inverse of a matrix at each time step (i.e., each time we get another set of data). Clearly, this is not desirable for real-time implementation, so we would like to avoid this. To do so, recall that the "matrix inversion lemma" indicates that if A, C, and $(C^{-1}+DA^{-1}B)$ are nonsingular square matrices, then $A+BCD$ is invertible and

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1}+DA^{-1}B)^{-1}DA^{-1}$$

We will use this fact to remove the need to compute the inverse of $P^{-1}(k)$ that comes from Equation (5.23) so that it can be used in Equation (5.22) to update $\hat{\theta}$. Notice that

$$\begin{aligned} P(k) &= (\Phi^\top(k)\Phi(k))^{-1} \\ &= (\Phi^\top(k-1)\Phi(k-1) + x^k(x^k)^\top)^{-1} \\ &= (P^{-1}(k-1) + x^k(x^k)^\top)^{-1} \end{aligned}$$

and that if we use the matrix inversion lemma with $A = P^{-1}(k-1)$, $B = x^k$, $C = I$, and $D = (x^k)^\top$, we get

$$P(k) = P(k-1) - P(k-1)x^k(I + (x^k)^\top P(k-1)x^k)^{-1}(x^k)^\top P(k-1) \quad (5.24)$$

which together with

$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)x^k(y^k - (x^k)^\top\hat{\theta}(k-1)) \quad (5.25)$$

(that was derived in Equation (5.22)) is called the "recursive least squares (RLS) algorithm." Basically, the matrix inversion lemma turns a matrix inversion into the inversion of a scalar (i.e., the term $(I + (x^k)^\top P(k-1)x^k)^{-1}$ is a scalar).

We need to initialize the RLS algorithm (i.e., choose $\hat{\theta}(0)$ and $P(0)$). One approach to do this is to use $\hat{\theta}(0) = 0$ and $P(0) = P_0$ where $P_0 = \alpha I$ for some large $\alpha > 0$. This is the choice that is often used in practice. Other times, you may pick $P(0) = P_0$ but choose $\hat{\theta}(0)$ to be the best guess that you have at what the parameter values are.

There is a "weighted recursive least squares" (WRLS) algorithm also. Suppose that the parameters of the physical system θ vary slowly. In this case it may be advantageous to choose

$$V(\theta, k) = \frac{1}{2}\sum_{i=1}^{k} \lambda^{k-i}(y^i - (x^i)^\top\theta)^2$$

where $0 < \lambda \leq 1$ is called a "forgetting factor" since it gives the more recent data higher weight in the optimization (note that this performance index V could also be used to derive weighted batch least squares). Using a similar approach to the

above, you can show that the equations for WRLS are given by

$$P(k) = \frac{1}{\lambda}\left(I - P(k-1)x^k(\lambda I + (x^k)^\top P(k-1)x^k)^{-1}(x^k)^\top\right)P(k-1) \quad (5.26)$$
$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)x^k(y^k - (x^k)^\top\hat{\theta}(k-1))$$

(where when $\lambda = 1$ we get standard RLS). This completes our description of the least squares methods. Next, we will discuss how they can be used to train fuzzy systems.

5.3.3 Tuning Fuzzy Systems

It is possible to use the least squares methods described in the past two sections to tune fuzzy systems either in a batch or real-time mode. In this section we will explain how to tune both standard and Takagi-Sugeno fuzzy systems that have many inputs and only one output. To train fuzzy systems with many outputs, simply repeat the procedure described below for each output.

Standard Fuzzy Systems

First, we consider a fuzzy system

$$y = f(x|\theta) = \frac{\sum_{i=1}^R b_i\mu_i(x)}{\sum_{i=1}^R \mu_i(x)} \quad (5.27)$$

where $x = [x_1, x_2, \ldots, x_n]^\top$ and $\mu_i(x)$ is defined in Chapter 2 as the certainty of the premise of the i^{th} rule (it is specified via the membership functions on the input universe of discourse together with the choice of the method to use in the triangular norm for representing the conjunction in the premise). The b_i, $i = 1, 2, \ldots, R$, values are the centers of the output membership functions. Notice that

$$f(x|\theta) = \frac{b_1\mu_1(x)}{\sum_{i=1}^R \mu_i(x)} + \frac{b_2\mu_2(x)}{\sum_{i=1}^R \mu_i(x)} + \cdots + \frac{b_R\mu_R(x)}{\sum_{i=1}^R \mu_i(x)}$$

and that if we define

$$\xi_i(x) = \frac{\mu_i(x)}{\sum_{i=1}^R \mu_i(x)} \quad (5.28)$$

then

$$f(x|\theta) = b_1\xi_1(x) + b_2\xi_2(x) + \cdots + b_R\xi_R(x)$$

Hence, if we define

$$\xi(x) = [\xi_1, \xi_2, \ldots, \xi_R]^\top$$

and

$$\theta = [b_1, b_2, \ldots, b_R]^\top$$

then

$$y = f(x|\theta) = \theta^\top \xi(x) \tag{5.29}$$

We see that the form of the model to be tuned is in only a slightly different form from the standard least squares case in Equation (5.14). In fact, if the μ_i are given, then $\xi(x)$ is given so that it is in *exactly* the right form for use by the standard least squares methods since we can view $\xi(x)$ as a known regression vector. Basically, the training data x^i are mapped into $\xi(x^i)$ and the least squares algorithms produce an estimate of the best centers for the output membership function centers b_i.

This means that either batch or recursive least squares can be used to train certain types of fuzzy systems (ones that can be parameterized so that they are "linear in the parameters," as in Equation (5.29)). All you have to do is replace x^i with $\xi(x^i)$ in forming the Φ vector for batch least squares, and in Equation (5.26) for recursive least squares. Hence, we can achieve either on- or off-line training of certain fuzzy systems with least squares methods. If you have some heuristic ideas for the choice of the input membership functions and hence $\xi(x)$, then this method can, at times, be quite effective (of course any known function can be used to replace any of the ξ_i in the $\xi(x)$ vector). We have found that some of the standard choices for input membership functions (e.g., uniformly distributed ones) work very well for some applications.

Takagi-Sugeno Fuzzy Systems

It is interesting to note that Takagi-Sugeno fuzzy systems, as described in Section 2.3.7 on page 73, can also be parameterized so that they are linear in the parameters, so that they can also be trained with either batch or recursive least squares methods. In this case, if we can pick the membership functions appropriately (e.g., using uniformly distributed ones), then we can achieve a nonlinear interpolation between the linear output functions that are constructed with least squares.

In particular, as explained in Chapter 2, a Takagi-Sugeno fuzzy system is given by

$$y = \frac{\sum_{i=1}^{R} g_i(x)\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

where

$$g_i(x) = a_{i,0} + a_{i,1}x_1 + \cdots + a_{i,n}x_n$$

Hence, using the same approach as for standard fuzzy systems, we note that

$$y = \frac{\sum_{i=1}^{R} a_{i,0}\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} + \frac{\sum_{i=1}^{R} a_{i,1}x_1\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} + \cdots + \frac{\sum_{i=1}^{R} a_{i,n}x_n\mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

We see that the first term is the standard fuzzy system. Hence, use the $\xi_i(x)$ defined in Equation (5.28) and redefine $\xi(x)$ and θ to be

$$\xi(x) = [\xi_1(x), \xi_2(x), \ldots, \xi_R(x), x_1\xi_1(x), x_1\xi_2(x), \ldots, x_1\xi_R(x), \ldots, x_n\xi_1(x), x_n\xi_2(x), \ldots, x_n\xi_R(x)]^\top$$

and

$$\theta = [a_{1,0}, a_{2,0}, \ldots, a_{R,0}, a_{1,1}, a_{2,1}, \ldots, a_{R,1}, \ldots, a_{1,n}, a_{2,n}, \ldots, a_{R,n}]^\top$$

so that

$$f(x|\theta) = \theta^\top \xi(x)$$

represents the Takagi-Sugeno fuzzy system, and we see that it too is linear in the parameters. Just as for a standard fuzzy system, we can use batch or recursive least squares for training $f(x|\theta)$. To do this, simply pick (a priori) the $\mu_i(x)$ and hence the $\xi_i(x)$ vector, process the training data x^i where $(x^i, y^i) \in G$ through $\xi(x)$, and replace x^i with $\xi(x^i)$ in forming the Φ vector for batch least squares, or in Equation (5.26) for recursive least squares.

Finally, note that the above approach to training will work for any nonlinearity that is linear in the parameters. For instance, if there are known nonlinearities in the system of the quadratic form, you can use the same basic approach as the one described above to specify the parameters of consequent functions that are quadratic (what is $\xi(x)$ in this case?).

5.3.4 Example: Batch Least Squares Training of Fuzzy Systems

As an example of how to train fuzzy systems with batch least squares, we will consider how to tune the fuzzy system

$$f(x|\theta) = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}$$

(however, other forms may be used equally effectively). Here, b_i is the point in the output space at which the output membership function for the i^{th} rule achieves a maximum, c_j^i is the point in the j^{th} input universe of discourse where the membership function for the i^{th} rule achieves a maximum, and $\sigma_j^i > 0$ is the relative width of the membership function for the j^{th} input and the i^{th} rule. Clearly, we are using

center-average defuzzification and product for the premise and implication. Notice that the outermost input membership functions do not saturate as is the usual case in control.

We will tune $f(x|\theta)$ to interpolate the data set G given in Equation (5.3) on page 236. Choosing $R = 2$ and noting that $n = 2$, we have $\theta = [b_1, b_2]^\top$ and

$$\xi_i(x) = \frac{\prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R}\prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}. \tag{5.30}$$

Next, we must pick the input membership function parameters c_j^i, $i = 1, 2$, $j = 1, 2$. One way to choose the input membership function parameters is to use the x^i portions of the first R data pairs in G. In particular, we could make the premise of rule i have unity certainty if x^i, $(x^i, y^i) \in G$, is input to the fuzzy system, $i = 1, 2, \ldots, R$, $R \leq M$. For instance, if $x^1 = [0, 2]^\top = [x_1^1, x_2^1]^\top$ and $x^2 = [2, 4]^\top = [x_1^2, x_2^2]^\top$, we would choose $c_1^1 = x_1^1 = 0$, $c_2^1 = x_2^1 = 2$, $c_1^2 = x_1^2 = 2$, and $c_2^2 = x_2^2 = 4$.

Another approach to picking the c_j^i is simply to try to spread the membership functions somewhat evenly over the input portion of the training data space. For instance, consider the axes on the left of Figure 5.2 on page 237 where the input portions of the training data are shown for G. From inspection, a reasonable choice for the input membership function centers could be $c_1^1 = 1.5$, $c_2^1 = 3$, $c_1^2 = 3$, and $c_2^2 = 5$ since this will place the peaks of the premise membership functions in between the input portions of the training data pairs. In our example, we will use this choice of the c_j^i.

Next, we need to pick the spreads σ_j^i. To do this we simply pick $\sigma_j^i = 2$ for $i = 1, 2$, $j = 1, 2$ as a guess that we hope will provide reasonable overlap between the membership functions. This completely specifies the $\xi_i(x)$ in Equation (5.30). Let $\xi(x) = [\xi_1(x), \xi_2(x)]^\top$.

We have $M = 3$ for G, so we find

$$\Phi = \begin{bmatrix} \xi^\top(x^1) \\ \xi^\top(x^2) \\ \xi^\top(x^3) \end{bmatrix} = \begin{bmatrix} 0.8634 & 0.1366 \\ 0.5234 & 0.4766 \\ 0.2173 & 0.7827 \end{bmatrix}$$

and $Y = [y^1, y^2, y^3]^\top = [1, 5, 6]^\top$. We use the batch least squares formula in Equation (5.15) on page 250 to find $\hat{\theta} = [0.3646, 8.1779]^\top$, and hence our fuzzy system is $f(x|\hat{\theta})$.

To test the fuzzy system, note that at the training data

$$f(x^1|\hat{\theta}) = 1.4320$$
$$f(x^2|\hat{\theta}) = 4.0883$$
$$f(x^3|\hat{\theta}) = 6.4798$$

so that the trained fuzzy system maps the training data reasonably accurately ($x^3 = [3, 6]^\top$). Next, we test the fuzzy system at some points not in the training data set to see how it interpolates. In particular, we find

$$f([1, 2]^\top|\hat{\theta}) = 1.8267$$
$$f([2.5, 5]^\top|\hat{\theta}) = 5.3981$$
$$f([4, 7]^\top|\hat{\theta}) = 7.3673$$

These values seem like good interpolated values considering Figure 5.2 on page 237, which illustrates the data set G for this example.

5.3.5 Example: Recursive Least Squares Training of Fuzzy Systems

Here, we illustrate the use of the RLS algorithm in Equation (5.26) on page 255 for training a fuzzy system to map the training data given in G in Equation (5.3) on page 236. First, we replace x^k with $\xi(x^k)$ in Equation (5.26) to obtain

$$P(k) = \frac{1}{\lambda}(I - P(k-1)\xi(x^k)(\lambda I + (\xi(x^k))^\top P(k-1)\xi(x^k))^{-1}(\xi(x^k))^\top)P(k-1)$$
$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)\xi(x^k)(y^k - (\xi(x^k))^\top\hat{\theta}(k-1)) \tag{5.31}$$

and we use this to compute the parameter vector of the fuzzy system. We will train the same fuzzy system that we considered in the batch least squares example of the previous section, and we pick the same c_j^i and σ_j^i, $i = 1, 2$, $j = 1, 2$ as we chose there so that we have the same $\xi(x) = [\xi_1, \xi_2]^\top$.

For initialization of Equation (5.31), we choose

$$\hat{\theta}(0) = [2, 5.5]^\top$$

as a guess of where the output membership function centers should be. Another guess would be to choose $\hat{\theta}(0) = [0, 0]^\top$. Next, using the guidelines for RLS initialization, we choose

$$P(0) = \alpha I$$

where $\alpha = 2000$. We choose $\lambda = 1$ since we do not want to discount old data, and hence we use the standard (nonweighted) RLS.

Before using Equation (5.31) to find an estimate of the output membership function centers, we need to decide in what order to have RLS process the training data pairs $(x^i, y^i) \in G$. For example, you could just take three steps with Equation (5.31), one for each training data pair. Another approach would be to use each $(x^i, y^i) \in G$ N_i times (in some order) in Equation (5.31) then stop the algorithm. Still another approach would be to cycle through all the data (i.e., (x^1, y^1) first, (x^2, y^2) second, up until (x^M, y^M) then go back to (x^1, y^1) and repeat), say, N_{RLS} times. It is this last approach that we will use and we will choose $N_{RLS} = 20$.

After using Equation (5.31) to cycle through the data N_{RLS} times, we get the last estimate

$$\hat{\theta}(N_{RLS} \cdot M) = \begin{bmatrix} 0.3647 \\ 8.1778 \end{bmatrix} \tag{5.32}$$

and

$$P(N_{RLS} \cdot M) = \begin{bmatrix} 0.0685 & -0.0429 \\ -0.0429 & 0.0851 \end{bmatrix}$$

Notice that the values produced for the estimates in Equation (5.32) are very close to the values we found with batch least squares—which we would expect since RLS is derived from batch least squares. We can test the resulting fuzzy system in the same way as we did for the one trained with batch least squares. Rather than showing the results, we simply note that since $\hat{\theta}(N_{RLS} \cdot M)$ produced by RLS is very similar to the $\hat{\theta}$ produced by batch least squares, the resulting fuzzy system is quite similar, so we get very similar values for $f(x|\hat{\theta}(N_{RLS} \cdot M))$ as we did for the batch least squares case.

5.4 Gradient Methods

As in the previous sections, we seek to construct a fuzzy system $f(x|\theta)$ that can appropriately interpolate to approximate the function g that is inherently represented in the training data G. Here, however, we use a gradient optimization method to try to pick the parameters θ that perform the best approximation (i.e., make $f(x|\theta)$ as close to $g(x)$ as possible). Unfortunately, while the gradient method tries to pick the best θ, just as for all the other methods in this chapter, there are no guarantees that it will succeed in achieving the best approximation. As compared to the least squares methods, it does, however, provide a method to tune all the parameters of a fuzzy system. For instance, in addition to tuning the output membership function centers, using this method we can also tune the input membership function centers and spreads. Next, we derive the gradient training algorithms for both standard fuzzy systems and Takagi-Sugeno fuzzy systems that have only one output. In Section 5.4.5 on page 270 we extend this to the multi-input multi-output case.

5.4.1 Training Standard Fuzzy Systems

The fuzzy system used in this section utilizes singleton fuzzification, Gaussian input membership functions with centers c_j^i and spreads σ_j^i, output membership function centers b_i, product for the premise and implication, and center-average defuzzification, and takes on the form

$$f(x|\theta) = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)} \tag{5.33}$$

Note that we use Gaussian-shaped input membership functions for the entire input universe of discourse for all inputs and do not use ones that saturate at the outermost endpoints as we often do in control. The procedure developed below works in a similar fashion for other types of fuzzy systems. Recall that c_j^i denotes the center for the i^{th} rule on the j^{th} universe of discourse, b_i denotes the center of the output membership function for the i^{th} rule, and σ_j^i denotes the spread for the i^{th} rule on the j^{th} universe of discourse.

Suppose that you are given the m^{th} training data pair $(x^m, y^m) \in G$. Let

$$e_m = \frac{1}{2}\left[f(x^m|\theta) - y^m\right]^2$$

In gradient methods, we seek to minimize e_m by choosing the parameters θ, which for our fuzzy system are b_i, c_j^i, and σ_j^i, $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$ (we will use $\theta(k)$ to denote these parameters' values at time k). Another approach would be to minimize a sum of such error values for a subset of the data in G or all the data in G; however, with this approach computational requirements increase and algorithm performance may not.

Output Membership Function Centers Update Law

First, we consider how to adjust the b_i to minimize e_m. We use an "update law" (update formula)

$$b_i(k+1) = b_i(k) - \lambda_1 \left.\frac{\partial e_m}{\partial b_i}\right|_k$$

where $i = 1, 2, \ldots, R$ and $k \geq 0$ is the index of the parameter update step. This is a "gradient descent" approach to choosing the b_i to minimize the quadratic function e_m that quantifies the error between the current data pair (x^m, y^m) and the fuzzy system. If e_m were quadratic in θ (which it is not; why?), then this update method would move b_i along the negative gradient of the e_m error surface—that is, down the (we hope) bowl-shaped error surface (think of the path you take skiing down a valley—the gradient descent approach takes a route toward the bottom of the valley). The parameter $\lambda_1 > 0$ characterizes the "step size." It indicates how big a step to take down the e_m error surface. If λ_1 is chosen too small, then b_i is adjusted very slowly. If λ_1 is chosen too big, convergence may come faster but you risk it stepping over the minimum value of e_m (and possibly never converging to a minimum). Some work has been done on adaptively picking the step size. For example, if errors are decreasing rapidly, take big steps, but if errors are decreasing slowly, take small steps. This approach attempts to speed convergence yet avoid missing a minimum.

Now, to simplify the b_i update formula, notice that using the chain rule from calculus

$$\frac{\partial e_m}{\partial b_i} = (f(x^m|\theta) - y^m)\frac{\partial f(x^m|\theta)}{\partial b_i}$$

so

$$\frac{\partial e_m}{\partial b_i} = (f(x^m|\theta) - y^m)\frac{\prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j^m - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R}\prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j^m - c_j^i}{\sigma_j^i}\right)^2\right)}$$

For notational convenience let

$$\mu_i(x^m, k) = \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j^m - c_j^i(k)}{\sigma_j^i(k)}\right)^2\right) \tag{5.34}$$

and let

$$\epsilon_m(k) = f(x^m|\theta(k)) - y^m$$

Then we get

$$b_i(k+1) = b_i(k) - \lambda_1 \epsilon_m(k)\frac{\mu_i(x^m, k)}{\sum_{i=1}^{R}\mu_i(x^m, k)} \tag{5.35}$$

as the update equation for the b_i, $i = 1, 2, \ldots, R$, $k \geq 0$.

The other parameters in θ, $c_j^i(k)$ and $\sigma_j^i(k)$, will also be updated with a gradient algorithm to try to minimize e_m, as we explain next.

Input Membership Function Centers Update Law

To train the c_j^i, we use

$$c_j^i(k+1) = c_j^i(k) - \lambda_2 \left.\frac{\partial e_m}{\partial c_j^i}\right|_k$$

where $\lambda_2 > 0$ is the step size (see the comments above on how to choose this step size), $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$, and $k \geq 0$. At time k using the chain rule,

$$\frac{\partial e_m}{\partial c_j^i} = \epsilon_m(k)\frac{\partial f(x^m|\theta(k))}{\partial \mu_i(x^m, k)}\frac{\partial \mu_i(x^m, k)}{\partial c_j^i}$$

for $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$, and $k \geq 0$. Now,

$$\frac{\partial f(x^m|\theta(k))}{\partial \mu_i(x^m, k)} = \frac{\left(\sum_{i=1}^{R}\mu_i(x^m, k)\right)b_i(k) - \left(\sum_{i=1}^{R} b_i(k)\mu_i(x^m, k)\right)(1)}{\left(\sum_{i=1}^{R}\mu_i(x^m, k)\right)^2}$$

so that

$$\frac{\partial f(x^m|\theta(k))}{\partial \mu_i(x^m,k)} = \frac{b_i(k) - f(x^m|\theta(k))}{\sum_{i=1}^R \mu_i(x^m,k)}$$

Also,

$$\frac{\partial \mu_i(x^m,k)}{\partial c_j^i} = \mu_i(x^m,k)\left(\frac{x_j^m - c_j^i(k)}{\left(\sigma_j^i(k)\right)^2}\right)$$

so we have an update method for the $c_j^i(k)$ for all $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$, and $k \geq 0$. In particular, we have

$$c_j^i(k+1) = c_j^i(k) - \lambda_2 \epsilon_m(k)\left(\frac{b_i(k) - f(x^m|\theta(k))}{\sum_{i=1}^R \mu_i(x^m,k)}\right)\mu_i(x^m,k)\left(\frac{x_j^m - c_j^i(k)}{\left(\sigma_j^i(k)\right)^2}\right) \quad (5.36)$$

for $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$, and $k \geq 0$.

Input Membership Function Spreads Update Law

To update the $\sigma_j^i(k)$ (spreads of the membership functions), we follow the same procedure as above and use

$$\sigma_j^i(k+1) = \sigma_j^i(k) - \lambda_3 \left.\frac{\partial e_m}{\partial \sigma_j^i}\right|_k$$

where $\lambda_3 > 0$ is the step size, $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$, and $k \geq 0$. Using the chain rule, we obtain

$$\frac{\partial e_m}{\partial \sigma_j^i} = \epsilon_m(k)\frac{\partial f(x^m|\theta(k))}{\partial \mu_i(x^m,k)}\frac{\partial \mu_i(x^m,k)}{\partial \sigma_j^i}$$

We have

$$\frac{\partial \mu_i(x^m,k)}{\partial \sigma_j^i} = \mu_i(x^m,k)\frac{\left(x_j^m - c_j^i(k)\right)^2}{\left(\sigma_j^i(k)\right)^3}$$

so that

$$\sigma_j^i(k+1) = \sigma_j^i(k) - \lambda_3 \epsilon_m(k)\frac{b_i(k) - f(x^m|\theta(k))}{\sum_{i=1}^R \mu_i(x^m,k)}\mu_i(x^m,k)\frac{(x_j^m - c_j^i(k))^2}{(\sigma_j^i(k))^3} \quad (5.37)$$

for $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$, and $k \geq 0$. This completes the definition of the gradient training method for the standard fuzzy system. To summarize, the equations for updating the parameters θ of the fuzzy system are Equations (5.35), (5.36), and (5.37).

Next, note that the gradient training method described above is for the case where we have Gaussian-shaped input membership functions. The update formulas would, of course, change if you were to choose other membership functions. For instance, if you use triangular membership functions, the update formulas can be developed, but in this case you will have to pay special attention to how to define the derivative at the peak of the membership function.

Finally, we would like to note that the gradient method can be used in either an off- or on-line manner. In other words, it can be used off-line to train a fuzzy system for system identification, or it can be used on-line to train a fuzzy system to perform real-time parameter estimation. We will see in Chapter 6 how to use such an adaptive parameter identifier in an adaptive control setting.

5.4.2 Implementation Issues and Example

In this section we discuss several issues that you will encounter if you implement a gradient approach to training fuzzy systems. Also, we provide an example of how to train a standard fuzzy system.

Algorithm Design

There are several issues to address in the design of the gradient algorithm for training a fuzzy system. As always, the choice of the training data G is critical. Issues in the choice of the training data, which we discussed in Section 5.2 on page 235, are relevant here. Next, note that you must pick the number of inputs n to the fuzzy system to be trained and the number of rules R; the method does not add rules, it just tunes existing ones.

The choice of the initial estimates $b_i(0)$, $c_j^i(0)$, and $\sigma_j^i(0)$ can be important. Sometimes picking them close to where they should be can help convergence. Notice that you should not pick $b_i = 0$ for all $i = 1, 2, \ldots, R$ or the algorithm for the b_i will stay at zero for all $k \geq 0$. Your computer probably will not allow you to pick $\sigma_j^i(0) = 0$ since you divide by this number in the algorithm. Also, you may need to make sure that in the algorithm $\sigma_j^i(k) \geq \bar{\sigma} > 0$ for some fixed scalar $\bar{\sigma}$ so that the algorithm does not tune the parameters of the fuzzy system so that the computer has to divide by zero (to do this, just monitor the $\sigma_j^i(k)$, and if there exists some k' where $\sigma_j^i(k') < \bar{\sigma}$, let $\sigma_j^i(k') = \bar{\sigma}$). Notice that for our choice of input membership functions

$$\sum_{i=1}^{R} \mu_i(x^m, k) \neq 0$$

so that we normally do not have to worry about dividing by it in the algorithm.

Note that the above gradient algorithm is for only *one* training data pair. That is, we could run the gradient algorithm for a long time (i.e., many values of k) for only one data pair to try to train the fuzzy system to match that data pair very well. Then we could go to the next data pair in G, begin with the final computed values of b_i, c_j^i, and σ_j^i from the last data pair we considered as the initial values for

this data pair, and run the gradient algorithm for as many steps as we would like for that data pair—and so on. Alternatively, we could cycle through the training data many times, taking one step with the gradient algorithm for each data pair. It is difficult to know how many parameter update steps should be made for each data pair and how to cycle through the data. It is generally the case, however, that if you use some of the data much more frequently than other data in G, then the trained fuzzy system will tend to be more accurate for that data rather than the data that was not used as many times in training. Some like to cycle through the data so that each data pair is visited the same number of times and use small step sizes so that the updates will not be too large in any direction.

Clearly, you must be careful with the choices for the λ_i, $i = 1, 2, 3$ step sizes as values for these that are too big can result in an unstable algorithm (i.e., θ values can oscillate or become unbounded), while values for these that are too small can result in very slow convergence. The main problem, however, is that in the general case there are no guarantees that the gradient algorithm will converge at all! Moreover, it can take a significant amount of training data and long training times to achieve good results. Generally, you can conduct some tests to see how well the fuzzy system is constructed by comparing how it maps the data pairs to their actual values; however, even if this comparison appears to indicate that the fuzzy system is mapping the data properly, there are no guarantees that it will "generalize" (i.e., interpolate) for data not in the training data set that it was trained with.

To terminate the gradient algorithm, you could wait until all the parameters stop moving or change very little over a series of update steps. This would indicate that the parameters are not being updated so the gradients must be small so we must be at a minimum of the e_m surface. Alternatively, we could wait until the e_m or $\sum_{m=1}^{M} e_m$ does not change over a fixed number of steps. This would indicate that even if the parameter values are changing, the value of e_m is not decreasing, so the algorithm has found a minimum and it can be terminated.

Example

As an example, consider the data set G in Equation (5.3) on page 236: we will train the parameters of the fuzzy system with $R = 2$ and $n = 2$. Choose $\lambda_1 = \lambda_2 = \lambda_3 = 1$. Choose

$$\begin{bmatrix} c_1^1(0) \\ c_2^1(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} \sigma_1^1(0) \\ \sigma_2^1(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_1(0) = 1$$

and

$$\begin{bmatrix} c_1^2(0) \\ c_2^2(0) \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \begin{bmatrix} \sigma_1^2(0) \\ \sigma_2^2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_2(0) = 5$$

In this way the two rules will begin by perfectly mapping the first two data pairs in G (why?). The gradient algorithm has to tune the fuzzy system so that it will

provide an approximation to the third data pair in G, and in doing this it will tend to somewhat degrade how well it represented the first two data pairs.

To train the fuzzy system, we could repeatedly cycle through the data in G so that the fuzzy system learns how to map the third data pair but does not forget how to map the first two. Here, for illustrative purposes, we will simply perform one iteration of the algorithm for the b_i parameters for the third data pair. That is, we use

$$x^m = x^3 = \begin{bmatrix} 3 \\ 6 \end{bmatrix}, \quad y^m = y^3 = 6$$

In this case we have

$$\mu_1(x^3, 0) = 0.000003724$$

and

$$\mu_2(x^3, 0) = 0.08208$$

so that $f(x^3|\theta(0)) = 4.99977$ and $\epsilon_m(0) = -1.000226$. With this and Equation (5.35), we find that $b_1(1) = 1.000045379$ and $b_2(1) = 6.0022145$. The calculations for the $c_j^i(1)$ and $\sigma_j^i(1)$ parameters, $i = 1, 2$, $j = 1, 2$, are made in a similar way, but using Equations (5.36) and (5.37), respectively.

Even with only one computation step, we see that the output centers b_i, $i = 1, 2$, are moving to perform an interpolation that is more appropriate for the third data point. To see this, notice that $b_2(1) = 6.0022145$ where $b_2(0) = 5.0$ so that the output center moved much closer to $y^3 = 6$.

To further study how the gradient algorithm works, we recommend that you write a computer program to implement the update formulas for this example. You may need to tune the λ_i and approach to cycling through the data. Then, using an appropriate termination condition (see the discussion above), stop the algorithm and test the quality of the interpolation by placing inputs into the fuzzy system and seeing if the outputs are good interpolated values (e.g., compare them to Figure 5.2 on page 237). In the next section we will provide a more detailed example, but for the training of Takagi-Sugeno fuzzy systems.

5.4.3 Training Takagi-Sugeno Fuzzy Systems

The Takagi-Sugeno fuzzy system that we train in this section takes on the form

$$f(x|\theta(k)) = \frac{\sum_{i=1}^{R} g_i(x,k)\mu_i(x,k)}{\sum_{i=1}^{R} \mu_i(x,k)}$$

where $\mu_i(x,k)$ is defined in Equation (5.34) on page 262 (of course, other definitions are possible), $x = [x_1, x_2, \ldots, x_n]^\top$, and

$$g_i(x,k) = a_{i,0}(k) + a_{i,1}(k)x_1 + a_{i,2}(k)x_2 + \cdots + a_{i,n}(k)x_n$$

(note that we add the index k since we will update the $a_{i,j}$ parameters). For more details on how to define Takagi-Sugeno fuzzy systems, see Section 2.3.7 on page 73.

Parameter Update Formulas

Following the same approach as in the previous section, we need to update the $a_{i,j}$ parameters of the $g_i(x,k)$ functions and c_j^i and σ_j^i. Notice, however, that most of the work is done since if in Equations (5.36) and (5.37) we replace $b_i(k)$ with $g_i(x^m,k)$, we get the update formulas for the c_j^i and σ_j^i for the Takagi-Sugeno fuzzy system.

To update the $a_{i,j}$ we use

$$a_{i,j}(k+1) = a_{i,j}(k) - \lambda_4 \left.\frac{\partial e_m}{\partial a_{i,j}}\right|_k \tag{5.38}$$

when $\lambda_4 > 0$ is the step size. Notice that

$$\frac{\partial e_m}{\partial a_{i,j}} = \epsilon_m(k) \frac{\partial f(x^m|\theta(k))}{\partial g_i(x^m,k)} \frac{\partial g_i(x^m,k)}{\partial a_{i,j}(k)}$$

for all $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$ (plus $j = 0$) and

$$\frac{\partial f(x^m|\theta(k))}{\partial g_i(x^m,k)} = \frac{\mu_i(x^m,k)}{\sum_{i=1}^R \mu_i(x^m,k)}$$

for all $i = 1, 2, \ldots, R$. Also,

$$\frac{\partial g_i(x^m,k)}{\partial a_{i,0}(k)} = 1$$

and

$$\frac{\partial g_i(x,k)}{\partial a_{i,j}(k)} = x_j$$

for all $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, R$.

This gives the update formulas for all the parameters of the Takagi-Sugeno fuzzy system. In the previous section we discussed issues in the choice of the step sizes and initial parameter values, how to cycle through the training data in G, and some convergence issues. All of this discussion is relevant to the training of Takagi-Sugeno models also. The training of more general functional fuzzy systems where the g_i take on more general forms proceeds in a similar manner. In fact, it is easy to develop the update formulas for any functional fuzzy system such that

$$\frac{\partial g_i(x^m,k)}{\partial a_{i,j}(k)}$$

can be determined analytically. Finally, we would note that Takagi-Sugeno or general functional fuzzy systems can be trained either off- or on-line. Chapter 6 discusses how such on-line training can be used in adaptive control.

Example

As an example, consider once again the data set G in Equation (5.3) on page 236. We will train the Takagi-Sugeno fuzzy system with two rules ($R = 2$) and $n = 2$ considered in Equation (5.33). We will cycle through the data set G 40 times (similar to how we did in the RLS example) to get the error between the fuzzy system output and the output portions of the training data to decrease to some small value.

We use Equations (5.38), (5.36), and (5.37) to update the $a_{i,j}(k)$, $c_j^i(k)$, and $\sigma_j^i(k)$ values, respectively, for all $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$, and we choose $\bar{\sigma}$ from the previous section to be 0.01. For initialization we pick $\lambda_4 = 0.01$, $\lambda_2 = \lambda_3 = 1$, $a_{i,j}(0) = 1$, and $\sigma_j^i = 2$ for all i and j, and $c_1^1(0) = 1.5$, $c_2^1(0) = 3$, $c_1^2(0) = 3$, and $c_2^2(0) = 5$. The step sizes were tuned a bit to improve convergence, but could probably be further tuned to improve it more. The $a_{i,j}(0)$ values are simply somewhat arbitrary guesses. The $\sigma_j^i(0)$ values seem like reasonable spreads considering the training data. The $c_j^i(0)$ values are the same ones used in the least squares example and seem like reasonable guesses since they try to spread the premise membership function peaks somewhat uniformly over the input portions of the training data. It is possible that a better initial guess for the $a_{i,j}(0)$ could be obtained by using the least squares method to pick these for the initial guesses for the $c_j^i(0)$ and $\sigma_j^i(0)$; in some ways this would make the guess for the $a_{i,j}(0)$ more consistent with the other initial parameters.

By the time the algorithm terminates, the error between the fuzzy system output and the output portions of the training data has reduced to less than 0.125 but is still showing a decreasing oscillatory behavior. At algorithm termination ($k = 119$), the consequent parameters are

$$a_{1,0}(119) = 0.8740,\ a_{1,1}(119) = 0.9998,\ a_{1,2}(119) = 0.7309$$

$$a_{2,0}(119) = 0.7642,\ a_{2,1}(119) = 0.3426,\ a_{2,2}(119) = 0.7642$$

the input membership function centers are

$$c_1^1(119) = 2.1982,\ c_1^2(119) = 2.6379$$

$$c_2^1(119) = 4.2833,\ c_2^2(119) = 4.7439$$

and their spreads are

$$\sigma_1^1(119) = 0.7654,\ \sigma_1^2(119) = 2.6423$$

$$\sigma_2^1(119) = 1.2713,\ \sigma_2^2(119) = 2.6636$$

These parameters, which collectively we call θ, specify the final Takagi-Sugeno fuzzy system.

To test the Takagi-Sugeno fuzzy system, we use the training data and some other cases. For the training data points we find

$$\begin{aligned} f(x^1|\theta) &= 1.4573 \\ f(x^2|\theta) &= 4.8463 \\ f(x^3|\theta) &= 6.0306 \end{aligned}$$

so that the trained fuzzy system maps the training data reasonably accurately. Next, we test the fuzzy system at some points not in the training data set to see how it interpolates. In particular, we find

$$\begin{aligned} f([1,2]^\top|\theta) &= 2.4339 \\ f([2.5,5]^\top|\theta) &= 5.7117 \\ f([4,7]^\top|\theta) &= 6.6997 \end{aligned}$$

These values seem like good interpolated values considering Figure 5.2 on page 237, which illustrates the data set G for this example.

5.4.4 Momentum Term and Step Size

There is some evidence that convergence properties of the gradient method can sometimes be improved via the addition of a "momentum term" to each of the update laws in Equations (5.35), (5.36), and (5.37). For instance, we could modify Equation (5.35) to

$$b_i(k+1) = b_i(k) - \lambda_1 \left.\frac{\partial e_m}{\partial b_i}\right|_k + \beta_i(b_i(k) - b_i(k-1))$$

$i = 1, 2, \ldots, R$ where β_i is the gain on the momentum term. Similar changes can be made to Equations (5.36) and (5.37). Generally, the momentum term will help to keep the updates moving in the right direction. It is a method that has found wide use in the training of neural networks.

While for some applications a fixed step size λ_i can be sufficient, there has been some work done on adaptively picking the step size. For example, if errors are decreasing rapidly, take big update steps, but if errors are decreasing slowly take small steps. Another option is to try to adaptively pick the λ_i step sizes so that they best minimize the error

$$e_m = \frac{1}{2}[f(x^m|\theta(k)) - y^m]^2$$

For instance, for Equation (5.35) you could pick at time k the step size to be λ_1^*

such that

$$\frac{1}{2}\left[f\left(x^m|\left\{\theta(k):b_i(k)-\lambda_1^*\left.\frac{\partial e_m}{\partial b_i}\right|_k\right\}\right)-y^m\right]^2=$$
$$\min_{\lambda_1\in[0,\bar{\lambda}_1]}\frac{1}{2}\left[f\left(x^m|\left\{\theta(k):b_i(k)-\lambda_1\left.\frac{\partial e_m}{\partial b_i}\right|_k\right\}\right)-y^m\right]^2$$

(where $\bar{\lambda}_1 > 0$ is some scalar that is fixed a priori) so that the step size will optimize the reduction of the error. Similar changes could be made to Equations (5.36) and (5.37). A vector version of the statement of how to pick the optimal step size is given by constraining all the components of $\theta(k)$, not just the output centers as we do above. The problem with this approach is that it adds complexity to the update formulas since at each step an optimization problem must be solved to find the step size.

5.4.5 Newton and Gauss-Newton Methods

There are many gradient-type optimization techniques that can be used to pick θ to minimize e_m. For instance, you could use Newton, quasi-Newton, Gauss-Newton, or Levenberg-Marquardt methods. Each of these has certain advantages and disadvantages and many deserve consideration for a particular application.

In this section we will develop vector rather than scalar parameter update laws so we define $\theta(k) = [\theta_1(k), \theta_2(k), \ldots, \theta_p(k)]^\top$ to be a $p \times 1$ vector. Also, we provide this development for n input, $\bar{N}$ output fuzzy systems so that $f(x^m|\theta(k))$ and y^m are both $\bar{N} \times 1$ vectors.

The basic form of the update using a gradient method to minimize the function

$$e_m(k|\theta(k)) = \frac{1}{2}|f(x^m|\theta(k)) - y^m|^2$$

(notice that we explicitly add the dependence of $e_m(k)$ on $\theta(k)$ by using this notation) via the choice of $\theta(k)$ is

$$\theta(k+1) = \theta(k) + \lambda_k d(k) \tag{5.39}$$

where $d(k)$ is the $p \times 1$ descent direction, and λ_k is a (scalar) positive step size that can depend on time k (not to be confused with the earlier notation for the step sizes). Here, $|x|^2 = x^\top x$. For the descent function

$$\left(\frac{\partial e_m(k|\theta(k))}{\partial\theta(k)}\right)^\top d(k) < 0$$

and if

$$\frac{\partial e_m(k|\theta(k))}{\partial\theta(k)} = 0$$

where "0" is a $p \times 1$ vector of zeros, the method does not update $\theta(k)$. Our update formulas for the fuzzy system in Equations (5.35), (5.36), and (5.37) use

$$d(k) = -\frac{\partial e_m(k|\theta(k))}{\partial \theta(k)} = -\nabla e_m(k|\theta(k))$$

(which is the gradient of e_m with respect to $\theta(k)$) so they actually provide for a "steepest descent" approach (of course, Equations (5.35), (5.36), and (5.37) are scalar update laws each with its own step size, while Equation (5.39) is a vector update law with a single step size). Unfortunately, this method can sometimes converge slowly, especially if it gets on a long, low slope surface.

Next, let

$$\nabla^2 e_m(k|\theta(k)) = \left[\frac{\partial^2 e_m(k|\theta(k))}{\partial \theta_i(k)\theta_j(k)}\right]$$

be the $p \times p$ "Hessian matrix," the elements of which are the second partials of $e_m(k|\theta(k))$ at $\theta(k)$. In "Newton's method" we choose

$$d(k) = -\left(\nabla^2 e_m(k|\theta(k))\right)^{-1} \nabla e_m(k|\theta(k)) \tag{5.40}$$

provided that $\nabla^2 e_m(k|\theta(k))$ is positive definite so that it is invertible (see Section 4.3.5 for a definition of "positive definite"). For a function $e_m(k|\theta(k))$ that is quadratic in $\theta(k)$, Newton's method provides convergence in one step; for some other functions, it can converge very fast. The price you pay for this convergence speed is computation of Equation (5.40) and the need to verify the existence of the inverse in that equation.

In "quasi-Newton methods" you try to avoid problems with existence and computation of the inverse in Equation (5.40) by choosing

$$d(k) = -\Lambda(k)\nabla e_m(k|\theta(k))$$

where $\Lambda(k)$ is a positive definite $p \times p$ matrix for all $k \geq 0$ and is sometimes chosen to approximate $\left(\nabla^2 e_m(k|\theta(k))\right)^{-1}$ (e.g., in some cases by using only the diagonal elements of $\left(\nabla^2 e_m(k|\theta(k))\right)^{-1}$). If $\Lambda(k)$ is chosen properly, for some applications much of the convergence speed of Newton's method can be achieved.

Next, consider the Gauss-Newton method that is used to solve a least squares problem such as finding $\theta(k)$ to minimize

$$e_m(k|\theta(k)) = \frac{1}{2}|f(x^m|\theta(k)) - y^m|^2 = \frac{1}{2}|\epsilon_m(k|\theta(k))|^2$$

where

$$\epsilon_m(k|\theta(k)) = f(x^m|\theta(k)) - y^m = [\epsilon_{m_1}, \epsilon_{m_2}, \ldots, \epsilon_{m_{\tilde{N}}}]^\top$$

First, linearize $\epsilon_m(k|\theta(k))$ around $\theta(k)$ (i.e., use a truncated Taylor series expansion) to get

$$\tilde{\epsilon}_m(\theta|\theta(k)) = \epsilon_m(k|\theta(k)) + \nabla\epsilon_m(k|\theta(k))^\top(\theta - \theta(k))$$

Here,

$$\nabla\epsilon_m(k|\theta(k)) = \left[\nabla\epsilon_{m_1}(k|\theta(k)), \nabla\epsilon_{m_2}(k|\theta(k)), \ldots, \nabla\epsilon_{m_{\bar{N}}}(k|\theta(k))\right]$$

is a $p \times \bar{N}$ matrix whose columns are gradient vectors

$$\nabla\epsilon_{m_i}(k|\theta(k)) = \frac{\partial\nabla\epsilon_{m_i}(k|\theta(k))}{\partial\theta(k)}$$

$i = 1, 2, \ldots, \bar{N}$. Notice that

$$\nabla\epsilon_m(k|\theta(k))^\top$$

is the "Jacobian." Also note that the notation $\tilde{\epsilon}_m(\theta|\theta(k))$ is used to emphasize the dependence on both $\theta(k)$ and θ.

Next, minimize the norm of the linearized function $\tilde{\epsilon}_m(\theta|\theta(k))$ by letting

$$\theta(k+1) = \arg\min_\theta \frac{1}{2}|\tilde{\epsilon}_m(\theta|\theta(k))|^2$$

Hence, in the Gauss-Newton approach we update $\theta(k)$ to a value that will best minimize a linear approximation to $\epsilon_m(k|\theta(k))$. Notice that

$$\begin{aligned}\theta(k+1) = \arg\min_\theta \frac{1}{2}\big[&|\epsilon_m(k|\theta(k))|^2 + 2(\theta - \theta(k))^\top\left(\nabla\epsilon_m(k|\theta(k))\right)\epsilon_m(k|\theta(k)) \\ &+ (\theta - \theta(k))^\top\nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top(\theta - \theta(k))\big] \\ = \arg\min_\theta \frac{1}{2}\big[&|\epsilon_m(k|\theta(k))|^2 + 2(\theta - \theta(k))^\top\left(\nabla\epsilon_m(k|\theta(k))\right)\epsilon_m(k|\theta(k)) \\ &+ \theta^\top\nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top\theta - 2\theta(k)^\top\nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top\theta \\ &+ \theta(k)^\top\nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top\theta(k)\big] \end{aligned} \tag{5.41}$$

To perform this minimization, notice that we have a quadratic function so we find

$$\begin{aligned}\frac{\partial[\cdot]}{\partial\theta} = &\nabla\epsilon_m(k|\theta(k))\epsilon_m(k|\theta(k)) + \nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top\theta \\ &- \nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top\theta(k)\end{aligned} \tag{5.42}$$

where $[\cdot]$ denotes the expression in Equation (5.41) in brackets multiplied by one half. Setting this equal to zero, we get the minimum achieved at θ^* where

$$\nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top(\theta^* - \theta(k)) = -\nabla\epsilon_m(k|\theta(k))^\top\epsilon_m(k|\theta(k))$$

or, if $\nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top$ is invertible,

$$\theta^* - \theta(k) = -\left(\nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top\right)^{-1}\nabla\epsilon_m(k|\theta(k))\epsilon_m(k|\theta(k))$$

Hence, the Gauss-Newton update formula is

$$\theta(k+1) = \theta(k) - \left(\nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top\right)^{-1}\nabla\epsilon_m(k|\theta(k))\epsilon_m(k|\theta(k))$$

To avoid problems with computing the inverse, the method is often implemented as

$$\theta(k+1) = \theta(k) - \lambda_k\left(\nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top + \Gamma(k)\right)^{-1}\nabla\epsilon_m(k|\theta(k))\epsilon_m(k|\theta(k))$$

where λ_k is a positive step size that can change at each time k, and $\Gamma(k)$ is a $p \times p$ diagonal matrix such that

$$\nabla\epsilon_m(k|\theta(k))\nabla\epsilon_m(k|\theta(k))^\top + \Gamma(k)$$

is positive definite so that it is invertible. In the Levenberg-Marquardt method you choose $\Gamma(k) = \alpha I$ where $\alpha > 0$ and I is the $p \times p$ identity matrix. Essentially, a Gauss-Newton iteration is an approximation to a Newton iteration so it can provide for faster convergence than, for instance, steepest descent, but not as fast as a pure Newton method; however, computations are simplified. Note, however, that for each iteration of the Gauss-Newton method (as it is stated above) we must find the inverse of a $p \times p$ matrix; there are, however, methods in the optimization literature for coping with this issue.

Using each of the above methods to train a fuzzy system is relatively straightforward. For instance, notice that many of the appropriate partial derivatives have already been found when we developed the steepest descent approach to training.

5.5 Clustering Methods

"Clustering" is the partitioning of data into subsets or groups based on similarities between the data. Here, we will introduce two methods to perform fuzzy clustering where we seek to use fuzzy sets to define soft boundaries to separate data into groups. The methods here are related to conventional ones that have been developed in the field of pattern recognition. We begin with a fuzzy "c-means" technique coupled with least squares to train Takagi-Sugeno fuzzy systems, then we briefly study a nearest neighborhood method for training standard fuzzy systems. In the c-means approach, we continue in the spirit of the previous methods in that we use optimization to pick the clusters and, hence, the premise membership function parameters. The consequent parameters are chosen using the weighted least squares approach developed earlier. The nearest neighborhood approach also uses a type of optimization in the construction of cluster centers and, hence, the fuzzy system. In the next section we break away from the optimization approaches to fuzzy system

construction and study simple constructive methods that are called "learning by examples."

5.5.1 Clustering with Optimal Output Predefuzzification

In this section we will introduce the clustering with optimal output predefuzzification approach to train Takagi-Sugeno fuzzy systems. We do this via the simple example we have used in previous sections.

Clustering for Specifying Rule Premises

Fuzzy clustering is the partitioning of a collection of data into fuzzy subsets or "clusters" based on similarities between the data and can be implemented using an algorithm called fuzzy c-means. Fuzzy c-means is an iterative algorithm used to find grades of membership μ_{ij} (scalars) and cluster centers $\underline{v}^j$ (vectors of dimension $n \times 1$) to minimize the objective function

$$J = \sum_{i=1}^{M} \sum_{j=1}^{R} (\mu_{ij})^m |x^i - \underline{v}^j|^2 \tag{5.43}$$

where $m > 1$ is a design parameter. Here, M is the number of input-output data pairs in the training data set G, R is the number of clusters (number of rules) we wish to calculate, x^i for $i = 1, ..., M$ is the input portion of the input-output training data pairs, $\underline{v}^j = [v_1^j, v_2^j, \ldots, v_n^j]^\top$ for $j = 1, ..., R$ are the cluster centers, and μ_{ij} for $i = 1, ..., M$ and $j = 1, ..., R$ is the grade of membership of x^i in the j^{th} cluster. Also, $|x| = \sqrt{x^\top x}$ where x is a vector. Intuitively, minimization of J results in cluster centers being placed to represent groups (clusters) of data.

Fuzzy clustering will be used to form the premise portion of the If-Then rules in the fuzzy system we wish to construct. The process of "optimal output predefuzzification" (least squares training for consequent parameters) is used to form the consequent portion of the rules. We will combine fuzzy clustering and optimal output predefuzzification to construct multi-input single-output fuzzy systems. Extension of our discussion to multi-input multi-output systems can be done by repeating the process for each of the outputs.

In this section we utilize a Takagi-Sugeno fuzzy system in which the consequent portion of the rule-base is a function of the crisp inputs such that

$$\textbf{If } H^j \textbf{ Then } g_j(x) = a_{j,0} + a_{j,1}x_1 + \cdots + a_{j,n}x_n \tag{5.44}$$

where n is the number of inputs and H^j is an input fuzzy set given by

$$H^j = \{(x, \mu_{H^j}(x)) : x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_n\} \tag{5.45}$$

where $\mathcal{X}_i$ is the i^{th} universe of discourse, and $\mu_{H^j}(x)$ is the membership function associated with H^j that represents the premise certainty for rule j; and $g_j(x) = \underline{a}_j^\top \hat{x}$ where $\underline{a}_j = [a_{j,0}, a_{j,1} \ldots, a_{j,n}]^\top$ and $\hat{x} = [1, x^\top]^\top$ where $j = 1, \ldots, R$. The resulting

fuzzy system is a weighted average of the output $g_j(x)$ for $j = 1, ..., R$ and is given by

$$f(x|\theta) = \frac{\sum_{j=1}^{R} g_j(x)\mu_{H^j}(x)}{\sum_{j=1}^{R} \mu_{H^j}(x)} \tag{5.46}$$

where R is the number of rules in the rule-base. Next, we will use the Takagi-Sugeno fuzzy model, fuzzy clustering, and optimal output defuzzification to determine the parameters $\underline{a}_j$ and $\mu_{H^j}(x)$, which define the fuzzy system. We will do this via a simple example.

Suppose we use the example data set in Equation (5.3) on page 236 that has been used in the previous sections. We first specify a "fuzziness factor" $m > 1$, which is a parameter that determines the amount of overlap of the clusters. If $m > 1$ is large, then points with less membership in the j^{th} cluster have less influence on the determination of the new cluster centers. Next, we specify the number of clusters R we wish to calculate. The number of clusters R equals the number of rules in the rule-base and must be less than or equal to the number of data pairs in the training data set G (i.e., $R \leq M$). We also specify the error tolerance $\epsilon_c > 0$, which is the amount of error allowed in calculating the cluster centers. We initialize the cluster centers $\underline{v}_0^j$ via a random number generator so that each component of $\underline{v}_0^j$ is no larger (smaller) than the largest (smallest) corresponding component of the input portion of the training data. The selection of $\underline{v}_0^j$, although somewhat arbitrary, may affect the final solution.

For our simple example, we choose $m = 2$ and $R = 2$, and let $\epsilon_c = 0.001$. Our initial cluster centers were randomly chosen to be

$$\underline{v}_0^1 = \begin{bmatrix} 1.89 \\ 3.76 \end{bmatrix}$$

and

$$\underline{v}_0^2 = \begin{bmatrix} 2.47 \\ 4.76 \end{bmatrix}$$

so that each component lies in between x_1^i and x_2^i for $i = 1, 2, 3$ (see the definition of G in Equation (5.3)).

Next, we compute the new cluster centers $\underline{v}^j$ based on the previous cluster centers so that the objective function in Equation (5.43) is minimized. The necessary conditions for minimizing J are given by

$$\underline{v}_{\text{new}}^j = \frac{\sum_{i=1}^{M} x^i(\mu_{ij}^{\text{new}})^m}{\sum_{i=1}^{M}(\mu_{ij}^{\text{new}})^m} \tag{5.47}$$

where

$$\mu_{ij}^{\text{new}} = \left[\sum_{k=1}^{R} \left(\frac{|x^i - \underline{v}_{\text{old}}^j|^2}{|x^i - \underline{v}_{\text{old}}^k|^2} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{5.48}$$

for each $i = 1, \ldots, M$ and for each $j = 1, 2, \ldots, R$ such that $\sum_{j=1}^{R} \mu_{ij}^{\text{new}} = 1$ (and $|x|^2 = x^\top x$). In Equation (5.48) we see that it is possible that there exists an $i = 1, 2, \ldots, M$ such that $|x^i - \underline{v}_{\text{old}}^j|^2 = 0$ for some $j = 1, 2, \ldots, R$. In this case the μ_{ij}^{new} is undefined. To fix this problem, let μ_{ij} for all i be any nonnegative numbers such that $\sum_{j=1}^{R} \mu_{ij} = 1$ and $\mu_{ij} = 0$, if $|x^i - \underline{v}_{\text{old}}^j|^2 \neq 0$.

Using Equation (5.48) for our example with $\underline{v}_{\text{old}}^j = \underline{v}_0^j$, $j = 1, 2$, we find that $\mu_{11}^{\text{new}} = 0.6729, \mu_{12}^{\text{new}} = 0.3271, \mu_{21}^{\text{new}} = 0.9197, \mu_{22}^{\text{new}} = 0.0803, \mu_{31}^{\text{new}} = 0.2254$, and $\mu_{32}^{\text{new}} = 0.7746$. We use these μ_{ij}^{new} from Equation (5.48) to calculate the new cluster centers

$$\underline{v}_{\text{new}}^1 = \begin{bmatrix} 1.366 \\ 3.4043 \end{bmatrix}$$

and

$$\underline{v}_{\text{new}}^2 = \begin{bmatrix} 2.5410 \\ 5.3820 \end{bmatrix}$$

using Equation (5.47).

Next, we compare the distances between the current cluster centers $\underline{v}_{\text{new}}^j$ and the previous cluster centers $\underline{v}_{\text{old}}^j$ (which for the first step is $\underline{v}_0^j$). If $|\underline{v}_{\text{new}}^j - \underline{v}_{\text{old}}^j| < \epsilon_c$ for all $j = 1, 2, \ldots, R$ then the cluster centers $\underline{v}_{\text{new}}^j$ accurately represent the input data, the fuzzy clustering algorithm is terminated, and we proceed on to the optimal output defuzzification algorithm (see below). Otherwise, we continue to iteratively use Equations (5.47) and (5.48) until we find cluster centers $\underline{v}_{\text{new}}^j$ that satisfy $|\underline{v}_{\text{new}}^j - \underline{v}_{\text{old}}^j| < \epsilon_c$ for all $j = 1, 2, \ldots, R$. For our example, $\underline{v}_{\text{old}}^j = \underline{v}_0^j$, and we see that $|\underline{v}_{\text{new}}^j - \underline{v}_{\text{old}}^j| = 0.6328$ for $j = 1$ and 0.6260 for $j = 2$. Both of these values are greater than ϵ_c, so we continue to update the cluster centers.

Proceeding to the next iteration, let $\underline{v}_{\text{old}}^j = \underline{v}_{\text{new}}^j$, $j = 1, 2, \ldots, R$ from the last iteration, and apply Equations (5.47) and (5.48) to find $\mu_{11}^{\text{new}} = 0.8233, \mu_{12}^{\text{new}} = 0.1767, \mu_{21}^{\text{new}} = 0.7445, \mu_{22}^{\text{new}} = 0.2555, \mu_{31}^{\text{new}} = 0.0593$, and $\mu_{32}^{\text{new}} = 0.9407$ using the cluster centers calculated above, yielding the new cluster centers

$$\underline{v}_{\text{new}}^1 = \begin{bmatrix} 0.9056 \\ 2.9084 \end{bmatrix}$$

and

$$\underline{v}^2_{\text{new}} = \begin{bmatrix} 2.8381 \\ 5.7397 \end{bmatrix}$$

Computing the distances between these cluster centers and the previous ones, we find that $|\underline{v}^j_{\text{new}} - \underline{v}^j_{\text{old}}| > \epsilon_c$, so the algorithm continues. It takes 14 iterations before the algorithm terminates (i.e., before we have $|\underline{v}^j_{\text{new}} - \underline{v}^j_{\text{old}}| \leq \epsilon_c = 0.001$ for all $j = 1, 2, \ldots, R$). When it does terminate, name the final membership grade values μ_{ij} and cluster centers $\underline{v}^j$, $i = 1, 2, \ldots, M$, $j = 1, 2, \ldots, R$.

For our example, after 14 iterations the algorithm finds $\mu_{11} = 0.9994$, $\mu_{12} = 0.0006$, $\mu_{21} = 0.1875$, $\mu_{22} = 0.8125$, $\mu_{31} = 0.0345$, $\mu_{32} = 0.9655$,

$$\underline{v}^1 = \begin{bmatrix} 0.0714 \\ 2.0725 \end{bmatrix}$$

and

$$\underline{v}^2 = \begin{bmatrix} 2.5854 \\ 5.1707 \end{bmatrix}$$

Notice that the clusters have converged so that $\underline{v}^1$ is near $x^1 = [0, 2]^\top$ and $\underline{v}^2$ lies in between $x^2 = [2, 4]^\top$ and $x^3 = [3, 6]^\top$.

The final values of $\underline{v}^j$, $j = 1, 2, \ldots, R$, are used to specify the premise membership functions for the i^{th} rule. In particular, we specify the premise membership functions as

$$\mu_{H^j}(x) = \left[\sum_{k=1}^{R} \left(\frac{|x - \underline{v}^j|^2}{|x - \underline{v}^k|^2} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{5.49}$$

$j = 1, 2, \ldots, R$ where $\underline{v}^j$, $j = 1, 2, \ldots, R$ are the cluster centers from the last iteration that uses Equations (5.47) and (5.48). It is interesting to note that for large values of m we get smoother (less distinctive) membership functions. This is the primary guideline to use in selecting the value of m; however, often a good first choice is $m = 2$. Next, note that $\mu_{H^j}(x)$ is a premise membership function that is different from any that we have considered. It is used to ensure certain convergence properties of the iterative fuzzy c-means algorithm described above. With the premises of the rules defined, we next specify the consequent portion.

Least Squares for Specifying Rule Consequents

We apply "optimal output predefuzzification" to the training data to calculate the function $g_j(x) = \underline{a}_j^\top \hat{x}$, $j = 1, 2, \ldots, R$ for each rule (i.e., each cluster center), by determining the parameters $\underline{a}_j$. There are two methods you can use to find the $\underline{a}_j$.

Approach 1: For each cluster center $\underline{v}^j$, we wish to minimize the squared error between the function $g_j(x)$ and the output portion of the training data pairs. Let $\hat{x}^i = [1, (x^i)^\top]^\top$ where $(x^i, y^i) \in G$. We wish to minimize the cost function J_j given by

$$J_j = \sum_{i=1}^{M} (\mu_{ij})^2 \left(y^i - (\hat{x}^i)^\top \underline{a}_j\right)^2 \tag{5.50}$$

for each $j = 1, 2, \ldots, R$ where μ_{ij} is the grade of membership of the input portion of the i^{th} data pair for the j^{th} cluster that resulted from the clustering algorithm after it converged, y^i is the output portion of the i^{th} data pair $d^{(i)} = (x^i, y^i)$, and the multiplication of $(\hat{x}^i)^\top$ and $\underline{a}_j$ defines the output associated with the j^{th} rule for the i^{th} training data point.

Looking at Equation (5.50), we see that the minimization of J_j via the choice of the $\underline{a}_j$ is a weighted least squares problem. From Section 5.3 and Equation (5.15) on page 250, the solution $\underline{a}_j$ for $j = 1, 2, \ldots, R$ to the weighted least squares problem is given by

$$\underline{a}_j = (\hat{X}^\top D_j^2 \hat{X})^{-1} \hat{X}^\top D_j^2 Y \tag{5.51}$$

where

$$\begin{aligned} \hat{X} &= \begin{bmatrix} 1 & \ldots & 1 \\ x^1 & \ldots & x^M \end{bmatrix}^\top \\ Y &= [y^1, \ldots, y^M]^\top, \\ D_j^2 &= (\mathrm{diag}([\mu_{1j}, \ldots, \mu_{Mj}]))^2 \end{aligned}$$

For our example the parameters that satisfy the linear function $g_j(x) = \underline{a}_j^\top \underline{\hat{x}}^i$ for $j = 1, 2$ such that J_j in Equation (5.50) is minimized were found to be $\underline{a}_1 = [3, 2.999, -1]^\top$ and $\underline{a}_2 = [3, 3, -1]^\top$, which are very close to each other.

Approach 2: As an alternative approach, rather than solving R least squares problems, one for each rule, we can use the least squares methods discussed in Section 5.3 to specify the consequent parameters of the Takagi-Sugeno fuzzy system. To do this, we simply parameterize the Takagi-Sugeno fuzzy system in Equation (5.46) in a form so that it is linear in the consequent parameters and of the form

$$f(x|\theta) = \theta^\top \xi(x)$$

where θ holds all the $a_{i,j}$ parameters and ξ is specified in a similar manner to how we did in Section 5.3.3. Now, just as we did in Section 5.3.3, we can use batch or recursive least squares methods to find θ. Unless we indicate otherwise, we will always use approach 1 in this book.

Testing the Approximator

Suppose that we use approach 1 to specify the rule consequents. To test how accurately the constructed fuzzy system represents the training data set G in Figure 5.2 on page 237, suppose that we choose the test point x' such that $(x', y') \notin G$. Specifically, we choose

$$x' = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

We would expect from Figure 5.2 on page 237 that the output of the fuzzy system would lie somewhere between 1 and 5. The output is 3.9999, so we see that the trained Takagi-Sugeno fuzzy system seems to interpolate adequately. Notice also that if we let $x = x^i$, $i = 1, 2, 3$ where $(x^i, y^i) \in G$, we get values very close to the y^i, $i = 1, 2, 3$, respectively. That is, for this example the fuzzy system nearly perfectly maps the training data pairs. We also note that if the input to the fuzzy system is $x = [2.5, 5]^\top$, the output is 5.5, so the fuzzy system seems to perform good interpolation near the training data points.

Finally, we note that the $\underline{a}_j$ will clearly not always be as close to each other as for this example. For instance, if we add the data pair $([4, 5]^\top, 5.5)$ to G (i.e., make $M = 4$), then the cluster centers converge after 13 iterations (using the same parameters m and ϵ_c as we did earlier). Using approach 1 to find the consequent parameters, we get

$$\underline{a}_1 = [-1.458, 0.7307, 1.2307]^\top$$

and

$$\underline{a}_2 = [2.999, 0.00004, 0.5]^\top$$

For the resulting fuzzy system, if we let $x = [1, 2]^\top$ in Equation (5.46), we get an output value of 1.8378, so we see that it performs differently than the case for $M = 3$, but that it does provide a reasonable interpolated value.

5.5.2 Nearest Neighborhood Clustering

As with the other approaches, we want to construct a fuzzy estimation system that approximates the function g that is inherently represented in the training data set G. We use singleton fuzzification, Gaussian membership functions, product inference, and center-average defuzzification, and the fuzzy system that we train is given by

$$f(x|\theta) = \frac{\sum_{i=1}^{R} A_i \prod_{j=1}^{n} \exp\left(-\left(\frac{x_j - v_j^i}{2\sigma}\right)^2\right)}{\sum_{i=1}^{R} B_i \prod_{j=1}^{n} \exp\left(-\left(\frac{x_j - v_j^i}{2\sigma}\right)^2\right)} \tag{5.52}$$

where R is the number of clusters (rules), n is the number of inputs,

$$\underline{v}^j = [v_1^j, v_2^j, \ldots, v_n^j]^\top$$

are the cluster centers, σ is a constant and is the width of the membership functions, and A_i and B_i are the parameters whose values will be specified below (to train a multi-output fuzzy system, simply apply the procedure to the fuzzy system that generates each output). From Equation (5.52), we see that the parameter vector θ is given by

$$\theta = [A_1, \ldots, A_R, B_1, \ldots, B_R, v_1^1, \ldots, v_n^1, ..., v_1^R, \ldots, v_n^R, \sigma]^\top$$

and is characterized by the number of clusters (rules) R and the number of inputs n. Next, we will explain, via a simple example, how to use the nearest neighborhood clustering technique to construct a fuzzy system by choosing the parameter vector θ.

Suppose that $n = 2$, $X \subset \Re^2$, and $Y \subset \Re$, and that we use the training data set G in Equation (5.3) on page 236. We first specify the parameter σ, which defines the width of the membership functions. A small σ provides narrow membership functions that may yield a less smooth fuzzy system mapping, which may cause fuzzy system mapping not to generalize well for the data points not in the training set. Increasing the parameter σ will result in a smoother fuzzy system mapping. Next, we specify the quantity ϵ_f, which characterizes the maximum distance allowed between each of the cluster centers. The smaller ϵ_f, the more accurate are the clusters that represent the function g. For our example, we chose $\sigma = 0.3$ and $\epsilon_f = 3.0$. We must also define an initial fuzzy system by initializing the parameters A_1, B_1, and $\underline{v}^1$. Specifically, we set $A_1 = y^1$, $B_1 = 1$, and $v_j^1 = x_j^1$ for $j = 1, 2, \ldots, n$. If we take our first data pair,

$$(x^1, y^1) = \left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, 1 \right)$$

we get $A_1 = 1$, $B_1 = 1$, and

$$\underline{v}^1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

which forms our first cluster (rule) for $f(x|\theta)$. Next, we take the second data pair,

$$(x^2, y^2) = \left(\begin{bmatrix} 2 \\ 4 \end{bmatrix}, 5 \right)$$

and compute the distance between the input portion of the data pair and each of the R existing cluster centers, and let the smallest distance be $|x^i - \underline{v}^l|$ (i.e., the nearest cluster to x^i is $\underline{v}^l$) where $|x| = \sqrt{x^\top x}$. If $|x^i - \underline{v}^l| < \epsilon_f$, then we do not add any clusters (rules) to the existing system, but we update the existing parameters

A_l and B_l for the nearest cluster $\underline{v}^l$ to account for the output portion y^i of the current input-output data pair (x^i, y^i) in the training data set G. Specifically, we let

$$A_l := A_l^{\text{old}} + y^i$$

and

$$B_l := B_l^{\text{old}} + 1$$

These values are incremented to represent adding the effects of another data pair to the existing cluster. For instance, A_l is incremented so that the sum in the numerator of Equation (5.52) is modified to include the effects of the additional data pair without adding another rule. The value of B_l is then incremented to represent that we have added the effects of another data pair (it normalizes the sum in Equation (5.52)). Note that we do not modify the cluster centers in this case, just the A_l and B_l values; hence we do not modify the premises (that are parameterized via the cluster centers and σ), just the consequents of the existing rule that the new data pair is closest to.

Suppose that $|x^i - \underline{v}^l| > \epsilon_f$. Then we add an additional cluster (rule) to represent the (x^2, y^2) information about the function g by modifying the parameter vector θ and letting $R = 2$ (i.e., we increase the number of clusters (rules)), $v_j^R = x_j^2$ for $j = 1, 2, \ldots, n$, $A_R = y^2$, and $B_R = 1$. These assignments of variables represent the explicit addition of a rule to the fuzzy system. Hence, for our example

$$\underline{v}^2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, A_2 = 5, B_2 = 1$$

The nearest neighbor clustering technique is implemented by repeating the above algorithm until all of the M data pairs in G are used.

Consider the third data pair,

$$(x^3, y^3) = \left(\begin{bmatrix} 3 \\ 6 \end{bmatrix}, 6 \right)$$

We would compute the distance between the input portion of the current data pair x^3 and each of the $R = 2$ cluster centers and find the smallest distance $|x^3 - \underline{v}^l|$. For our example, what is the value of $|x^3 - \underline{v}^l|$ and which cluster center is closest? Explain how to update the fuzzy system (specifically, provide values for A_2 and B_2). To test how accurately the fuzzy system f represents the training data set G, suppose that we choose a test point x' such that $(x', y') \notin G$. Specifically, we choose

$$x' = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

We would expect the output value of the fuzzy system for this input to lie somewhere between 1 and 5 (why?).

5.6 Extracting Rules from Data

In this section we discuss two very intuitive approaches to the construction of a fuzzy system f so that it approximates the function g. These approaches involve showing how to directly specify rules that represent the data pairs ("examples"). Our two "learning from examples" approaches depart significantly from the approaches used up to this point that relied on optimization to specify fuzzy system parameters. In our first approach, the training procedure relies on the complete specification of the membership functions and only constructs the rules. The second approach constructs all the membership functions and rules, and for this reason can be considered a bit more general.

5.6.1 Learning from Examples (LFE)

In this section we show how to construct fuzzy systems using the "learning from examples" (LFE) technique. The LFE technique generates a rule-base for a fuzzy system by using numerical data from a physical system and possibly linguistic information from a human expert. We will describe the technique for multi-input single-output (MISO) systems. The technique can easily be extended to apply to MIMO systems by repeating the procedure for each of the outputs. We will use singleton fuzzification, minimum to represent the premise and implication, and COG defuzzification; however, the LFE method does not explicitly depend on these choices. Other choices outlined in Chapter 2 can be used as well.

Membership Function Construction

The membership functions are chosen a priori for each of the input universes of discourse and the output universe of discourse. For a two-input one-output fuzzy system, one typical choice for membership functions is shown in Figure 5.8, where

1. $\mathcal{X}_i = [x_i^-, x_i^+]$, i = 1,2, and $\mathcal{Y} = [y^-, y^+]$ are chosen according to the expected range of variation in the input and output variables.
2. The number of membership functions on each universe of discourse affects the accuracy of the function approximation (with fewer generally resulting in lower accuracy).
3. X_i^j and Y^j denote the fuzzy sets with associated membership functions $\mu_{X_i^j}(x_i)$ and $\mu_{Y^j}(y)$, respectively.

In other cases you may want to choose Gaussian or trapezoidal-shaped membership functions. The choice of these membership functions is somewhat ad hoc for the LFE technique.

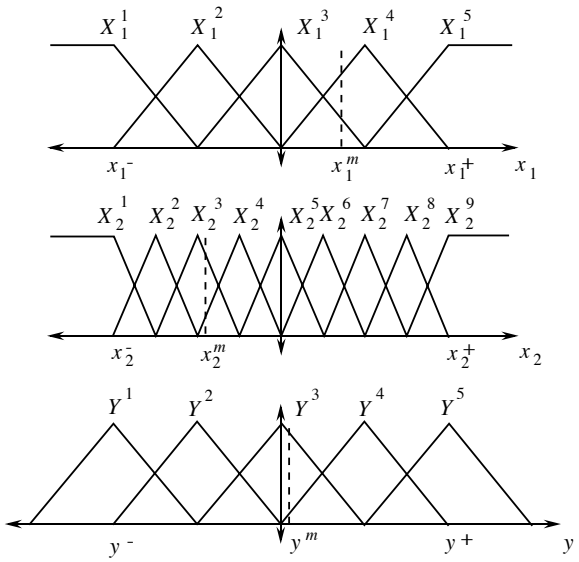


FIGURE 5.8 Example membership functions for input and output universes of discourse for learning from examples technique.

Rule Construction

We finish the construction of the fuzzy system by using the training data in G to form the rules. Generally, the input portions of the training data pairs x^j, where $(x^j, y^j) \in G$, are used to form the premises of rules, while the output portions of the data pairs y^j are used to form the consequents. For our two-input one-output example above, the rule-base to be constructed contains rules of the form

$$\mathcal{R}_i = \textbf{If } x_1 \text{ is } X_1^j \textbf{ and } x_2 \text{ is } X_2^k \textbf{ Then } y \text{ is } Y^l$$

where associated with the i^{th} rule is a "degree" defined by

$$degree(\mathcal{R}_i) = \mu_{X_1^j}(x_1) * \mu_{X_2^k}(x_2) * \mu_{Y^l}(y) \tag{5.53}$$

where "$*$" represents the triangular norm defined in Chapter 2. We will use the standard algebraic product for the definition of the degree of a rule throughout this section so that "$*$" represents the product (of course, you could use, e.g., the minimum operator also). With this, $degree(\mathcal{R}_i)$ quantifies how certain we are that rule $\mathcal{R}_i$ represents some input-output data pair $([x_1^j, x_2^j]^\top, y^j) \in G$ (why?). As an example, suppose that $degree(\mathcal{R}_i) = 1$ for $([x_1^j, x_2^j]^\top, y^j) \in G$. Using the above membership functions, if the input to the fuzzy system is $x = [x_1^j, x_2^j]^\top$ then y^j will be the output of the fuzzy system (i.e., the rule perfectly represents this data pair). If, on the other hand, $x \neq [x_1^j, x_2^j]^\top$, then $degree(\mathcal{R}_i) < 1$ and the mapping induced by rule $\mathcal{R}_i$ does not perfectly match the data pair $([x_1^j, x_2^j]^\top, y^j) \in G$.

The LFE technique is a procedure where we form rules directly from data pairs in G. Assume that several rules have already been constructed from the data pairs in G and that we want to next consider the m^{th} piece of training data $d^{(m)}$. For our example, suppose

$$d^{(m)} = ([x_1^m, x_2^m]^\top, y^m)$$

where example values of x_1^m, x_2^m, and y^m are shown in Figure 5.8. In this case $\mu_{X_1^3}(x_1^m) = 0.3$, $\mu_{X_1^4}(x_1^m) = 0.7$, $\mu_{X_2^3}(x_2^m) = 0.8$, $\mu_{X_2^4}(x_2^m) = 0.2$, $\mu_{Y^3}(y^m) = 0.9$, and $\mu_{Y^4}(y^m) = 0.1$. In the learning from examples approach, you choose input and output membership functions for the rule to be synthesized from $d^{(m)}$ by choosing the ones with the highest degree of membership (resolve ties arbitrarily). For our example, from Figure 5.8 we would *consider* adding the rule

$$\mathcal{R}_m = \textbf{If } x_1 \text{ is } X_1^4 \textbf{ and } x_2 \text{ is } X_2^3 \textbf{ Then } y \text{ is } Y^3$$

to the rule-base since $\mu_{X_1^4}(x_1^m) > \mu_{X_1^3}(x_1^m)$, $\mu_{X_2^3}(x_2^m) > \mu_{X_2^4}(x_2^m)$, and $\mu_{Y^3}(y^m) > \mu_{Y^4}(y^m)$ (i.e., it has a form that appears to best fit the data pair $d^{(m)}$).

Notice that we have $degree(\mathcal{R}_m) = (0.7)(0.8)(0.9) = 0.504$ if $x_1 = x_1^m$, $x_2 = x_2^m$, and $y = y^m$. We use the following guidelines for adding new rules:

- If $degree(\mathcal{R}_m) > degree(\mathcal{R}_i)$, for all $i \neq m$ such that rules $\mathcal{R}_i$ are already in the rule-base (and $degree(\mathcal{R}_i)$ is evaluated for $d^{(m)}$) and the premises for $\mathcal{R}_i$ and $\mathcal{R}_m$ are the same, then the rule $\mathcal{R}_m$ (the rule with the highest degree) would replace rule $\mathcal{R}_i$ in the existing rule-base.
- If $degree(\mathcal{R}_m) \leq degree(\mathcal{R}_i)$ for some i, $i \neq m$, and the premises for $\mathcal{R}_i$ and $\mathcal{R}_m$ are the same, then rule $\mathcal{R}_m$ is not added to the rule-base since the data pair $d^{(m)}$ is already adequately represented with rules in the fuzzy system.
- If rule $\mathcal{R}_m$ does not have the same premise as any other rule already in the rule-base, then it is added to the rule-base to represent $d^{(m)}$.

This process repeats by considering each data pair $i = 1, 2, 3, \ldots, M$. Once you have considered each data pair in G, the process is completed.

Hence, we add rules to represent data pairs. We associate the left-hand side of the rules with the x^i portion of the training data pairs and the consequents with the y^i, $(x^i, y^i) \in G$. We only add rules to represent a data pair if there is not already a rule in the rule-base that represents the data pair better than the one we are considering adding. We are assured that there will be a bounded number of rules added since for a fixed number of inputs and membership functions we know that there are a limited number of possible rules that can be formed (and there is only a finite amount of data). Notice that the LFE procedure constructs rules but does not modify membership functions to help fit the data. The membership functions must be specified a priori by the designer.

Example

As an example, consider the formation of a fuzzy system to approximate the data set G in Equation (5.3) on page 236, which is shown in Figure 5.2. Suppose that we use the membership functions pictured in Figure 5.8 with $x_1^- = 0$, $x_1^+ = 4$, $x_2^- = 0$, $x_2^+ = 8$, $y^- = 0$, and $y^+ = 8$ as a choice for known regions within which all the data points lie (see Figure 5.2). Suppose that

$$d^{(1)} = (x^1, y^1) = \left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, 1 \right)$$

is considered first. With this we would consider adding the rule

$$\mathcal{R}_1 = \textbf{If } x_1 \text{ is } X_1^1 \textbf{ and } x_2 \text{ is } X_2^3 \textbf{ Then } y \text{ is } Y^1$$

(notice that we resolved the tie between choosing Y^1 or Y^2 for the consequent fuzzy set arbitrarily). Since there are no other rules in the rule-base, we will put $\mathcal{R}_1$ in the rule-base and go to the next data pair. Next, consider

$$d^{(2)} = \left(\begin{bmatrix} 2 \\ 4 \end{bmatrix}, 5 \right)$$

With $d^{(2)}$ from Figure 5.8, we would consider adding rule

$$\mathcal{R}_2 = \textbf{If } x_1 \text{ is } X_1^3 \textbf{ and } x_2 \text{ is } X_2^5 \textbf{ Then } y \text{ is } Y^3$$

(where once again we arbitrarily chose Y^3 rather than Y^4). Should we add rule $\mathcal{R}_2$ to the rule-base? Notice that $degree(\mathcal{R}_2) = 0.5$ for $d^{(2)}$ and that $degree(\mathcal{R}_1) = 0$ for $d^{(2)}$ so that $\mathcal{R}_2$ represents the data pair $d^{(2)}$ better than any other rule in the rule-base; hence, we will add it to the rule-base. Proceeding in a similar manner, we will also add a third rule to represent the third data pair in G (show this) so that our final fuzzy system will have three rules, one for each data pair.

If you were to train a fuzzy system with a much larger data set G, you would find that there will not be a rule for each of the M data pairs in G since some rules will adequately represent more than one data pair. Generally, if some x such that $(x, y) \notin G$ is put into the fuzzy system, it will try to interpolate to produce a reasonable output y. You can test the quality of the estimator by putting inputs x into the fuzzy system and checking that the outputs y are such that $(x, y) \in G$, or that they are close to these.

5.6.2 Modified Learning from Examples (MLFE)

We will introduce the "modified learning from examples" (MLFE) technique in this section. In addition to synthesizing a rule-base, in MLFE we also modify the membership functions to try to more effectively tailor the rules to represent the data.

Fuzzy System and Its Initialization

The fuzzy system used in this section utilizes singleton fuzzification, Gaussian membership functions, product for the premise and implication, and center-average defuzzification, and takes on the form

$$f(x|\theta) = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)} \tag{5.54}$$

(however, other forms may be used equally effectively). In Equation (5.54), the parameter vector θ to be chosen is

$$\theta = [b_1, \ldots, b_R, \quad c_1^1, \ldots, c_n^1, \ldots, c_1^R, \ldots, c_n^R, \sigma_1^1, \ldots, \sigma_n^1, \ldots, \sigma_1^R, \ldots, \sigma_n^R]^\top \tag{5.55}$$

where b_i is the point in the output space at which the output membership function for the i^{th} rule achieves a maximum, c_j^i is the point in the j^{th} input universe of discourse where the membership function for the i^{th} rule achieves a maximum, and $\sigma_j^i > 0$ is the width (spread) of the membership function for the j^{th} input and the i^{th} rule. Notice that the dimensions of θ are determined by the number of inputs n and the number of rules R in the rule-base. Next, we will explain how to construct the rule-base for the fuzzy estimator by choosing R, n, and θ. We will do this via the simple example data set G where $n = 2$ that is given in Equation (5.3) on page 236.

We let the quantity ϵ_f characterize the accuracy with which we want the fuzzy system f to approximate the function g at the training data points in G. We also define an "initial fuzzy system" that the MLFE procedure will begin with by initializing the parameters θ. Specifically, we set $R = 1$, $b_1 = y^1$, $c_j^1 = x_j^1$, and $\sigma_j^1 = \sigma_0$ for $j = 1, 2, \ldots, n$ where the parameter $\sigma_0 > 0$ is a design parameter. If we take $\sigma_0 = 0.5$ and

$$(x^1, y^1) = \left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, 1\right)$$

we get $b_1 = 1$, $c_1^1 = 0$, $c_2^1 = 2$, $\sigma_1^1 = 0.5$, and $\sigma_2^1 = 0.5$, which forms our first rule for f.

Next, we describe how to add rules to the fuzzy system and modify the membership functions so that the fuzzy system matches the data and properly interpolates. In the first approach that we describe, we will assume that for the training data $(x^i, y^i) \in G$, $x_i^j \neq x_{i'}^{j'}$ for any $i' \neq i$, for each $j' \neq j$ (i.e., the data values are all distinct element-wise). Later, we will show several ways to remove this restriction. Notice, however, that for practical situations where, for example, you use a noise input for training, this assumption will likely be satisfied.

Adding Rules, Modifying Membership Functions

Following the initialization procedure, for our example we take the second data pair

$$(x^2, y^2) = \left(\begin{bmatrix} 2 \\ 4 \end{bmatrix}, 5 \right)$$

and compare the data pair output portion y^2 with the existing fuzzy system $f(x^2|\theta)$ (i.e., the one with only one rule). If

$$\left| f(x^2|\theta) - y^2 \right| \leq \epsilon_f$$

then the fuzzy system f already adequately represents the mapping information in (x^2, y^2) and hence no rule is added to f and we consider the next training data pair by performing the same type of ϵ_f test.

Suppose that

$$\left| f(x^2|\theta) - y^2 \right| > \epsilon_f$$

Then we add a rule to represent the (x^2, y^2) information about g by modifying the current parameters θ by letting $R = 2$ (i.e., increasing the number of rules by one), $b_2 = y^2$, and $c_j^2 = x_j^2$ for all $j = 1, 2, \ldots, n$ (hence, $b_2 = 5$, $c_1^2 = 2$, and $c_2^2 = 4$).

Moreover, we modify the widths σ_j^i for rule $i = R$ ($i = 2$ for this example) to adjust the spacing between membership functions so that

1. The new rule does not distort what has already been learned.
2. There is smooth interpolation between training points.

Modification of the σ_j^i for $i = R$ is done by determining the "nearest neighbor" n_j^* in terms of the membership function centers that is given by

$$n_j^* = \arg\min\{|c_j^{i'} - c_j^i| : i' = 1, 2, \ldots, R, \quad i' \neq i\} \tag{5.56}$$

where $j = 1, 2, \ldots, n$ and c_j^i is fixed. Here, n_j^* denotes the i' index of the $c_j^{i'}$ that minimizes the expression (hence, the use of the term "argmin"). For our example where we have just added a second rule, $n_1^* = 1$ and $n_2^* = 1$ (the only possible nearest neighbor for each universe of discourse is found from the initial rule in the system).

Next, we update the σ_j^i for $i = R$ by letting

$$\sigma_j^i = \frac{1}{W} |c_j^i - c_j^{n_j^*}| \tag{5.57}$$

for $j = 1, 2, \ldots, n$, where W is a weighting factor that determines the amount of overlap of the membership functions. Notice that since we assumed that for the training data $(x^i, y^i) \in G$, $x_i^j \neq x_{i'}^{j'}$ for any $i' \neq i$, for each $j' \neq j$ we will never

have $\sigma_j^i = 0$, which would imply a zero width input membership function that could cause implementation problems. From Equation (5.57), we see that the weighting factor W and the widths σ_j^i have an inverse relationship. That is, a larger W implies less overlap. For our example, we choose $W = 2$ so $\sigma_1^2 = \frac{1}{2} \mid c_1^2 - c_1^1 \mid = \frac{1}{2} \mid 2 - 0 \mid = 1$ and $\sigma_2^2 = \frac{1}{2} \mid c_2^2 - c_2^1 \mid = \frac{1}{2} \mid 4 - 2 \mid = 1$. The MLFE algorithm is implemented by repeating the above procedure until all the M data pairs are exhausted.

For instance, for our third training data pair,

$$(x^3, y^3) = \left(\begin{bmatrix} 3 \\ 6 \end{bmatrix}, 6 \right)$$

we would test if $\left| f(x^3|\theta) - y^3 \right| \leq \epsilon_f$. If it is, then no new rule is added. If $\left| f(x^3|\theta) - y^3 \right| > \epsilon_f$, then we let $R = 3$ and add a new rule letting $b_R = y^R$ and $c_j^R = x_j^R$ for all $j = 1, 2, \ldots, n$. Then we set the σ_j^R, $j = 1, 2, \ldots, n$ by finding the nearest neighbor n_j^* (nearest in terms of the closest premise membership function centers) and using $\sigma_j^R = \frac{1}{W} \mid c_j^R - c_j^{n_j^*} \mid$, $j = 1, 2, \ldots, n$.

For example, for (x^3, y^3) suppose that ϵ_f is chosen so that $\left| f(x^3|\theta) - y^3 \right| > \epsilon_f$ so that we add a new rule letting $R = 3$, $b_3 = 6$, $c_1^3 = 3$, and $c_2^3 = 6$. It is easy to see from Figure 5.2 on page 237 that with $i = 3$, for $j = 1$, $n_1^* = \arg\min\{|c_j^{i'} - c_j^3| : i' = 1, 2\} = \arg\min\{3, 1\} = 2$ and, for $j = 2$, $n_2^* = \arg\min\{|c_j^{i'} - c_j^3| : i' = 1, 2\} = \arg\min\{4, 2\} = 2$. In other words, $[2, 4]^\top$ is the closest to $[3, 6]^\top$. Hence, via Equation (5.57) with $W = 2$, $\sigma_1^3 = \frac{1}{2}(3 - 2) = \frac{1}{2}$ and $\sigma_2^3 = \frac{1}{2}(6 - 4) = 1$.

Testing the Approximator

To test how accurately the fuzzy system represents the training data set G, note that since we added a new rule for each of the three training data points it will be the case that the fuzzy system $f(x|\theta) = y$ for all $(x, y) \in G$ (why?). If $(x', y') \notin G$ for some x', the fuzzy system f will attempt to interpolate. For instance, for our example above if

$$x' = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

we would expect from Figure 5.2 on page 237 that $f(x'|\theta)$ would lie somewhere between 1 and 5. In fact, for the three-rule fuzzy system we constructed above, $f(x'|\theta) = 4.81$ for this x'. Notice that this value of $f(x'|\theta)$ is quite reasonable as an interpolated value for the given data in G (see Figure 5.2).

Alternative Methods to Modify the Membership Functions

Here, we first remove the restriction that for the training data $(x^i, y^i) \in G$, $x_i^j \neq x_{i'}^{j'}$ for any $i' \neq i$, for each $j' \neq j$ and consider any set of training data G. Following this we will briefly discuss other ways to tune membership functions.

Recall that the only reason that we placed the restriction on G was to avoid having a value of $\sigma_j^i = 0$ in Equation (5.57). One way to avoid having a value of $\sigma_j^i = 0$ is simply to make the computation in Equation (5.57) and if

$$\sigma_j^i < \bar{\sigma}$$

for some small $\bar{\sigma} > 0$, let $\sigma_j^i = \bar{\sigma}$. This ensures that the algorithm will never pick σ_j^i smaller than some preset value. We have found this method to work quite well in some applications.

Another way to avoid having a value of $\sigma_j^i = 0$ from Equation (5.57) is simply to set

$$\sigma_j^i = \sigma_j^{n_j^*}$$

This says that we find the closest membership function center $c_j^{n_j^*}$, and if it is the same as c_j^i then let the width of the membership function associated with c_j^i be the same as that of the membership function associated with $c_j^{n_j^*}$ (i.e., $\sigma_j^{n_j^*}$). Yet another approach would be to compute the width of the c_j^i based not on $c_j^{n_j^*}$ but on the other nearest neighbors that do not have identical centers, provided that there are such centers currently loaded into the rule-base.

There are many other approaches that can be used to train membership functions. For instance, rather than using Equation (5.56), we could let $\underline{c}^i = [c_1^i, c_2^i, \ldots, c_n^i]^\top$ and compute

$$n^* = \arg\min\{|\underline{c}^{i'} - \underline{c}^i| : i' = 1, 2, \ldots, R, \quad i' \neq i\}$$

and then let

$$\sigma_j^i = \frac{1}{W} \mid \underline{c}^i - \underline{c}^{n^*} \mid$$

for $j = 1, 2, \ldots, n$. This approach will, however, need similar fixes to the one above in case the assumption that the input portions of the training data are distinct element-wise is not satisfied.

As yet another approach, suppose that we use triangular membership functions. For initialization we use some fixed base width for the first rule and choose its center $c_j^1 = x_j^1$ as before. Use the same ϵ_f-test to decide whether to add rules. If you add a rule, let $c_j^i = x_j^i$, $i = R$, $j = 1, 2, \ldots, n$ as before. Next, to fully specify the membership functions, compute

$$n_j^- = \arg\min\{|c_j^i - c_j^{i'}| : i' = 1, 2, \ldots, M, c_j^{i'} < c_j^i\}$$
$$n_j^+ = \arg\min\{|c_j^i - c_j^{i'}| : i' = 1, 2, \ldots, M, c_j^{i'} > c_j^i\}.$$

These are the indices of the nearest neighbor membership functions both above

and below c_i^j. Then draw a line from the point $(c_j^{n_j^-}, 0)$ to $(c_j^i, 1)$ to specify the left side of the triangle and another line from $(c_j^i, 1)$ to $(c_j^{n_j^+}, 0)$ to specify the right side of the triangle. Clearly, there is a problem with this approach if there is no $i' = 1, 2, \ldots, M$ such that $c_j^{i'} < c_j^i$ ($c_j^{i'} > c_j^i$) in computing n_j^- (n_j^+). If there is such a problem, then simply use some fixed parameter (say, c^-), draw a line from $(c_j^i - c^-, 0)$ to $(c_j^i, 1)$ for the left side of the triangle, and use the above approach for the right side of the triangle assuming that n_j^+ can be computed. Similarly, if there is such a problem in computing n_j^+, then simply use some fixed parameter (say, c^+), draw a line from $(c_j^i, 1)$ to $(c_j^i + c^+, 0)$ for the right side of the triangle, and use the above approach for the left side of the triangle assuming that n_j^- can be computed. If both n_j^+ and n_j^- cannot be computed, put the center at c_j^i and draw a line from $(c_j^i - c^-, 0)$ to $(c_j^i, 1)$ for the left side of the triangle and a line from $(c_j^i, 1)$ to $(c_j^i + c^+, 0)$ for the right side.

Clearly, the order of processing the data will affect the results using this approach. Also, we would need a fix for the method to make sure that there are no zero base width triangles (i.e., singletons). Approaches analogous to our fix for the Gaussian input membership functions could be used. Overall, we have found that this approach to training fuzzy systems can perform quite well for some applications.

Design Guidelines

In this section we investigated the LFE and MLFE approaches to construct fuzzy estimators. At this point the reader may wonder which technique is the best. While no theoretical comparisons have been done, we have found that for a variety of applications the MLFE technique does tend to use fewer rules to get comparable accuracy to the LFE technique; however, we have found some counterexamples to this. While the LFE technique does require the designer to specify all the membership functions, it is relatively automatic after that. The MLFE does not require the designer to pick the membership functions but does require specification of three design parameters. We have found that most often we can use intuition gained from the application to pick these parameters.

Overall, we must emphasize that there seems to be no clear winner when comparing the LFE and MLFE techniques. It seems best to view them as techniques that provide valuable insight into how fuzzy systems operate and how they can be constructed to approximate functions that are inherently represented in data. The LFE technique shows how rules can be used as a simple representation for data pairs. Since the constructed rules are added to a fuzzy system, we capitalize on its interpolation capabilities and hence get a mapping for data pairs that are not in the training data set. The MLFE technique shows how to tailor membership functions and rules to provide for an interpolation that will attempt to model the data pairs. Hence, the MLFE technique specifies both the rules and membership functions.

5.7 Hybrid Methods

In this chapter we have discussed least squares (batch and recursive), gradient (steepest descent, Newton, and Gauss-Newton), clustering (with optimal output predefuzzification and nearest neighbor), learning from examples, and modified learning from examples methods for training standard and Takagi-Sugeno fuzzy systems. In this section we will discuss hybrid approaches where we combine two or more of the above methods to train a fuzzy system.

Basically, the hybrid methods can be classified three ways:

- *Hybrid initialization/training:* You could initialize the parameters of the fuzzy system with one method and then use a different method for the training. For instance, you could use the learning by examples methods to create a fuzzy system that you could later tune with a gradient or least squares method. Alternatively, you could use a least squares method to initialize the output centers of a standard fuzzy system and then use a gradient method to tune the premise parameters and to fine-tune the output centers.

- *Hybrid premise/consequent training:* You could train the premises of the rules with one method and the consequents with another. This is exactly what is done in clustering with optimal output predefuzzification in Section 5.5.1 on page 274: A clustering method is used to specify the premise parameters, and least squares is used to train the consequent functions since they are linear in the parameters. Other examples of hybrid training of this type include the use of least squares for training the consequent functions of a Takagi-Sugeno fuzzy system (since they enter linearly) and a gradient method for training the premise parameters (since they enter in a nonlinear fashion). Alternatively, you could train the premises with a clustering method and the consequents with a gradient method (especially for a functional fuzzy system that has consequent functions that are not linear in the parameters). Still another option would be to use ideas from the learning from examples techniques to train the premises and use the least squares or gradient methods for the consequents.

- *Hybrid interleaved training:* You could train with one method then another, followed by another, and so on. For instance, you could use a learning by examples method to initialize the fuzzy system parameters, then train the fuzzy system with a gradient method, with periodic updates to the output membership function centers coming from a least squares method.

Basically, all these methods provide the advantage of design flexibility for the tuning of fuzzy systems. While some would view this flexibility in a positive light, it does have its drawbacks, primarily in trying to determine which approach to use, or the best combination of approaches.

Indeed, it is very difficult to know which of the methods in this chapter to use as the choice ultimately depends on the application. Moreover, it is important to have in mind what you mean by "best." This would likely involve accuracy of esti-

mation or identification, but it could also focus on the computational complexity of implementing the resulting fuzzy system. We have found that for some applications the LFE and MLFE approaches can take many rules to get good identification accuracy (as can the nearest neighborhood clustering approach). Sometimes the computations for large data sets are prohibitive for batch least squares, but recursive least squares can then be used. Sometimes gradient training takes a large amount of training data and extremely long training times. On the other hand, the clustering with optimal output predefuzzification method needs relatively few rules (partially because one Takagi-Sugeno rule carries more information than one standard fuzzy rule) and hence often results in low computational complexity while at the same time providing better accuracy; it seems to exploit the advantages of the least squares approach and ideas in clustering that result in well-tuned input membership functions. We do not, however, consider this finding universal. For other applications, one of the other methods (e.g., gradient and least squares), or a combination of the above methods, may provide a better approach.

In the next section we provide a design and implementation case study where we use the clustering with optimal output predefuzzification approach.

5.8 Case Study: FDI for an Engine

In recent years more attention has been given to reducing exhaust gas emissions produced by internal combustion engines. In addition to overall engine and emission system design, correct or fault-free engine operation is a major factor determining the amount of exhaust gas emissions produced in internal combustion engines. Hence, there has been a recent focus on the development of on-board diagnostic systems that monitor relative engine health. Although on-board vehicle diagnostics can often detect and isolate some major engine faults, due to widely varying driving environments they may be unable to detect minor faults, which may nonetheless affect engine performance. Minor engine faults warrant special attention because they do not noticeably hinder engine performance but may increase exhaust gas emissions for a long period of time without the problem being corrected.

The minor faults we consider in this case study include "calibration faults" (for our study, the occurrence of a calibration fault means that a sensed or commanded signal is multiplied by a gain factor not equal to one, while in the no-fault case the sensed or commanded signal is multiplied by one) in the throttle and mass fuel actuators, and in the engine speed and mass air sensors (we could also consider "bias"-type faults even though we do not do so in this case study). The reliability of these actuators and sensors is particularly important to the engine controller since their failure can affect the performance of the emissions control system.

Our particular focus in this design and implementation case study is to show how to construct fuzzy estimators to perform failure detection and identification (FDI) for certain actuator and sensor calibration faults. We compare the results from the fuzzy estimators to a nonlinear autoregressive moving average with exogenous inputs (ARMAX) technique and provide experimental results showing the effectiveness of the technique. Next, we provide an overview of the experimental

engine test bed and testing conditions that we use.

5.8.1 Experimental Engine and Testing Conditions

All investigations in this case study were performed using the experimental engine test cell shown in Figure 5.9. The experimental setup in the engine test cell consists of a Ford 3.0 L V-6 engine coupled to an electric dynamometer through an automatic transmission. An air charge temperature sensor (ACT), a throttle position sensor (TPS), and a mass airflow sensor (MAF) are installed in the engine to measure the air charge temperature, throttle position, and air mass flow rate. Two heated exhaust gas oxygen sensors (HEGO) are located in the exhaust pipes upstream of the catalytic converter. The resultant airflow information and input from the various engine sensors are used to compute the required fuel flow rate necessary to maintain a prescribed air-to-fuel ratio for the given engine operation. The central processing unit (EEC-IV) determines the needed injector pulse width and spark timing, and outputs a command to the injector to meter the exact quantity of fuel. An ECM (electronic control module) breakout box is used to provide external connections to the EEC-IV controller and the data acquisition system. The angular velocity sensor system consists of a digital magnetic zero-speed sensor and a specially designed frequency-to-voltage converter, which converts frequency signals proportional to the rotational speed into an analog voltage. Data is sampled every engine revolution. A variable load is produced through the dynamometer, which is controlled by a DYNLOC IV speed/torque controller in conjunction with a DTC-1 throttle controller installed by DyneSystems Company. The load torque and dynamometer speed are obtained through a load cell and a tachometer, respectively. The throttle and the dynamometer load reference inputs are generated through a computer program and sent through an RS-232 serial communication line to the controller. Physical quantities of interest are digitized and acquired utilizing a National Instruments AT-MIO-16F-5 A/D timing board for a personal computer.

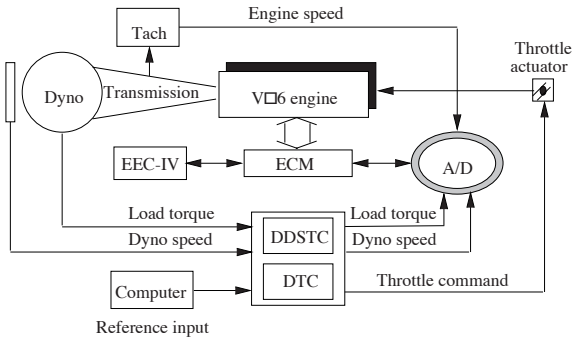


FIGURE 5.9 The experimental engine test cell (figure taken from [109], © IEEE).

Due to government mandates, periodic inspections and maintenance for engines are becoming more common. One such test developed by the Environmental Protection Agency (EPA) is the Inspection and Maintenance (IM) 240 cycle. The EPA IM240 cycle (see Figure 5.10 for vehicle speed, in mph, plotted versus time) represents a driving scenario developed for the purpose of testing compliance of vehicle emissions systems for contents of carbon monoxide (CO), unburned hydrocarbons (HC), and nitrogen oxides (NO_x). The IM240 cycle is designed to be performed under laboratory conditions with a chassis dynamometer and is patterned after the Urban Dynamometer Driving Schedule (UDDS), which approximates a portion of a morning commute within an urban area. This test is designed to evaluate the emissions of a vehicle under real-world conditions. In [97], the authors propose an additional diagnostic test to be performed during the IM240 cycle to detect and isolate a class of minor engine faults that may hinder vehicle performance and increase the level of exhaust emissions. Since the EPA proposes to make the test mandatory for all vehicles, performing an additional diagnostic analysis in parallel would provide a controlled test that might allow for some minor faults to be detected and corrected, thus reducing overall exhaust emissions in a large number of vehicles.

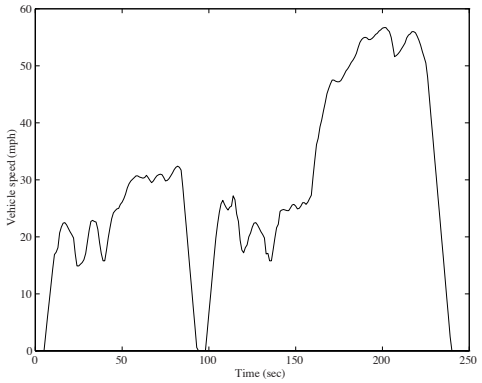


FIGURE 5.10 The EPA IM240 engine cycle (figure taken from [109], © IEEE).

5.8.2 Fuzzy Estimator Construction and Results

In system identification, which forms the basis for our FDI technique, we wish to construct a model of a dynamic system using input-output data from the system. The types of engine faults that the FDI strategy is designed to detect include the calibration faults given in Table 5.3. These faults directly affect the resulting fuel-to-air ratio and spark timing in combustion, which subsequently affects the level of exhaust gas emissions. The fault detection and isolation strategy relies on estimates of ω (engine speed, in rpm), m_a (mass rate of air entering the intake manifold, in lb-m/sec), α (actuated throttle angle, expressed as a percentage of a

TABLE 5.3 Types of Faults Detectable with FDI Strategy

Fault	Type	Description
m_a	sensor calibration	Measures amount of air intake for combustion
ω	sensor calibration	Measures engine speed
α	actuator calibration	Actuates throttle angle
m_f	actuator calibration	Actuates amount of fuel for combustion

full-scale opening), m_f (mass of fuel entering the combustion chamber, in lb-m), and T_L (the load torque on the engine, in ft-lb) (which we denote by $\hat{\omega}$, $\hat{m}_a$, $\hat{\alpha}$, $\hat{m}_f$, and $\hat{T}_L$, respectively) that are provided by identifying models f_ω, f_{m_a}, f_α, f_{m_f}, and f_{T_L} of how the engine operates. In particular, we have

$$\hat{\omega} = f_\omega(x_\omega) \tag{5.58}$$
$$\hat{m}_a = f_{m_a}(x_{m_a}) \tag{5.59}$$
$$\hat{\alpha} = f_\alpha(x_\alpha) \tag{5.60}$$
$$\hat{m}_f = f_{m_f}(x_{m_f}) \tag{5.61}$$
$$\hat{T}_L = f_{T_L}(x_{T_L}). \tag{5.62}$$

where the inputs are given in Equations (5.63)–(5.67) (k is a discrete time in the crankshaft domain where physical quantities are sampled every turn of the engine crankshaft)

$$\begin{aligned} x_\omega = [&\hat{\omega}(k-1), \hat{\omega}(k-2), \hat{\omega}(k-3), \alpha(k-1), \alpha(k-2), \\ &\alpha(k-3), m_f(k-1), m_f(k-2), m_f(k-3), \hat{T}_L(k-2)]^\top \end{aligned} \tag{5.63}$$
$$\begin{aligned} x_{m_a} = [&\hat{m}_a(k-1), \hat{m}_a(k-2), \hat{m}_a(k-3), \alpha(k-1), \alpha(k-2), \\ &m_f(k-1), m_f(k-2), m_f(k-3), \hat{T}_L(k-1), \hat{T}_L(k-3)]^\top \end{aligned} \tag{5.64}$$
$$\begin{aligned} x_\alpha = [&\hat{\alpha}(k-1), \hat{\alpha}(k-2), \hat{\alpha}(k-3), m_a(k-1), \\ &m_a(k-2), m_a(k-3), \omega_{dy}(k-1), \omega_{dy}(k-2)]^\top \end{aligned} \tag{5.65}$$
$$\begin{aligned} x_{m_f} = [&\hat{m_f}(k-1), \hat{m_f}(k-2), \hat{m_f}(k-3), m_a(k-1), \\ &m_a(k-2), m_a(k-3), \omega(k-1), \omega(k-2), \omega(k-3)]^\top \end{aligned} \tag{5.66}$$
$$\begin{aligned} x_{T_L} = [&\hat{T}_L(k-1), \hat{T}_L(k-2), \hat{T}_L(k-3), \alpha(k-1), \alpha(k-2), \\ &m_f(k-1), m_a(k-1), m_a(k-3), \omega_{dy}(k-1), \omega_{dy}(k-3)]^\top \end{aligned} \tag{5.67}$$

where ω_{dy} is an output of the dynamometer. These regression vectors were chosen using simulation and experimental studies to determine which variables are useful in estimating others and how many delayed values must be used to get accurate estimation.

One approach to nonlinear system identification that has been found to be particularly useful for this application [119, 120] and that we will employ in the current study in addition to the fuzzy estimation approach is the NARMAX (nonlinear ARMAX) method, which is an extension of the linear ARMAX system identification technique. The general model structure for NARMAX uses scaled polynomial

combinations of the arguments contained in the regression vector; here we use the NARMAX model structure given by

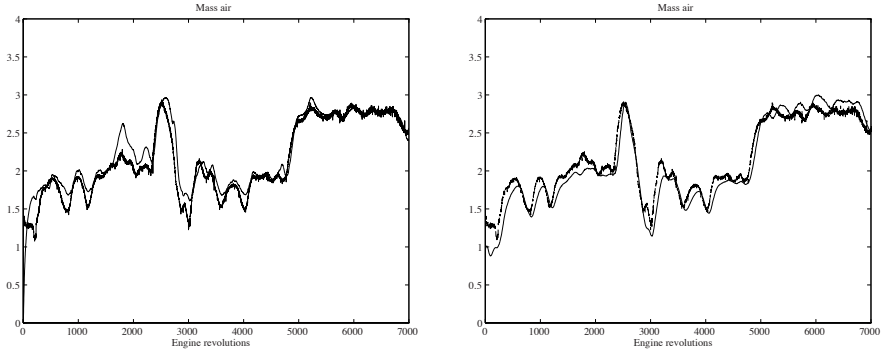
$$\hat{y}(k) = \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} x_i x_j \tag{5.68}$$

where β_i, β_{ij} are parameters to be adjusted so that $\hat{y}(k)$ is as close as possible to $y(k)$ for all $x \in \Re^n$ (i.e., we use only one second-order polynomial term in our model structure). As is usual, in this case study we will use the standard batch least squares approach to adjust the β_i, β_{ij} since they enter linearly.

For training purposes, data were collected to calculate the necessary models f_ω, f_{m_a}, f_α, f_{m_f}, and f_{T_L}. Due to mechanical constraints on the electric dynamometer, we reduced the IM240 cycle to only 7000 engine revolutions for the tests that we ran. In addition, a uniformly distributed random signal was added to the throttle and torque inputs in order to excite the system. The data generated were utilized to construct five multi-input single-output fuzzy systems, one for each of the Equations (5.58)–(5.62). In fuzzy clustering we choose 10 clusters ($R = 10$), a fuzziness factor $m = 2$, and tolerance $\epsilon_c = 0.01$ for each of the constructed fuzzy systems. These were derived via experimentation until desired accuracy was achieved (e.g., increasing R to more than 10 did not provide improved estimation accuracy). For comparison purposes, we also calculated models utilizing the nonlinear ARMAX technique based on the same experimental data. Then the experimental test cell was run, and the models derived through fuzzy clustering and the nonlinear ARMAX technique were validated by collecting data for similar tests run on different days.

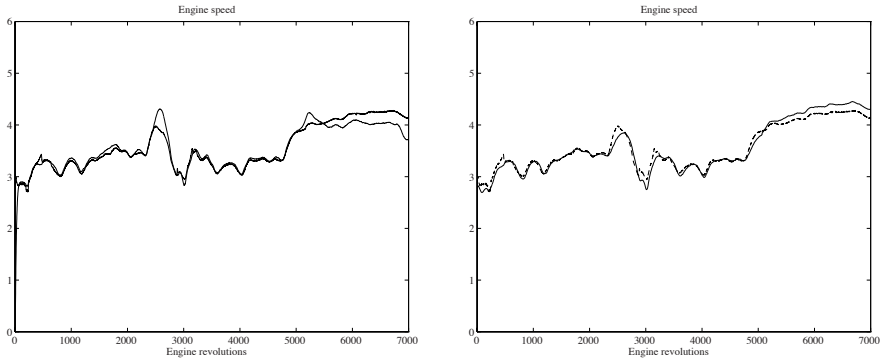
The results in identification with the validation data (not the training data) for both techniques are given in Figures 5.11, 5.12, 5.13, 5.14, and 5.15 (plots in (a) show the results for the fuzzy identification approach and in (b) for the NARMAX approach). We measure approximation error by evaluating the squared error between the real and estimated value (which we denote by $\sum_k e^2$ where k ranges over the entire simulation time). As the results show, both techniques approximate the real system fairly well; however, for the mass air and engine speed estimates, the NARMAX technique performed slightly better than the clustering technique. For the throttle, load torque, and the mass fuel estimates, the clustering technique estimated slightly better than the NARMAX technique.

Overall, we see that there is no clear overall advantage to using NARMAX or the fuzzy estimator even though the fuzzy estimator performs better for estimating several variables. We would comment, however, that it took a significant amount of experimental work to determine where to truncate the polynomial expansion in Equation (5.68) for the NARMAX model structure. The parameters R, m, and ϵ_c for the fuzzy estimator construction were, however, quite easy to select. Moreover, the fuzzy estimation approach provides the additional useful piece of information that the underlying system seems to be adequately represented by interpolating between 10 linear systems each of which is represented by the output of the ten



(a) Clustering $\sum_k e^2 = 37.3813$ (b) NARMAX $\sum_k e^2 = 34.9451$

FIGURE 5.11 Mass air for (a) clustering, (noisy signal) measured, (smooth signal) estimate and (b) for NARMAX, (noisy signal) measured, (smooth signal) estimate (figure taken from [109], © IEEE).



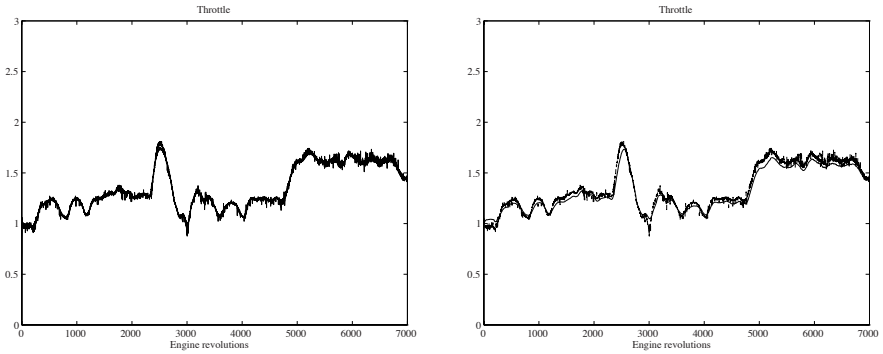
(a) Clustering $\sum_k e^2 = 43.6903$ (b) NARMAX $\sum_k e^2 = 12.7043$

FIGURE 5.12 Engine speed for (a) clustering, (measured one is higher than the estimate near 7000 engine revolutions) and (b) for NARMAX, (solid) measured, (dashed) estimate (figure taken from [109], © IEEE).

rules ($R = 10$).

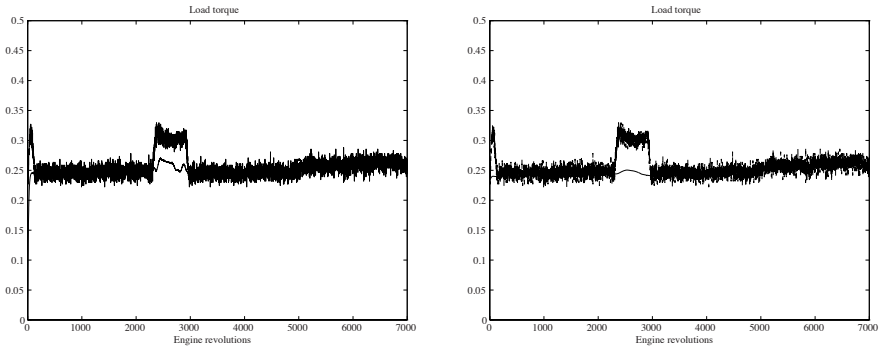
5.8.3 Failure Detection and Identification (FDI) Strategy

The models identified through fuzzy clustering and optimal output predefuzzification allow us to utilize system residuals (e.g., $\hat{\omega} - \omega$, $\hat{m}_a - m_a$, $\hat{\alpha} - \alpha$, and $\hat{m}_f - m_f$) to detect and isolate failures. A specific fault may be isolated by referring to the fault isolation logic given in Table 5.4 that was developed by the "indirect decoupling method" outlined in [98]. In the body of Table 5.4 we indicate a pattern of "zero", "nonzero" and "—" (don't care) residuals that will allow us to identify the appropriate failure. We use thresholds to define what we mean by "zero" and



(a) Clustering $\sum_k e^2 = 1.4896$ (b) NARMAX $\sum_k e^2 = 2.8904$

FIGURE 5.13 Throttle for (a) clustering, (solid) measured, (dotted) estimate and (b) for NARMAX, (noisy signal) measured, (smooth signal) estimate (figure taken from [109], © IEEE).

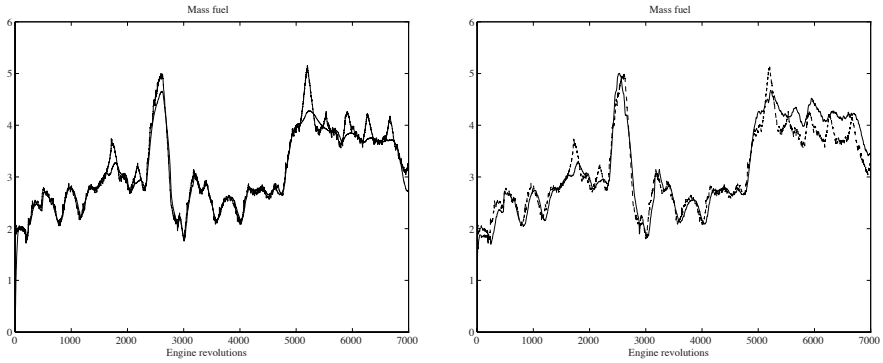


(a) Clustering $\sum_k e^2 = 0.5336$ (b) NARMAX $\sum_k e^2 = 0.5896$

FIGURE 5.14 Load torque for (a) clustering, (noisy signal) measured, (smooth signal) estimate and (b) for NARMAX, (noisy signal) measured, (smooth signal) estimate (figure taken from [109], © IEEE).

"nonzero" and explain how we choose these thresholds below. As an example, if the scaled values (we will explain how the residuals are scaled below) of $|\hat{\omega} - \omega|$, $|\hat{m}_a - m_a|$, and $|\hat{m}_f - m_f|$ are above some thresholds, and $|\hat{\alpha} - \alpha|$ is below some threshold, then we say that there is an m_f actuator calibration fault. For an m_a sensor calibration fault, the (scaled) value of $|\hat{\omega} - \omega|$ should be nonzero since this residual is not completely decoupled, but it is very weakly coupled through the load torque model. Therefore, we have the "—" (don't care) term for the $\hat{\alpha} - \alpha$ residual for an m_a sensor calibration fault.

The models developed via fuzzy clustering and optimal output predefuzzification are only approximations of the real engine dynamics. Therefore, since the system residuals do not identically equal zero during nominal no-fault operation, it



(a) Clustering $\sum_k e^2 = 48.5399$ (b) NARMAX $\sum_k e^2 = 83.8961$

FIGURE 5.15 Mass fuel for (a) clustering, (noisy signal with peaks) measured, (smooth signal) estimate and (b) for NARMAX, (dashed) measured, (thin solid) estimate (figure taken from [109], © IEEE).

TABLE 5.4 Catalog of System Residuals and Corresponding Faults

Fault Location	$\hat{\omega} - \omega$	$\hat{m}_a - m_a$	$\hat{\alpha} - \alpha$	$\hat{m}_f - m_f$
m_a sensor	—	Nonzero	Nonzero	Nonzero
ω sensor	Nonzero	Zero	Nonzero	Nonzero
α input	Nonzero	Nonzero	Nonzero	Zero
m_f input	Nonzero	Nonzero	Zero	Nonzero

is necessary to perform some postprocessing of the residuals to detect and isolate the faults we consider. We perform a low pass filtering of system residuals and a setting of thresholds to determine nonzero residuals. We implement a fourth-order Butterworth low pass filter with a cutoff frequency of $\frac{\pi}{100}$ and pass the residuals through this filter. Next, we take the filtered residual and scale it by dividing by the maximum value of the signal over the entire IM240 cycle. The filtered and scaled residual is then compared against a threshold, and if the threshold is exceeded, then a binary signal of one is given for that particular residual for the remainder of the test. The threshold values for each residual used in the FDI strategy are computed empirically by analyzing the deviation of the residuals from zero during no-fault operation. These thresholds are given in Table 5.5 (e.g., from Table 5.5, if the filtered and scaled residual for m_a is greater than 0.30, then we say the $\hat{m}_a - m_a$ residual threshold has been exceeded—i.e., that it is "nonzero").

We perform tests utilizing the FDI strategy by simulating calibration faults and using the filtered residuals. Specifically, calibration faults are simulated by multiplying the experimental data for a specific fault by the desired calibration fault value. For instance, to obtain a 20% ω calibration fault, we multiply ω by 1.20. Through experimentation we have found this to be an accurate representation of a true calibration fault.

TABLE 5.5 Thresholds for System Residuals

Residual	Threshold
$\hat{m}_a - m_a$ sensor	$\pm$ 0.30
$\hat{\omega} - \omega$ sensor	$\pm$ 0.10
$\hat{\alpha} - \alpha$ input	$\pm$ 0.04
$\hat{m}_f - m_f$ input	$\pm$ 0.15

We look at only a portion of the IM240 cycle when we test for faults. The portion we observe is between 3000 and 5000 revolutions of the engine. During this portion the best model matching occurred. Figure 5.16 shows the residuals lying within the thresholds for the duration of the test, signaling a no-fault condition.

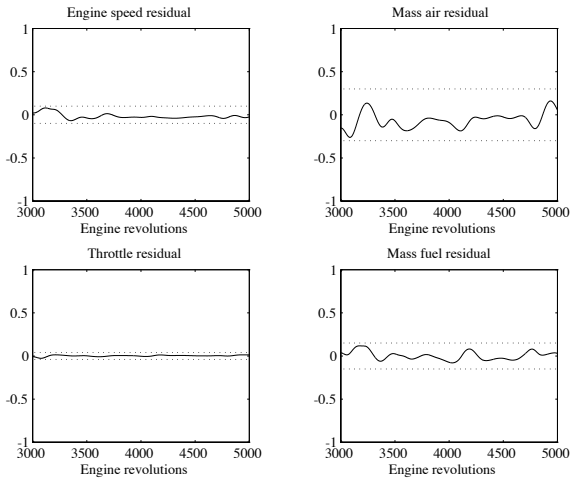


FIGURE 5.16 System residuals with no fault (the vertical axes are dimensionless; they represent the size of the filtered residual divided by the maximum value of the signal achieved over the IM240 cycle) (figure taken from [109], © IEEE).

In the second test a 20% calibration fault exists in the throttle actuator, meaning that the throttle angle is 1.20 times the commanded value. As Figure 5.17 illustrates, all residuals exceed the threshold except the m_f residual—which according to Table 5.4, indicates that a throttle fault is present. Similar results are obtained for a 20% calibration fault in the mass air sensor (meaning that the mass air sensor reads 1.20 times the real value), a 40% calibration fault in the mass fuel actuator (meaning that the mass fuel actuator injects 1.40 times the commanded value), and a 20% calibration fault in the engine speed sensor. In a similar manner, engine failures can be detected utilizing the models calculated via the NARMAX

technique; however, the resulting residuals are not shown here as they were very similar. Overall, we see that by combining the estimates from the fuzzy estimators with the FDI logic, we were able to provide an effective FDI strategy for a class of minor engine failures.

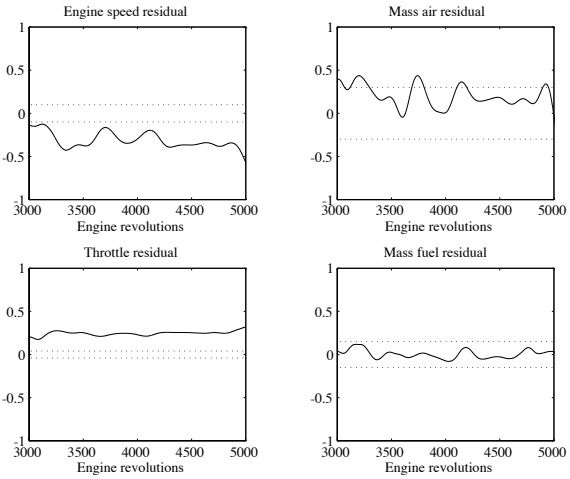


FIGURE 5.17 System residuals with 20% throttle calibration fault (the vertical axes are dimensionless; they represent the size of the filtered residual divided by the maximum value of the signal achieved over the IM240 cycle) (figure taken from [109], © IEEE).

5.9 Summary

In this chapter we have provided an introduction to several techniques on how to construct fuzzy systems using numerical data. We used a simple example to illustrate how several of the methods operate. The least squares method can be used to train linear systems and should be considered as a conventional alternative to the other methods (since it can sometimes be easier to implement). Gradient methods are especially useful for training parameters that enter in a nonlinear fashion. The clustering and optimal output predefuzzification method combined the conventional least squares method with the c-means clustering technique that provided for the specification of the input membership functions and served to interpolate between the linear models that were specified via least squares. Clustering based on nearest neighborhood methods helps to provide insight into fuzzy system construction. The LFE and MLFE techniques provide unique insights into how to associate rules with data pairs to train the fuzzy system to map the input-output data. The chapter closed with a discussion on how to combine the methods of this chapter into hybrid

training techniques and a design and implementation case study.

Upon completing this chapter, you should understand the following:

- The function approximation problem.
- How construction of models, estimators, predictors, and controllers can be viewed as a special case of the function approximation problem.
- The issues involved in choosing a training data set.
- How to incorporate linguistic information into a fuzzy system that you train with data.
- The batch and recursive least squares methods.
- How to train standard or Takagi-Sugeno fuzzy systems with least squares methods.
- The gradient algorithm method for training a standard or Takagi-Sugeno fuzzy system.
- The clustering with optimal output predefuzzification method for constructing a Takagi-Sugeno fuzzy system.
- The nearest neighborhood clustering method for training standard fuzzy systems.
- The learning from examples (LFE) method for constructing a fuzzy system.
- The modified learning from examples (MLFE) method for constructing a fuzzy system.
- How different methods can be combined into hybrid training techniques for fuzzy systems.
- How the clustering with optimal output predefuzzification method can be used for failure detection and identification in an internal combustion engine.

Essentially, this is a checklist of the major topics of this chapter. The gradient or recursive least squares methods are essential for understanding the indirect adaptive fuzzy control method treated in Chapter 6, and some of the supervisory control ideas in Chapter 7 rely on the reader's knowledge of at least one method from this chapter.

5.10 For Further Study

An earlier version of the problem formulation for the function approximation problem appeared in [108]. The idea of combining linguistic information with the fuzzy system constructed from numerical training data has been used by several researchers and is exploited in a particularly coherent way in [229].

The idea of using least squares to train fuzzy systems was first introduced in [207] and was later studied in [232] and other places. For more details on least squares methods, see [127]. The gradient method for training fuzzy systems was originally developed as the "back-propagation" approach for training neural networks, and many people recognized that such gradient methods could also be used for training fuzzy systems (e.g., see [231] for a treatment of the steepest descent approach). For more details on gradient methods, see [128, 22]. The clustering with optimal output predefuzzification approach was introduced in [187] but is modified somewhat from its original form in our presentation. The nearest neighbor clustering approach was introduced in [228]. For more details on fuzzy clustering, see [24, 23, 89, 187, 236, 177]. The learning from examples technique was first introduced in [233], and the modified learning from examples approach [108] was developed using ideas from the approaches in [133, 233]. A slightly different approach to the computation in Equation (5.56) on page 287 is taken in [108], where the MLFE was first introduced. The hybrid methods have been used by a variety of researchers; a particularly nice set of applications were studied in [81, 82].

The case study in implementation of the fuzzy estimators for an internal combustion engine was taken from [109]. All investigations in this case study were performed using the experimental engine test cell in [129, 130]. We take the same basic approach to FDI for minor engine faults as in [97] except that we utilize a fuzzy estimation approach rather than the nonlinear ARMAX approach used in [97]. Related work on the use of nonlinear ARMAX is given in [130].

The case study in engine failure estimation used in the chapter and in the problems at the end of the chapter, and the cargo ship failure estimation problem used in the problems at the end of the chapter were developed in [143]. For a general overview of the field of FDI, see [166].

Some other methods related to the topics in this chapter are given in [73, 18, 237, 117, 1, 76]. There is related work in the area of neural networks also. See, for example, [150, 26]. A good introduction to the topical area of this chapter is given in [188, 86], where the authors also cover wavelets, neural networks, and other approximators and properties in some detail.

5.11 Exercises

Exercise 5.1 (Training Fuzzy Systems to Approximate a Simple Data Set): In this problem you will study the training of standard and Takagi-Sugeno fuzzy systems to represent the data in G in Equation (5.3) on page 236. This is the data set that was used as an example throughout the chapter. You can program the methods on the computer yourself or use the Matlab code provided at the web and ftp sites listed in the Preface.

(a) Batch least squares: For the example in Section 5.3.3 on page 255, find the value of $\hat{\theta}$ and compare it with the value found there. Also, test the fuzzy system with the same six test inputs and verify that the outputs are as given in Section 5.3.3. Next, let $c_1^1 = 0$, $c_2^1 = 2$, $c_1^2 = 2$, and $c_2^2 = 4$; find $\hat{\theta}$; and

repeat the testing process for the six test inputs. Does the fuzzy system seem to interpolate well?

(b) Weighted batch least squares: Choose $W = diag([10, 1, 1])$ so that we weight the first data pair in G (i.e., $([0, 2]^\top, 1)$) as being the most important one. Does this make $f([0, 2]^\top|\hat{\theta})$ closer to one (y^1) than for the fuzzy system trained in (a)?

(c) Recursive least squares: Repeat (a) but use RLS with $\lambda = 1$.

(d) Weighted recursive least squares: Repeat (a) but use weighted RLS with $\lambda = 0.9$.

(e) Gradient method: Verify all computed values for the gradient training of the Takagi-Sugeno fuzzy system in Section 5.4.3 (this includes finding the fuzzy system parameter values and the fuzzy system outputs for the six test inputs).

(f) Clustering with optimal output predefuzzification: Verify all computed values for all cases in the example of Section 5.5.1. Be sure to include the case where we use one more training data pair (i.e., when $M = 4$).

Exercise 5.2 (Training a Fuzzy System to Approximate a Simple Function): In this problem you will study methods for constructing fuzzy systems to approximate the mapping defined by a quadratic function

$$y = 2x_1^2 + x_2^2 \tag{5.69}$$

over the range $x_1 \in [-12, 12]$, $x_2 \in [-12, 12]$ (note that here x_1^2 is x_1 squared).

To train and test the fuzzy system, use the training data set G and the test data set Γ, defined as

$$G = \{([x_1, x_2]^\top, y) \mid x_1, x_2 \in \{-12, -10.5, -9, \ldots, 9, 10.5, 12\}, y = 2x_1^2 + x_2^2\} \tag{5.70}$$

$$\Gamma = \{([x_1, x_2]^\top, y) \mid x_1, x_2 \in \{-12, -11, -10, \ldots, 10, 11, 12\}, y = 2x_1^2 + x_2^2\} \tag{5.71}$$

The training data set G and the test data set Γ are used for each technique in this problem. Plot the function over the range of values provided in the input portions of the data in Γ.

For each constructed fuzzy system for each part below, calculate the maximum error, e_{max}, defined as

$$e_{max} = \max\{|f(x) - y| : (x, y) \in \Gamma\}$$

where f is the fuzzy system output and y is the output of the quadratic function and the "percentage maximum error," which is defined as

$$e_{pmax} = 100\frac{e_{max}}{y^*}$$

where y^* is the y value of the quadratic function that makes the error maximum—that is,

$$y^* \in \{y' : (x', y') \in \Gamma \text{ and } |f(x') - y'| \geq |f(x) - y| \text{ for all } (x, y) \in \Gamma\}$$

Clearly, we would like to construct the fuzzy system so that e_{max} and e_{pmax} are minimized.

(a) Batch least squares: Consider the fuzzy system

$$y = f(x|\theta) = \frac{\sum_{i=1}^{R} b_i \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} \tag{5.72}$$

where $x = [x_1, x_2]^\top$ and $\mu_i(x)$ is the certainty of the premise of the i^{th} rule that is specified by Gaussian membership functions. We use singleton fuzzification and product for premise and implication, and b_i is the center of the output membership function for the i^{th} rule. The Gaussian membership functions have the form

$$\mu(x) = \exp\left(-\frac{1}{2}\left(\frac{x - c}{\sigma}\right)^2\right)$$

so that

$$\mu_i(x) = \prod_{j=1}^{2} \exp\left(-\frac{1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right)$$

is defined where c_j^i is the center of the membership function of the i^{th} rule for the j^{th} universe of discourse and σ_j^i is the relative width of the membership function of the i^{th} rule of the j^{th} universe of discourse.

We use nine membership functions for each input universe of discourse with centers given by the elements of the set

$$\{-12, -9, -6, -3, 0, 3, 6, 9, 12\}$$

from which we can see that the membership functions are distributed uniformly. From these centers we will form $R = 9 \times 9 = 81$ rules—that is, a rule for every possible combination. We must specify the input membership function centers c_j^i. Let the columns of the matrix

$$\begin{bmatrix} -12 & -12 & \cdots & -12 & -9 & -9 & \cdots & -9 & \cdots & 12 \\ -12 & -9 & \cdots & 12 & -12 & -9 & \cdots & 12 & \cdots & 12 \end{bmatrix}$$

be denoted by c^i, and let $c^i = [c_1^i, c_2^i]^\top$, $i = 1, 2, \ldots, 81$. The relative widths of the membership functions, σ_j^i for all $j = 1, 2$ and $i = 1, 2, \ldots, R$, are

chosen as 4 to get a reasonable coverage of each universe of discourse.

Write a computer program to implement batch least squares to find the output centers. Compute e_{pmax}, plot the input-output map of the resulting fuzzy system, and compare it to the plot of the quadratic function.

(b) Gradient method: Use the "standard" fuzzy system defined in Section 5.4 for which the update formulas are already derived in Equations (5.35), (5.36), and (5.37). You have $n = 2$. The centers of the membership functions should be initialized to the values that we used in part (a), and their relative widths should be initialized as $\sigma_j^i = 1$. Also, let

$$b_i = 2(c_1^i)^2 + (c_2^i)^2$$

for all $i = 1, 2, \ldots, 81$ and $j = 1, 2$. Why is this a reasonable choice for the initial values? As can be seen from the definition above, you use 9 membership functions for each input universe of discourse, which means 81 rules (i.e., $R = 81$). Choose λ_1, λ_2, and λ_3 as 0.0001. In your algorithm, cycle through the entire training data set G many times, processing one data pair in every step of the the gradient algorithm, until the error e_m is less than or equal to 1.6.

Write a computer program to implement the gradient method to train the fuzzy system. Compute e_{pmax}, plot the input-output map of the resulting fuzzy system, and compare it to the plot of the quadratic function.

(c) Clustering with optimal output predefuzzification: Here, you train the fuzzy system defined in the chapter. Choose $R = 49$, $m = 2$, and $\epsilon_c = 0.0001$. In this problem, two definitions of g_j are used. The first one is given by

$$g_j = a_{j,0} + a_{j,1}x_1 + a_{j,2}x_2$$

and the second one is defined as

$$g_j = a_{j,0} + a_{j,1}(x_1)^2 + a_{j,2}(x_2)^2$$

where x_i is the i^{th} input value, $j = 1, 2, \ldots, R$, and the $(x_i)^2$ terms represent that we assume in this case that we have special knowledge about the function we want to approximate (i.e., in this case we assume that we know it is a quadratic function). Initialize the cluster centers as

$$v_i^j = (v_i^j)' + \epsilon_p(r - 0.5)$$

where $(v_i^j)' \in \{-12, -8, \ldots, 8, 12\}$, $j = 1, 2, \ldots, R$, $i = 1, 2$, $\epsilon_p = 0.001$, and "r" is a random number between 0 and 1.

Write a computer program to implement the clustering with optimal output predefuzzification method to train the fuzzy system. Compute e_{pmax}, plot the input-output map of the resulting fuzzy system, and compare it to the plot of the quadratic function.

(d) Nearest neighborhood clustering: The fuzzy system used in this part is the one that the nearest neighborhood clustering method was developed for in the chapter. Initialize the parameters $A_1 = y_1 = 432$, $B_1 = 1$, and $v_j^1 = x_j^1$ for $j = 1, 2$ (i.e., $v_1^1 = -12$ and $v_2^1 = -12$). Let $\epsilon_f = 0.5$ and the relative width of the membership functions $\sigma = 5$, which affects the accuracy of the approximation.

Write a computer program to implement the nearest neighborhood clustering method to train the fuzzy system. What is the number of clusters that are produced? Compute e_{pmax}, plot the input-output map of the resulting fuzzy system, and compare it to the plot of the quadratic function.

(e) Learning from examples: For our fuzzy system we use singleton fuzzification, minimum to represent the premise and implication, and "center of gravity" (COG) defuzzification. Choose the effective universes of discourse for the two inputs and the output as

$$\mathcal{X}_1 = [x_1^-, x_1^+] = [-12, 12]$$

$$\mathcal{X}_2 = [x_2^-, x_2^+] = [-12, 12]$$

$$\mathcal{Y} = [y^-, y^+] = [0, 432]$$

This choice is made since we seek to approximate our unknown function over $x_1 \in [-12, 12], x_2 \in [-12, 12]$ and we know that for these values $y \in [0, 432]$. You should use triangular-shaped membership functions for $\mu_{X_1^j}(x_1)$, $\mu_{X_2^j}(x_2)$, $\mu_{Y^j}(y)$, associated with the fuzzy sets X_1^j, X_2^j, and Y^j, respectively. In particular, use 9 membership functions for inputs x_1 and x_2 and 45 membership functions for the output y; these membership functions are shown in Figures 5.18, 5.19, and 5.20.

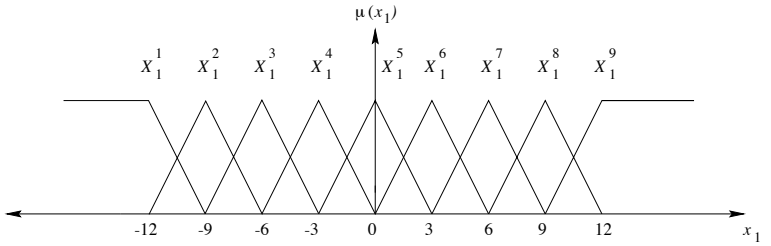


FIGURE 5.18 Membership functions for the x_1 universe of discourse (figure created by Mustafa K. Guven).

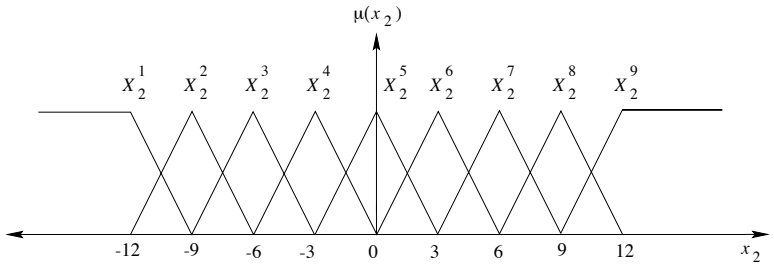


FIGURE 5.19 Membership functions for the x_2 universe of discourse (figure created by Mustafa K. Guven).

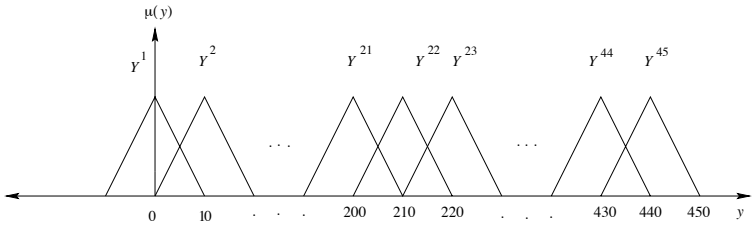


FIGURE 5.20 Membership functions for the y universe of discourse (figure created by Mustafa K. Guven).

Write a computer program to implement the learning from examples method to train the fuzzy system. Compute e_{pmax}, plot the input-output map of the resulting fuzzy system, and compare it to the plot of the quadratic function.

Note that the choice of G results in a "full" rule-base table (i.e., the table is completely filled in). For this example, if you reduce the number of training data enough, you will not end up with a full rule-base table.

(f) Modified learning from examples: For the fuzzy system use singleton fuzzification, Gaussian membership functions, product for the premise and implication, and center-average defuzzification (i.e., the one used to introduce the modified learning from examples technique in the chapter). To start training, first initialize the fuzzy system parameters including the number of rules, $R = 1$; the center of the output membership function for the first rule, $b_1 = 432$; the centers of the input membership functions for the first rule, $c_1^1 = -12$ and $c_2^1 = -12$; and the relative widths of the membership functions for the first rule, $\sigma_1^1 = 1$ and $\sigma_2^1 = 1$. This forms the first rule. Choose $\epsilon_f = 0.001$.

When a new rule is added, the relative widths of the membership functions for the new rule should be updated. To do that, we need to compare the x_1^2

with the centers of the membership functions that are in the first universe of discourse, c_1^i for all $i = 1, 2, \ldots, R$ and x_2^2 with c_2^i for all $i = 1, 2, \ldots, R$, to find the nearest membership functions for each universe of discourse. Using the distance between the input portion of the training data and the nearest membership function, we determine the σ_j^i for all $i = 1, 2, \ldots, R$ and $j = 1, 2$. Specifically, let

$$\sigma_j^i = \frac{|c_j^i - c_j^{n_j^*}|}{W}$$

for all $i = 1, 2, \ldots, R$ and $j = 1, 2$ and where W is the weighting factor, which we choose as 2.

Since we have repeated numbers in our data set (i.e., there exist i' and j' such that $x_j^i = x_{j'}^{i'}$, for $i \neq i'$, $j \neq j'$), we will have a problem for some σ_j^i for $i = 1, 2, \ldots, R$ and $j = 1, 2$. For instance, assume that we have $R = 1$ (i.e., we have one rule in our rule-base) and let

$$[c_1^1, c_2^1]^\top = [-12, -12]^\top$$

and the next training data is

$$([x_1, x_2]^\top, y) = ([-12, -10.5]^\top, 398.25)$$

For the x_1 universe of discourse, the nearest membership function is the one that has -12 as its center. Therefore, the distance and the σ_1^2 will be zero. Because of this, during testing of the fuzzy system, we will have

$$\frac{x_1 - c_1^2}{\sigma_1^2} = \frac{x_1 - c_1^2}{0}$$

which is not well-defined.

To avoid this situation, the following procedure can be used. If for $[x_1, x_2]^\top$, $c_j^i = x_j$, then let

$$\sigma_j^{R+1} = \sigma_j^R$$

for $j = 1, 2$. For our example,

$$\sigma_1^2 = \sigma_1^1$$

With this procedure, instead of updating the relative width of the new rule, we will keep one of the old relative widths for the new rule. Other ways to solve the problem of updating the relative widths are given in the chapter.

Write a computer program to implement the modified learning from examples method to train the fuzzy system. Compute e_{pmax}, plot the input-

output map of the resulting fuzzy system, and compare it to the plot of the quadratic function.

Exercise 5.3 (Estimation of ζ for a Second-Order System): In this problem suppose that you are given a plant that is perfectly represented by a second-order transfer function

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Suppose that $\omega_n = 1$ and that you know $0.1 \leq \zeta \leq 1$, but that you do not know its value and you would like to estimate it using input-output data from the plant. Assume that you know that the input to $G(s)$ and in each case below choose the input trajectory and provide a rationale for your choice. Assume that you do not know that the system is perfectly linear and second order (so that you cannot simply use, e.g., standard least squares for estimating ζ).

(a) Generate the set G of training data. Provide a rationale for your choice.

(b) Use the batch least squares method to construct a fuzzy estimator for ζ. Provide all the details on how you construct your estimator (including all design parameters).

(c) Use the gradient method to construct a fuzzy estimator for ζ. Provide all the details on how you construct your estimator (including all design parameters).

(d) Use the clustering with optimal output predefuzzification method to construct a fuzzy estimator for ζ. Provide all the details on how you construct your estimator (including all design parameters).

(e) Use the LFE method to construct a fuzzy estimator for ζ. Provide all the details on how you construct your estimator (including all design parameters).

(f) Use the MLFE method to construct a fuzzy estimator for ζ. Provide all the details on how you construct your estimator (including all design parameters).

(g) Test the ζ estimators that you constructed in (b)–(f). Be sure to test them for data that you trained them with, and with data that you did not use in training. Provide plots that show both the estimated and actual values of ζ on the same plot.

Exercise 5.4 (Least Squares Derivation): Recall that for batch least squares we had

$$V(\theta, M) = \frac{1}{2}E^\top E$$

as a measure of the approximation error. In this problem you will derive several of the least squares methods that were developed in this chapter.

(a) Using basic ideas from calculus, take the partial of V with respect to θ and set it equal to zero. From this derive an equation for how to pick $\hat{\theta}$. Compare it to Equation (5.15). Hint: If m and b are two $n \times 1$ vectors and A is an $n \times n$ symmetric matrix (i.e., $A = A^\top$), then $\frac{d}{dm} b^\top m = b$, $\frac{d}{dm} m^\top b = b$, and $\frac{d}{dm} m^\top A m = 2Am$.

(b) Repeat (a) for the weighted batch least squares approach, where V is chosen as in Equation (5.16), and compare it to Equation (5.17).

(c) Derive the update Equations (5.26) for the weighted recursive least squares approach.

Exercise 5.5 (Gradient Training of Fuzzy Systems): In this problem you will derive gradient update formulas for fuzzy systems by directly building on the discussion in the chapter.

(a) Derive the update equations for $b_i(k)$, $c_j^i(k)$, and $\sigma_j^i(k)$ for the gradient training method described in Section 5.4 on page 260. Show the full details of the derivations for all three cases.

(b) Repeat (a) but for the Takagi-Sugeno fuzzy system so that you will find the update formula for $a_{i,j}(k)$ rather than for $b_i(k)$.

(c) Repeat (a) but for a generalization of the Takagi-Sugeno fuzzy system (i.e., a functional fuzzy system) with the same parameters as in the chapter except

$$g_i(x) = a_{i,0} + a_{i,1}(x_1)^2 + \cdots + a_{i,n}(x_n)^2$$

$i = 1, 2, \ldots, R$. In this case our gradient method will try to train the $a_{i,j}$, c_j^i, and σ_j^i to find a fuzzy system that provides nonlinear interpolation between R quadratic functions.

(d) Repeat (c) but for

$$g_i(x) = a_{i,0} + \exp\left[a_{i,1}(x_1)^2\right] + \cdots + \exp\left[a_{i,n}(x_n)^2\right]$$

5.12 Design Problems

Design Problem 5.1 (Identification of a Fuzzy System Model of a Tank): Suppose that you are given the "surge tank" system that is shown in Figure 6.44 on page 399. Suppose that the differential equation representing this system is

$$\frac{dh(t)}{dt} = \frac{-c\sqrt{2gh(t)}}{A(h(t))} + \frac{1}{A(h(t))} u(t)$$

where $u(t)$ is the input flow (control input) that can be positive or negative (it can pull liquid out of the tank), $h(t)$ is the liquid level (the output $y(t) = h(t)$), $A(h(t))$ is the cross-sectional area of the tank, $g = 9.8$ m/sec^2 is acceleration due to gravity, and $c = 1$ is the known cross-sectional area of the output pipe. Assume

that $a = 1$ and that $A(h) = ah^2 + b$ where $a = 1$ and $b = 2$ (i.e., that we know the tank characteristics exactly). Assume, however, that you have only an idea about what the order of the system is. That is, assume that you do not know that the system is governed *exactly* by the above differential equation. Hence, you will treat this system as your physical system ("truth model") when you gather data to perform identification of the model. When you then want to test the validity of the model that you construct with your identification approaches, you test it against the truth model.

(a) Use the fuzzy clustering with optimal output predefuzzification approach to construct a fuzzy system whose input-output behavior is similar to that of the surge tank. Clearly explain your approach, any assumptions that you make, and the design parameters you choose. Also, be careful in your choice of the training data set. Make sure that the input $u(t)$ properly excites the system dynamics.

(b) Develop a second identification approach to producing a fuzzy system model, different from the one in (a).

(c) Perform a comparative analysis between the approaches in (a) and (b) focusing on how well the fuzzy system models you produced via the identification approaches model the physical system represented by the truth model. To do this, be sure to test the system with inputs different from those you used to train the models.

Design Problem 5.2 (Gasket Leak Estimation for an Engine)*: Government regulations that attempt to minimize environmental impact and safety hazards for automobiles have motivated the need for estimation of engine parameters that will allow us to determine if an engine has failed. In this problem you will develop a fuzzy estimator for estimating the parameter k_2 for the engine described in Section 5.2.5 on page 243 (i.e., use the data in G_{k_2}).

(a) Establish an engine failure simulator as it is described in Section 5.2.5 on page 243. Demonstrate that your engine failure simulator produces the same results as in Section 5.2.5. Develop the training data set G_{k_2} for the engine as it is described in Section 5.2.5.

(b) Choose a method from the chapter and develop a fuzzy estimator for k_2. You can either use the data G_{k_2} or compute your own training data set.

(c) Next, test the failure estimator using the engine simulator for the failure scenario for k_2 in Table 5.2. The testing process is implemented using the engine failure simulator and a constant step of $\Theta = 0.1$ for both an ideal no-disturbance condition and a disturbance input T_L of the form shown in Figure 5.4 on page 245. Plot the estimates and actual values on the same graph, and evaluate the accuracy of the estimator.

Design Problem 5.3 (Engine Friction Estimation)*: In this problem you will use the data set G_{k_5} in Section 5.2.5 on page 243 to design parameter estimators for the parameter k_5, which can indicate whether there is excessive friction in the engine.

(a) Establish an engine failure simulator as it is described in Section 5.2.5 on page 243. Demonstrate that your engine failure simulator produces the same results as in Section 5.2.5. Develop the training data set G_{k_5} for the engine as it is described in Section 5.2.5.

(b) Choose a method from the chapter and develop a fuzzy estimator for k_5. You can either use the data G_{k_5} or compute your own training data set.

(c) Test the quality of the estimators that you developed in (b). The testing process should be implemented using the engine failure simulator from (a) and a constant step of $\Theta = 0.1$ for both an ideal no-disturbance condition and a disturbance input T_L of the form shown in Figure 5.4 on page 245. Plot the estimates and actual values on the same graph, and evaluate the accuracy of the estimator.

Design Problem 5.4 (Cargo Ship Failure Estimator)*: In this problem we introduce the cargo ship models and the failure modes to be considered, and you will first develop a failure simulator test bed for the cargo ship. We use the cargo ship model given in Chapter 6, Section 6.3.1 on page 333, where the rudder angle δ is used to steer the ship along a heading ψ (see Figure 6.5). The reference input is ψ_d and $e = \psi_d - \psi$. A simple control law, such as proportional-integral-derivative (PID) control, is typically used in autopilot regulation of the ship. In this problem we will use a proportional derivative (PD) controller of the form

$$\delta = k_p e + k_d \dot{e} \tag{5.73}$$

where we choose $k_p = -3.1$ and $k_d = 105$. Closed-loop control of this form will be used in both training and testing of failure estimators for the ship.

The inputs to the failure simulator should be the desired ship heading ψ_d and the possible parameter changes representing failures. Again, there are training and testing inputs; these are shown in Figure 5.21. Unlike the engine failure simulator developed in Section 5.2.5, there exist two cargo ship models where the model to be used depends whether we are constructing the failure identifier or testing it. If the simulator is being used to train the fuzzy failure estimator, then we select the training input and use the third-order linear model in Equation (6.6) on page 334. We use the nonlinear model, in Equation (6.7) on page 335, when testing the failure estimator methods.

There are two parameters that are varied to represent failures in the cargo ship. They are the velocity parameter u (which represents an inaccurate speed sensor reading), and a bias in the control variable δ. The value of the failed parameter for

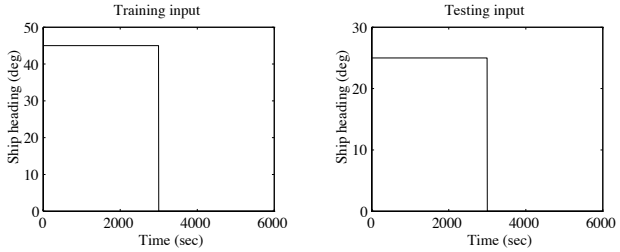


FIGURE 5.21 Failure simulator inputs (plots created by Sashonda Morris).

the velocity u is determined by equations similar to Equations (5.11) and (5.12) that were used for the engine failure simulator. The value of the rudder angle failure δ(failure) is determined by adding a constant bias, $\pm\ \bar{\delta}$, to the nominal rudder angle value.

Table 5.6 shows the failure scenarios that we would like to be able to predict (so we would like to estimate u and δ).

TABLE 5.6 Failure Scenarios for Cargo Ship

Parameter	Nominal Value	Failure Setting
u	5 m/s	-80%
δ	$\delta \in [-45^\circ, +45^\circ]$	$\delta = \pm 5$

(a) Do four simulations, all for the "testing" input for the ship when the nonlinear ship model is used. The first should show the response of the closed-loop control system for the nominal no-failure case. The second should show how the closed-loop control system will respond to a speed sensor failure of -80% induced at $t = 0$. The third (fourth) should show how the closed-loop control system will respond to a rudder angle bias error of $+5$ degrees (-5 degrees) that is induced at $t = 0$. Each of the four simulations should be run for 6000 seconds. What effect does the speed sensor failure have on rise-time, settling time, and overshoot? What effect does the rudder angle bias have on the steady-state error?

(b) Using the cargo ship failure simulator from (a) with the linear ship model and the training input ψ_d shown in Figure 5.21, data sets should be generated for training fuzzy estimators. The parameters u and δ should be varied over a specified range of values to account for the possible failure scenarios the cargo ship steering system might encounter. The parameter u should be varied between 0% and 90% of its nominal value (i.e., $\Delta u \in [0, 0.9]$) at 10%

increments yielding $M_u = 10$ output responses. The constant bias value $\bar{\delta}$ should be varied between $\pm$ 10 at an increment of 2, yielding $M_\delta = 11$ output responses. Plot the output responses when the parameters u and δ are varied in this way.

Explain why a good way to estimate a failure in the velocity parameter u is based on the rise-time and percent overshoot of the output responses, while a failure in the control parameter δ is best characterized by the steady-state error $e = \psi_d - \psi$. The percent overshoot can be best characterized by the error $e = \psi_d - \psi$ responses for the given parameter u. Plot the error responses for variations in the velocity parameter u and the control parameter δ.

The error responses of the cargo ship when the parameter u is varied should be used to form the data $d^{(m)}$ for training the estimators for the velocity parameter. A moving window of length = 200 seconds should be used to sample the response at an interval of $T = 50$ seconds. Notice that most information about a particular failure is contained between 100 and 1200 seconds for a step input of $\psi_d = 45°$, and between 3100 and 4200 seconds for a step input of $\psi_d = 0°$. The error responses should be sampled over these ranges. The full set of cargo ship failure data for the speed sensor is then given by

$$\begin{aligned} G_u = \{([&e^j(kT), e^j(kT-T), e^j(kT-2T), e^j(kT-3T), \\ &e^j(kT-4T) ce^j(kT)]^\top, u^j) : \\ &k \in \{1, 2, \ldots, 23\}, 1 \le j \le M_u \ \} \end{aligned} \tag{5.74}$$

where u^j denotes the j^{th} value ($1 \le j \le M_u$) of u and $e^j(kT), e^j(kT - T), e^j(kT-2T), e^j(kT-3T), e^j(kT-4T)$, and $ce^j(kT)$ represent the corresponding values of $e(kT), e(kT-T), e(kT-2T), e(kT-3T), e(kT-4T)$, and $ce(kT)$ that were generated using this u^j. The value of $ce^j(kT)$ is the current change in error given by

$$ce^j(kT) = \frac{e^j(kt) - e^j(kT-T)}{T} \tag{5.75}$$

and u^j represents the size of the failure and the parameter we want to estimate. Generate the data set G_u.

(c) The failure data sets for the control parameter δ should be formed using the error responses generated when the parameter δ was varied. Because the responses for this parameter settle within 500 seconds, the fuzzy estimator is trained between 50 and 500 seconds at a sampling period $T = 50$ seconds. The full set of cargo ship failure data for the control variable δ is given by

$$\begin{aligned} G_\delta = \{([&e^j(kT), e^j(kT-T), e^j(kT-2T)]^\top, \delta^j) : \\ &k \in \{1, 2, \ldots, 8\}, 1 \le j \le M_\delta\} \end{aligned} \tag{5.76}$$

where $e^j(kT), \ldots, e^j(kT-2T)$ are the sampled values of the error $e = \psi_d -$

ψ, and δ^j represents the failure and the parameter we want to estimate. Generate the data set G_δ.

(d) Using a method of your choice, train the estimators for both failures. Test the quality of the estimators and provide plots of estimated and actual values on the same graph.

CHAPTER 6

Adaptive Fuzzy Control

They know enough who know how to learn.
–Henry Brooks Adams

6.1 Overview

The design process for fuzzy controllers that is based on the use of heuristic information from human experts has found success in many industrial applications. Moreover, the approach to constructing fuzzy controllers via numerical input-output data, which we described in Chapter 5, is increasingly finding use. Regardless of which approach is used, however, there are certain problems that are encountered for practical control problems, including the following: (1) The design of fuzzy controllers is performed in an ad hoc manner so it is often difficult to choose at least some of the controller parameters. For example, it is sometimes difficult to know how to pick the membership functions and rule-base to meet a specific desired level of performance. (2) The fuzzy controller constructed for the nominal plant may later perform inadequately if significant and unpredictable plant parameter variations occur, or if there is noise or some type of disturbance or some other environmental effect. Hence, it may be difficult to perform the initial synthesis of the fuzzy controller, and if the plant changes while the closed-loop system is operating we may not be able to maintain adequate performance levels.

As an example, in Chapter 3 we showed how our heuristic knowledge can be used to design a fuzzy controller for the rotational inverted pendulum. However, we also showed that if a bottle half-filled with water is attached to the endpoint, the performance of the fuzzy controller degraded. While we certainly could have tuned the controller for this new situation, it would not then perform as well without a bottle of liquid at the endpoint. It is for this reason that we need a way to automatically tune the fuzzy controller so that it can adapt to different plant

conditions. Indeed, it would be nice if we had a method that could automatically perform the whole design task for us initially so that it would also synthesize the fuzzy controller for the nominal condition. In this chapter we study systems that can automatically synthesize and tune (direct) fuzzy controllers.

There are two general approaches to adaptive control, the first of which is depicted in Figure 6.1. In this approach the "adaptation mechanism" observes the signals from the control system and adapts the parameters of the controller to maintain performance even if there are changes in the plant. Sometimes, the desired performance is characterized with a "reference model," and the controller then seeks to make the closed-loop system behave as the reference model would even if the plant changes. This is called "model reference adaptive control" (MRAC).

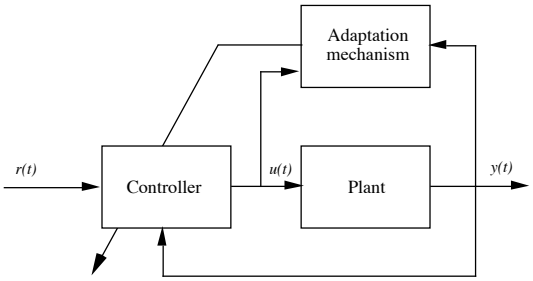


FIGURE 6.1 Direct adaptive control.

In Section 6.2 we use a simple example to introduce a method for direct (model reference) adaptive fuzzy control where the controller that is tuned is a fuzzy controller. Next, we provide several design and implementation case studies to show how it compares to conventional adaptive control for a ship steering application, how to make it work for a multi-input multi-output (MIMO) fault-tolerant aircraft control problem, and how it can perform in implementation for the two-link flexible robot from Chapter 3 to compensate for the effect of a payload variation.

Following this, in Section 6.4 we show several ways to "dynamically focus" the learning activities of an adaptive fuzzy controller. A simple magnetic levitation control problem is used to introduce the methods, and we compare the performance of the methods to a conventional adaptive control technique. Design and implementation case studies are provided for the rotational inverted pendulum (with a sloshing liquid in a bottle at the endpoint) and the machine scheduling problems from Chapter 3.

In the second general approach to adaptive control, which is shown in Figure 6.2, we use an on-line system identification method to estimate the parameters of the plant and a "controller designer" module to subsequently specify the parameters of the controller. If the plant parameters change, the identifier will provide estimates of these and the controller designer will subsequently tune the controller. It is inherently assumed that we are certain that the estimated plant parameters are

equivalent to the actual ones at all times (this is called the "certainty equivalence principle"). Then if the controller designer can specify a controller for each set of plant parameter estimates, it will succeed in controlling the plant. The overall approach is called "indirect adaptive control" since we tune the controller indirectly by first estimating the plant parameters (as opposed to direct adaptive control, where the controller parameters are estimated directly without first identifying the plant parameters). In Section 6.6 we explain how to use the on-line estimation techniques described in Chapter 5 (recursive least squares and gradient methods), coupled with a controller designer, to achieve indirect adaptive fuzzy control for nonlinear systems. We discuss two approaches, one based on feedback linearization and the other we name "adaptive parallel distributed compensation" since it builds on the parallel distributed compensator discussed in Chapter 4.

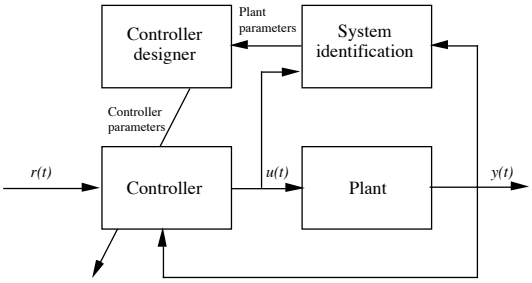


FIGURE 6.2 Indirect adaptive control.

Upon completing this chapter, the reader will be able to design a variety of adaptive fuzzy controllers for practical applications. The reader should consider this chapter fundamental to the study of fuzzy control systems as adaptation techniques such as the ones presented in this chapter have proven to be some of the most effective fuzzy control methods. Given a firm understanding of Chapter 2 (and parts of Chapter 3), it is possible to cover the material in this chapter on direct adaptive fuzzy control in Sections 6.2–6.5 without having read anything else in the book. The reader wanting to cover this entire chapter will, however, need a firm understanding of all the previous chapters except Chapter 4. The reader does not need to cover this chapter to understand the basic concepts in the next one; however, a deeper understanding of the concepts in this chapter will certainly be beneficial for the next chapter since fuzzy supervisory control provides yet another approach to adaptive control.

6.2 Fuzzy Model Reference Learning Control (FMRLC)

A "learning system" possesses the capability to improve its performance over time by interacting with its environment. A learning control system is designed so that

its "learning controller" has the ability to improve the performance of the closed-loop system by generating command inputs to the plant and utilizing feedback information from the plant.

In this section we introduce the "fuzzy model reference learning controller" (FMRLC), which is a (direct) model reference adaptive controller. The term "learning" is used as opposed to "adaptive" to distinguish it from the approach to the conventional model reference adaptive controller for linear systems with unknown plant parameters. In particular, the distinction is drawn since the FMRLC will tune and to some extent *remember* the values that it had tuned in the past, while the conventional approaches for linear systems simply continue to tune the controller parameters. Hence, for some applications when a properly designed FMRLC returns to a familiar operating condition, it will already know how to control for that condition. Many past conventional adaptive control techniques for linear systems would have to retune each time a new operating condition is encountered.

The functional block diagram for the FMRLC is shown in Figure 6.3. It has four main parts: the plant, the fuzzy controller to be tuned, the reference model, and the learning mechanism (an adaptation mechanism). We use discrete time signals since it is easier to explain the operation of the FMRLC for discrete time systems. The FMRLC uses the learning mechanism to observe numerical data from a fuzzy control system (i.e., $r(kT)$ and $y(kT)$ where T is the sampling period). Using this numerical data, it characterizes the fuzzy control system's current performance and automatically synthesizes or adjusts the fuzzy controller so that some given performance objectives are met. These performance objectives (closed-loop specifications) are characterized via the reference model shown in Figure 6.3. In a manner analogous to conventional MRAC where conventional controllers are adjusted, the learning mechanism seeks to adjust the fuzzy controller so that the closed-loop system (the map from $r(kT)$ to $y(kT)$) acts like the given reference model (the map from $r(kT)$ to $y_m(kT)$). Basically, the fuzzy control system loop (the lower part of Figure 6.3) operates to make $y(kT)$ track $r(kT)$ by manipulating $u(kT)$, while the upper-level adaptation control loop (the upper part of Figure 6.3) seeks to make the output of the plant $y(kT)$ track the output of the reference model $y_m(kT)$ by manipulating the fuzzy controller parameters.

Next, we describe each component of the FMRLC in more detail for the case where there is one input and one output from the plant (we will use the design and implementation case studies in Section 6.3 to show how to apply the approach to MIMO systems).

6.2.1 The Fuzzy Controller

The plant in Figure 6.3 has an input $u(kT)$ and output $y(kT)$. Most often the inputs to the fuzzy controller are generated via some function of the plant output $y(kT)$ and reference input $r(kT)$. Figure 6.3 shows a simple example of such a map that has been found to be useful in some applications. For this, the inputs to the

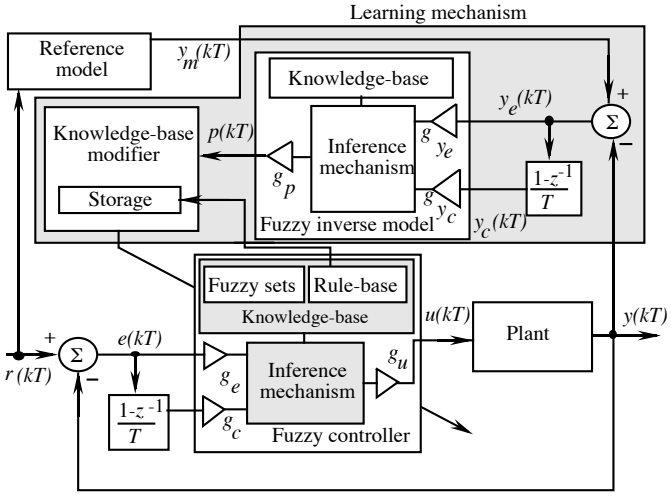


FIGURE 6.3 Fuzzy model reference learning controller (figure taken from [112], © IEEE).

fuzzy controller are the error $e(kT) = r(kT) - y(kT)$ and change in error

$$c(kT) = \frac{e(kT) - e(kT - T)}{T}$$

(i.e., a PD fuzzy controller). There are times when it is beneficial to place a smoothing filter between the $r(kT)$ reference input and the summing junction. Such a filter is sometimes needed to make sure that smooth and reasonable requests are made of the fuzzy controller (e.g., a square wave input for $r(kT)$ may be unreasonable for some systems that you know cannot respond instantaneously). Sometimes, if you ask for the system to perfectly track an unreasonable reference input, the FMRLC will essentially keep adjusting the "gain" of the fuzzy controller until it becomes too large. Generally, it is important to choose the inputs to the fuzzy controller, and how you process $r(kT)$ and $y(kT)$, properly; otherwise performance can be adversely affected and it may not be possible to maintain stability.

Returning to Figure 6.3, we use scaling gains g_e, g_c, and g_u for the error $e(kT)$, change in error $c(kT)$, and controller output $u(kT)$, respectively. A first guess at these gains can be obtained in the following way: The gain g_e can be chosen so that the range of values that $e(kT)$ typically takes on will not make it so that its values will result in saturation of the corresponding outermost input membership functions. The gain g_c can be determined by experimenting with various inputs to the fuzzy control system (without the adaptation mechanism) to determine the normal range of values that $c(kT)$ will take on. Using this, we choose the gain g_c so that normally encountered values of $c(kT)$ will not result in saturation of the outermost input membership functions. We can choose g_u so that the range of

outputs that are possible is the maximum one possible yet still so that the input to the plant will not saturate (for practical problems the inputs to the plant will always saturate at some value). Clearly, this is a very heuristic choice for the gains and hence may not always work. Sometimes, tuning of these gains will need to be performed when we tune the overall FMRLC.

Rule-Base

The rule-base for the fuzzy controller has rules of the form

$$\textbf{If } \tilde{e} \text{ is } \tilde{E}^j \textbf{ and } \tilde{c} \text{ is } \tilde{C}^l \textbf{ Then } \tilde{u} \text{ is } \tilde{U}^m$$

where $\tilde{e}$ and $\tilde{c}$ denote the linguistic variables associated with controller inputs $e(kT)$ and $c(kT)$, respectively, $\tilde{u}$ denotes the linguistic variable associated with the controller output u, $\tilde{E}^j$ and $\tilde{C}^l$ denote the j^{th} (l^{th}) linguistic value associated with $\tilde{e}$ ($\tilde{c}$), respectively, and $\tilde{U}^m$ denotes the consequent linguistic value associated with $\tilde{u}$. Hence, as an example, one fuzzy control rule could be

If error is positive-large **and** change-in-error is negative-small
Then plant-input is positive-big

(in this case $\tilde{e}$ = "error", $\tilde{E}^4$ = "positive-large", etc.). We use a standard choice for all the membership functions on all the input universes of discourse, such as the ones shown in Figure 6.4. Hence, we would simply use some membership functions similar to those in Figure 6.4, but with a scaled horizontal axis, for the $c(kT)$ input.

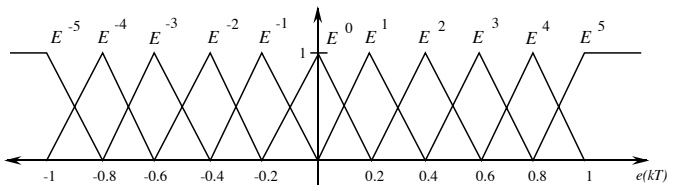


FIGURE 6.4 Membership functions for input universe of discourse (figure taken from [112], © IEEE).

We will use all possible combinations of rules for the rule-base. For example, we could choose to have 11 membership functions on each of the two input universes of discourse, in which case we would have $11^2 = 121$ rules in the rule-base. At first glance it would appear that the complexity of the controller could make implementation prohibitive for applications where it is necessary to have many inputs to the fuzzy controller. However, we must remind the reader of the results in Section 2.6 on page 97 where we explain how implementation tricks can be used to significantly reduce computation time when there are input membership functions of the form shown in Figure 6.4.

Rule-Base Initialization

The input membership functions are defined to characterize the premises of the rules that define the various situations in which rules should be applied. The input membership functions are left constant and are not tuned by the FMRLC. The membership functions on the output universe of discourse are assumed to be unknown. They are what the FMRLC will automatically synthesize or tune. Hence, the FMRLC tries to fill in what actions ought to be taken for the various situations that are characterized by the premises.

We must choose initial values for each of the output membership functions. For example, for an output universe of discourse $[-1, 1]$ we could choose triangular-shaped membership functions with base widths of 0.4 and centers at zero. This choice represents that the fuzzy controller initially knows nothing about how to control the plant so it inputs $u = 0$ to the plant initially (well, really it does know something since we specify the remainder of the fuzzy controller a priori). Of course, one can often make a reasonable best guess at how to specify a fuzzy controller that is "more knowledgeable" than simply placing the output membership function centers at zero. For example, we could pick the initial fuzzy controller to be the best one that we can design for the nominal plant. Notice, however, that this choice is not always the best one. Really, what you often want to choose is the fuzzy controller that is best for the operating condition that the plant will begin in (this may not be the nominal condition). Unfortunately, it is not always possible to pick such a controller since you may not be able to measure the operating condition of the plant, so making a best guess or simply placing the membership function centers at zero are common choices.

To complete the specification of the fuzzy controller, we use minimum or product to represent the conjunction in the premise and the implication (in this book we will use minimum unless otherwise stated) and the standard center-of-gravity defuzzification technique. As an alternative, we could use appropriately initialized singleton output membership functions and center-average defuzzification.

Learning, Memorization, and Controller Input Choice

For some applications you may want to use an integral of the error or other preprocessing of the inputs to the fuzzy controller. Sometimes the same guidelines that are used for the choice of the inputs for a nonadaptive fuzzy controller are useful for the FMRLC. We have found, however, times where it is advantageous to replace part of a conventional controller with a fuzzy controller and use the FMRLC to tune it (see the fault-tolerant control application in Section 6.3). In these cases the complex preprocessing of inputs to the fuzzy controller is achieved via a conventional controller. Sometimes there is also the need for postprocessing of the fuzzy controller outputs.

Generally, however, choice of the inputs also involves issues related to the learning dynamics of the FMRLC. As the FMRLC operates, the learning mechanism will tune the fuzzy controller's output membership functions. In particular, in our example, for each different combination of $e(kT)$ and $c(kT)$ inputs, it will try to learn

what the best control actions are. In general, there is a close connection between what inputs are provided to the controller and the controller's ability to learn to control the plant for different reference inputs and plant operating conditions. We would like to be able to design the FMRLC so that it will learn and remember different fuzzy controllers for all the different plant operating conditions and reference inputs; hence, the fuzzy controller needs information about these. Often, however, we cannot measure the operating condition of the plant, so the FMRLC does not know exactly what operating condition it is learning the controller for. Moreover, it then does not know exactly when it has returned to an operating condition. Clearly, then, if the fuzzy controller has better information about the plant's operating conditions, the FMRLC will be able to learn and apply better control actions. If it does not have good information, it will continually adapt, but it will not properly remember.

For instance, for some plants $e(kT)$ and $c(kT)$ may only grossly characterize the operating conditions of the plant. In this situation the FMRLC is not able to learn different controllers for different operating conditions; it will use its limited information about the operating condition and continually adapt to search for the best controller. It degrades from a learning system to an adaptive system that will not properly remember the control actions (this is not to imply, however, that there will automatically be a corresponding degradation in performance).

Generally, we think of the inputs to the fuzzy controller as specifying what conditions we need to learn different controllers for. This should be one guideline used for the choice of the fuzzy controller inputs for practical applications. A competing objective is, however, to keep the number of fuzzy controller inputs low due to concerns about computational complexity. In fact, to help with computational complexity, we will sometimes use multiple fuzzy controllers with fewer inputs to each of them rather than one fuzzy controller with many inputs; then we may, for instance, sum the outputs of the individual controllers.

6.2.2 The Reference Model

Next, you must decide what to choose for the reference model that quantifies the desired performance. Basically, you want to specify a desirable performance, but also a reasonable one. If you ask for too much, the controller will not be able to deliver it; certain characteristics of real-world plants place practical constraints on what performance can be achieved. It is not always easy to pick a good reference model since it is sometimes hard to know what level of performance we can expect, or because we have no idea how to characterize the performance for some of the plant output variables (see the flexible robot application in Section 6.3 where it is difficult to know a priori how the acceleration profiles of the links should behave).

In general, the reference model may be discrete or continuous time, linear or nonlinear, time-invariant or time-varying, and so on. For example, suppose that we would like to have the response track the continuous time model

$$G(s) = \frac{1}{s+1}$$

Suppose that for your discrete-time implementation you use $T = 0.1$ sec. Using a bilinear (Tustin) transformation to find the discrete equivalent to the continuous-time transfer function $G(s)$, we replace s with $\frac{2}{T}\frac{z-1}{z+1}$ to obtain

$$\frac{Y_m(z)}{R(z)} = H(z) = \frac{\frac{1}{21}(z+1)}{z - \frac{19}{21}}$$

where $Y_m(z)$ and $R(z)$ are the z-transform of $y_m(kT)$ and $r(kT)$, respectively. Now, for a discrete-time implementation we would choose

$$y_m(kT+T) = \frac{19}{21}y_m(kT) + \frac{1}{21}r(kT+T) + \frac{1}{21}r(kT)$$

This choice would then represent that we would like our output $y(kT)$ to track a smooth, stable, first-order type response of $y_m(kT)$. A similar approach can be used to, for example, track a second-order system with a specified damping ratio ζ and undamped natural frequency ω_n.

The performance of the overall system is computed with respect to the reference model by the learning mechanism by generating an error signal

$$y_e(kT) = y_m(kT) - y(kT)$$

Given that the reference model characterizes design criteria such as rise-time and overshoot and the input to the reference model is the reference input $r(kT)$, the desired performance of the controlled process is met if the learning mechanism forces $y_e(kT)$ to remain very small for all time no matter what the reference input is or what plant parameter variations occur. Hence, the error $y_e(kT)$ provides a characterization of the extent to which the desired performance is met at time kT. If the performance is met (i.e., $y_e(kT)$ is small), then the learning mechanism will not make significant modifications to the fuzzy controller. On the other hand if $y_e(kT)$ is big, the desired performance is not achieved and the learning mechanism must adjust the fuzzy controller. Next, we describe the operation of the learning mechanism.

6.2.3 The Learning Mechanism

The learning mechanism tunes the rule-base of the direct fuzzy controller so that the closed-loop system behaves like the reference model. These rule-base modifications are made by observing data from the controlled process, the reference model, and the fuzzy controller. The learning mechanism consists of two parts: a "fuzzy inverse model" and a "knowledge-base modifier." The fuzzy inverse model performs the function of mapping $y_e(kT)$ (representing the deviation from the desired behavior), to changes in the process inputs $p(kT)$ that are necessary to force $y_e(kT)$ to zero. The knowledge-base modifier performs the function of modifying the fuzzy controller's rule-base to affect the needed changes in the process inputs. We explain each of these components in detail next.

Fuzzy Inverse Model

Using the fact that most often a control engineer will know how to roughly characterize the inverse model of the plant (examples of how to do this will be given in several examples in this chapter), we use a fuzzy system to map $y_e(kT)$, and possibly functions of $y_e(kT)$ such as $y_c(kT) = \frac{1}{T}(y_e(kT) - y_e(kT - T))$ (or any other closed-loop system data), to the necessary changes in the process inputs $p(kT)$. This fuzzy system is sometimes called the "fuzzy inverse model" since information about the plant inverse dynamics is used in its specification. Some, however, avoid this terminology and simply view the fuzzy system in the adaptation loop in Figure 6.3 to be a controller that tries to pick $p(kT)$ to reduce the error $y_e(kT)$. This is the view taken for some of the design and implementation case studies in the next section.

Note that similar to the fuzzy controller, the fuzzy inverse model shown in Figure 6.3 contains scaling gains, but now we denote them with g_{y_e}, g_{y_c}, and g_p. We will explain how to choose these scaling gains below. Given that $g_{y_e} y_e$ and $g_{y_c} y_c$ are inputs to the fuzzy inverse model, the rule-base for the fuzzy inverse model contains rules of the form

$$\textbf{If } \tilde{y}_e \text{ is } \tilde{Y}_e^j \textbf{ and } \tilde{y}_c \text{ is } \tilde{Y}_c^l \textbf{ Then } \tilde{p} \text{ is } \tilde{P}^m$$

where $\tilde{Y}_e^j$ and $\tilde{Y}_c^l$ denote linguistic values and $\tilde{P}^m$ denotes the linguistic value associated with the m^{th} output fuzzy set. In this book we often utilize membership functions for the input universes of discourse as shown in Figure 6.4, symmetric triangular-shaped membership functions for the output universes of discourse, minimum to represent the premise and implication, and COG defuzzification. Other choices can work equally well. For instance, we could make the same choices, except use singleton output membership functions and center-average defuzzification.

Knowledge-Base Modifier

Given the information about the necessary changes in the input, which are represented by $p(kT)$, to force the error y_e to zero, the knowledge-base modifier changes the rule-base of the fuzzy controller so that the previously applied control action will be modified by the amount $p(kT)$. Consider the previously computed control action $u(kT - T)$, and assume that it contributed to the present good or bad system performance (i.e., it resulted in the value of $y(kT)$ such that it did not match $y_m(kT)$). Hence, for illustration purposes we are assuming that in one step the plant input can affect the plant output; in Section 6.2.4 we will explain what to do if it takes d steps for the plant input to affect the plant output. Note that $e(kT - T)$ and $c(kT - T)$ would have been the error and change in error that were input to the fuzzy controller at that time. By modifying the fuzzy controller's knowledge-base, we may force the fuzzy controller to produce a desired output $u(kT - T) + p(kT)$, which *we should have put in at time* $kT - T$ *to make* $y_e(kT)$ *smaller*. Then, the next time we get similar values for the error and change in error, the input to the plant will be one that will reduce the error between the reference model and plant output.

Assume that we use symmetric output membership functions for the fuzzy controller, and let b_m denote the center of the membership function associated with $\tilde{U}^m$. Knowledge-base modification is performed by shifting centers b_m of the membership functions of the output linguistic value $\tilde{U}^m$ that are associated with the fuzzy controller rules that contributed to the previous control action $u(kT-T)$. This is a two-step process:

1. Find all the rules in the fuzzy controller whose premise certainty

$$\mu_i(e(kT-T), c(kT-T)) > 0 \tag{6.1}$$

   and call this the "active set" of rules at time $kT-T$. We can characterize the active set by the indices of the input membership functions of each rule that is on (since we use all possible combinations of rules, there will be one output membership function for each possible rule that is on).

2. Let $b_m(kT)$ denote the center of the m^{th} output membership function at time kT. For all rules in the active set, use

$$b_m(kT) = b_m(kT-T) + p(kT) \tag{6.2}$$

   to modify the output membership function centers. Rules that are not in the active set do not have their output membership functions modified.

Notice that for our development, when COG is used, this update will guarantee that the previous input would have been $u(kT-T)+p(kT)$ for the same $e(kT-T)$ and $c(kT-T)$ (to see this, simply analyze the formula for COG to see that adding the amount $p(kT)$ to the centers of the rules that were on will make the output shift by $p(kT)$). For the case where the fuzzy controller has input membership functions of the form shown in Figure 6.4, there will only be at most four rules in the active set at any one time instant (i.e., four rules with $\mu_i(e(kT-T), c(kT-T)) > 0$ at time kT). Then we only need to update at most four output membership function centers via Equation (6.2).

Example

As an example of the knowledge-base modification procedure, assume that all the scaling gains for both the fuzzy controller and the fuzzy inverse model are one. Suppose that the fuzzy inverse model produces an output $p(kT) = 0.5$, indicating that the value of the output to the plant at time $kT-T$ should have been $u(kT-T)+0.5$ to improve performance (i.e., to force $y_e \approx 0$). Next, suppose that $e(kT-T) = 0.75$ and $c(kT-T) = -0.2$ and that the membership functions for the inputs to the fuzzy controller are given in Figure 6.4. Then rules

$$\mathcal{R}_1\text{: } \textbf{If } E^3 \textbf{ and } C^{-1} \textbf{ Then } U^1$$

and

$$\mathcal{R}_2\text{: } \textbf{If } E^4 \textbf{ and } C^{-1} \textbf{ Then } U^2$$

are the only rules that are in the active set (notice that we chose to use the indices for the rule "1" and "2" simply for convenience). In particular, from Figure 6.4 we have $\mu_1 = 0.25$ and $\mu_2 = 0.75$, so rules $\mathcal{R}_1$ and $\mathcal{R}_2$ are the only ones that have their consequent fuzzy sets (U^1, U^2) modified. Suppose that at time $kT-T$ we had $b_1(kT-T) = 1$ and $b_2(kT-T) = 3$. To modify these fuzzy sets we simply shift their centers according to Equation (6.2) to get

$$b_1(kT) = b_1(kT-T) + p(kT) = 1 + 0.5 = 1.5$$

and

$$b_2(kT) = b_2(kT-T) + p(kT) = 3 + 0.5 = 3.5$$

Learning, Memorization, and Inverse Model Input Choice

Notice that the changes made to the rule-base are only *local* ones. That is, the entire rule-base is not updated at every time step, just the rules that needed to be updated to force $y_e(kT)$ to zero. Notice that this local learning is important since it allows the changes that were made in the past to be remembered by the fuzzy controller. Recall that the type and amount of memory depends critically on the inputs to the fuzzy controller. Different parts of the rule-base are "filled in" based on different operating conditions for the system (as characterized by the fuzzy controller inputs), and when one area of the rule-base is updated, other rules are not affected. Hence, if the appropriate inputs are provided to the fuzzy controller so that it can distinguish between the situations in which it should behave differently, the controller adapts to new situations and also remembers how it has adapted to past situations.

Just as the choice of inputs to the fuzzy controller has a fundamental impact on learning and memorization, so does the choice of inputs to the inverse model. For instance, you may want to choose the inputs to the inverse model so that it will adapt differently in different operating conditions. In one operating condition we may want to adapt more slowly than in another. In some operating condition the direction of adjustment of the output membership function centers may be the opposite of that in another. If there are multiple fuzzy controllers, you may want multiple inverse models to adjust them. This can sometimes help with computational complexity since we could then be using fewer inputs per fuzzy inverse model.

The choice of inputs to the fuzzy inverse model shown in Figure 6.3 indicates that we want to adapt differently for different errors and error rates between the reference model and plant output. The inverse model may be designed so that, for example, if the error is small, then the adjustments to the fuzzy controller should be small, and if the error is small but the rate of error increase is high, then the adjustments should be larger. It is rules such as these that are loaded into the fuzzy inverse model.

6.2.4 Alternative Knowledge-Base Modifiers

Recall that we had assumed that the plant input $u(kT)$ would affect the plant output in one time step so that $y(kT+T)$ would be affected by $u(kT)$. To remove this assumption and hence generalize the approach, let d denote the number of time steps that it takes for an input to the plant $u(kT)$ to first affect its output. That is, $y(kT+dT)$ is affected by $u(kT)$. To handle this case, we use the same approach but we go back d steps to modify the rules. Hence, we use

$$\mu_i(e(kT-dT), c(kT-dT)) > 0 \tag{6.3}$$

to form the "active set" of rules at time $kT-dT$. To update the rules in the active set, we let

$$b_m(kT) = b_m(kT-dT) + p(kT) \tag{6.4}$$

(when $d=1$, we get the case in Equations (6.1) and (6.2)). This ensures that we modify the rules that actually contributed to the current output $y(kT)$ that resulted in the performance characterization $y_e(kT)$. For applications we have found that we can most often perform a simple experiment with the plant to find d (e.g., put a short-duration pulse into the plant and determine how long it takes for the input to affect the output), and with this choice we can often design a very effective FMRLC. However, this has not always been the case. Sometimes we need to treat d as a tuning parameter for the knowledge-base modifier.

There are several alternatives to how the basic knowledge-base modification procedure can work that can be used in conjunction with the d-step back approach. For instance, note that an alternative to Equation (6.1) would be to include rules in the active set that have

$$\mu_i(e(kT-dT), c(kT-dT)) > \alpha$$

where $0 \leq \alpha < 1$. In this case we will not modify rules whose premise certainty is below some given threshold α. This makes some intuitive sense since we will then not modify rules if the fuzzy system is not too sure that they should be on. However, one could argue that any rule that contributed to the computation of $u(kT-dT)$ should be modified. This approach may be needed if you choose to use Gaussian membership functions for the input universes of discourse since it will ensure that you will not have to modify all the output centers at each time step, and hence the local learning characteristic is maintained.

There are also alternatives to the center update procedure given in Equation (6.2). For instance, we could choose

$$b_m(kT) = b_m(kT-dT) + \mu_m(e(kT-dT), c(kT-dT))p(kT)$$

so that we scale the amount we shift the membership functions by the μ_m certainty of their premises. Intuitively, this makes sense since we will then change the membership functions from rules that were on more by larger amounts, and for rules

that are not on as much we will not modify them as much. This approach has proven to be more effective than the one in Equation (6.2) for some applications; however, it is difficult to determine a priori which approach to use. We usually try the scaled approach if the one in Equation (6.2) does not seem to work well, particularly if there are some unwanted oscillations in the system that seem to result from excessive modification of output membership function center positions.

Another modification to the center update law is also necessary in some practical applications to ensure that the centers stay in some prespecified range. For instance, you may want the centers to always be positive so that the controller will never provide a negative output. Other times you may want the centers no larger than some prespecified value to ensure that the control output will become no larger than this value. In general, suppose that we know a priori that the centers should be in the range $[b_{min}, b_{max}]$ where b_{min} and b_{max} are given scalars. We can modify the output center update rule to ensure that if the centers start in this range they will stay in the range by adding the following two rules after the update formula:

$$\textbf{If } b_m(kT) < b_{min} \textbf{ Then } b_m(kT) = b_{min}$$

$$\textbf{If } b_m(kT) > b_{max} \textbf{ Then } b_m(kT) = b_{max}.$$

In other words, if the centers jump over the boundaries, they are set equal to the boundary values.

Notice that you could combine the above alternatives to knowledge-base modification so that we set a threshold for including rules in the active set, scale the updates to the centers, bound the updates to the centers, and use any number of time steps back to form the active set. There are yet other alternatives that can be used for knowledge-base modification procedures. For instance, parts of the rule-base could be left intact (i.e., we would not let them be modified). This can be useful when we know part of the fuzzy controller that is to be learned, we embed this part into the fuzzy controller that is tuned, and do not let the learning mechanism change it. Such an approach is used for the vibration damping problem for the two-link flexible robot in Section 6.3. As another alternative, when a center is updated, you could always wait d or more steps before updating the center again. This can be useful as a more "cautious" update procedure. It updates, then waits to see if the update was sufficient to correct the error y_e before it updates again. We have successfully used this approach to avoid inducing oscillations when operating at a set-point.

6.2.5 Design Guidelines for the Fuzzy Inverse Model

In this section we provide some design guidelines for the fuzzy inverse model and its scaling gains. The choice of a particular fuzzy inverse model is application-dependent, so we will use the case studies in Section 6.3 to show how it is chosen. There are, however, some general guidelines for the the choice of the fuzzy inverse model that we will outline here.

First, we note that for a variety of applications we find that the specification of the fuzzy inverse model is not much more difficult than the specification of a direct fuzzy controller. In fact, the fuzzy inverse model often takes on a form that is quite similar to a direct fuzzy controller. For instance, as you will see in the case studies in Section 6.3, the rule-base often has some typical symmetry properties.

Second, for some practical applications it is necessary to define the inverse model so that when the response of the plant is following the output of the reference model very closely, the fuzzy inverse model turns off the adaptation (such an approach is used in the aircraft application in Section 6.3). In this way once the inputs to the fuzzy inverse model get close to zero, the output of the fuzzy inverse model becomes zero. We think of this as forcing the fuzzy inverse model to be satisfied with the response as long as it is quite close to the reference model; there is no need to make it exact in many applications. Designing this characteristic into the fuzzy inverse model can sometimes help ensure stability of the overall closed-loop system. Another way to implement such a strategy is to directly modify the output of the fuzzy inverse model by using the rule:

$$\textbf{If } |p(kT)| < \epsilon_p \textbf{ Then } p(kT) = 0$$

where $\epsilon_p > 0$ is a small number that is specified a priori. For typical fuzzy inverse model designs (i.e., ones where the size of the output of the fuzzy inverse model is directly proportional to the size of the inputs to the fuzzy inverse model), this rule will make sure that when the inputs to the fuzzy inverse model are in a region of zero, its output will be modified to zero. Hence, for small fuzzy inverse model inputs the learning mechanism will turn off. If, however, the error between the plant output and the reference input grows, then the learning mechanism will turn back on and it will try to reduce the error. Such approaches to modifying the adaptation on-line are related to "robustification" methods in conventional adaptive control.

Next, we provide general tuning procedures for the scaling gains of a given fuzzy inverse model. For the sake of discussion, assume that both the fuzzy controller and fuzzy inverse model are normalized so that their input and output effective universes of discourse are all $[-1, 1]$.

Fuzzy Inverse Model Design Procedure 1

Generally, we have found the following procedure to be useful for tuning the scaling gains of the inverse model:

1. Select the gain g_{y_e} so that $y_e(kT)$ will not saturate the input membership function certainty (near the endpoints). This is a heuristic choice since we cannot know a priori how big $y_e(kT)$ will get; however, we have found that for many applications intuition about the process can be quite useful in determining the maximum value.

2. Choose the gain g_p to be the same as for the fuzzy controller output gain g_u. Let $g_{y_c} = 0$.

3. Apply a step reference input $r(kT)$ that is of a magnitude that may be typical during normal operation.

4. Observe the plant and reference model responses. There are three cases:

   (a) If there exist unacceptable oscillations in the plant output response about the reference model response, then increase g_{y_c} (we need additional derivative action in the learning mechanism to reduce the oscillations). Go to step 3.

   (b) If the plant output is unable to "keep up" with the reference model response, then decrease g_{y_c}. Go to step 3.

   (c) If the plant response is acceptable with respect to the reference model response, then the controller design is completed.

We will use this gain selection procedure for the ship steering application in Section 6.3.

Fuzzy Inverse Model Design Procedure 2

For a variety of applications, the above gain selection procedure has proven to be very successful. For other applications, however, it has been better to use an alternative procedure. In this procedure you pick the fuzzy controller and inverse model using intuition, then focus on tuning the scaling gain g_p, which we will call the "adaptation gain" using an analogy with conventional adaptive controllers. The procedure is as follows:

1. Begin with $g_p = 0$ (i.e., with the adaptation mechanism turned off) and simulate the system. With a well-designed direct fuzzy controller you should get a reasonable response, but if there is good reason to have adaptive control you will find that the performance is not what you specified in the reference model (at least for some plant conditions).

2. Choose the gains of the inverse model so that there is no saturation on its input universes of discourse.

3. Increase g_p slightly so that you just turn on the learning mechanism and it makes only small changes to the rule-base at each step. For small g_p you will allow only very small updates to the fuzzy controller so that the learning rate (adaptation rate) will be very slow. Perform any necessary tuning for the inverse model.

4. Continue to increase g_p and subsequently tune the inverse model as needed. With g_p large, you increase the adaptation rate, and hence if you increase it too much, you can get undesirable oscillations and sometimes instability. You should experiment and then choose an adaptation rate that is large enough to make it so that the FMRLC can quickly adapt to changes in the plant, yet slow enough so that it does not cause oscillations and instability.

This design approach is what we will use for the fault-tolerant aircraft control problem in Section 6.3, and it is one that we have successfully used for several other FMRLC applications.

6.3 FMRLC: Design and Implementation Case Studies

The FMRLC has been used in simulation studies for a variety of applications, including an inverted pendulum (translational for swing-up and balancing, and rotational for balancing); rocket velocity control; a rigid two-link robot; fault-tolerant control for aircraft; ship steering; longitudinal and lateral control for automated highway systems; antilock brake systems; base braking control; temperature, pressure, and level control in a glass furnace; and others. It has been implemented for balancing the rotational inverted pendulum, a ball-on-a-beam experiment, a liquid level control problem, a single-link flexible robot, the two-link flexible robot, and an induction machine. See the references at the end of the chapter for more details.

In this section we will study the cargo ship and fault-tolerant aircraft control problems in simulation and provide implementation results for the two-link flexible robot that was studied in Chapter 3. The cargo ship application helps to illustrate all the steps in how to design an FMRLC. Moreover, we design two conventional adaptive controllers and compare their performance to that of the FMRLC. The fault-tolerant aircraft control problem helps to illustrate issues in fuzzy controller initialization and how to design a more complex fuzzy inverse model by viewing it as a controller in the adaptation loop. The two-link flexible link robot application is used to show how the FMRLC can automatically synthesize a direct fuzzy controller; you will want to compare its performance with the one that we manually constructed in Chapter 3. It also illustrates how to develop and implement an FMRLC that can tune the fuzzy controller to compensate for changes in a MIMO plant—in this case, payload variations.

6.3.1 Cargo Ship Steering

To improve fuel efficiency and reduce wear on ship components, autopilot systems have been developed and implemented for controlling the directional heading of ships. Often, the autopilots utilize simple control schemes such as PID control. However, the capability for manual adjustments of the parameters of the controller is added to compensate for disturbances acting upon the ship such as wind and currents. Once suitable controller parameters are found manually, the controller will generally work well for small variations in the operating conditions. For large variations, however, the parameters of the autopilot must be continually modified. Such continual adjustments are necessary because the dynamics of a ship vary with, for example, speed, trim, and loading. Also, it is useful to change the autopilot control law parameters when the ship is exposed to large disturbances resulting from changes in the wind, waves, current, and water depth. Manual adjustment of the controller parameters is often a burden on the crew. Moreover, poor adjustment

may result from human error. As a result, it is of great interest to have a method for automatically adjusting or modifying the underlying controller.

Ship Model

Generally, ship dynamics are obtained by applying Newton's laws of motion to the ship. For very large ships, the motion in the vertical plane may be neglected since the "bobbing" or "bouncing" effects of the ship are small for large vessels. The motion of the ship is generally described by a coordinate system that is fixed to the ship [11, 149]. See Figure 6.5.

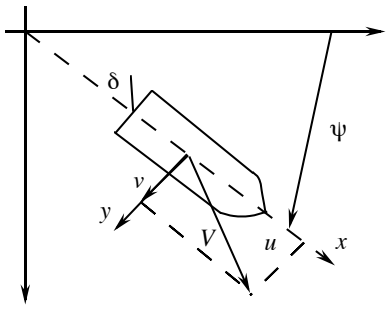


FIGURE 6.5 Cargo ship.

A simple model of the ship's motion is given by

$$\dddot{\psi}(t) + \left(\frac{1}{\tau_1} + \frac{1}{\tau_2}\right)\ddot{\psi}(t) + \left(\frac{1}{\tau_1\,\tau_2}\right)\dot{\psi}(t) = \frac{K}{\tau_1\,\tau_2}\left(\tau_3\,\dot{\delta}(t) + \delta(t)\right) \tag{6.5}$$

where ψ is the heading of the ship and δ is the rudder angle. Assuming zero initial conditions, we can write Equation (6.5) as

$$\frac{\psi(s)}{\delta(s)} = \frac{K(s\tau_3 + 1)}{s(s\tau_1 + 1)(s\tau_2 + 1)} \tag{6.6}$$

where 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,

$$K = K_0\ \left(\frac{u}{l}\right)$$
$$\tau_i = \tau_{i0}\ \left(\frac{l}{u}\right) \qquad i = 1, 2, 3$$

where we assume that for a cargo ship $K_0 = -3.86$, $\tau_{10} = 5.66$, $\tau_{20} = 0.38$, $\tau_{30} = 0.89$, and $l = 161$ meters [11]. Also, we will assume that the ship is traveling in the x direction at a velocity of 5 m/s.

In normal steering, a ship often makes only small deviations from a straight-line path. Therefore, the model in Equation (6.5) is obtained by linearizing the equations of motion around the zero rudder angle ($\delta = 0$). As a result, the rudder angle should not exceed approximately 5 degrees, otherwise the model will be inaccurate. For our purposes, we need a model suited for rudder angles that are larger than 5 degrees; hence, we use the model proposed in [20]. This extended model is given by

$$\dddot{\psi}\,(t) + \left(\frac{1}{\tau_1} + \frac{1}{\tau_2}\right)\ddot{\psi}(t) + \left(\frac{1}{\tau_1\ \tau_2}\right)H(\dot{\psi}(t)) = \frac{K}{\tau_1\ \tau_2}\left(\tau_3\ \dot{\delta}(t) + \delta(t)\right) \tag{6.7}$$

where $H(\dot{\psi})$ is a nonlinear function of $\dot{\psi}(t)$. The function $H(\dot{\psi})$ can be found from the relationship between δ and $\dot{\psi}$ in steady state such that $\dddot{\psi} = \ddot{\psi} = \dot{\delta} = 0$. An experiment known as the "spiral test" has shown that $H(\dot{\psi})$ can be approximated by

$$H(\dot{\psi}) = \bar{a}\dot{\psi}^3 + \bar{b}\dot{\psi}$$

where $\bar{a}$ and $\bar{b}$ are real-valued constants such that $\bar{a}$ is always positive. For our simulations, we choose the values of both $\bar{a}$ and $\bar{b}$ to be one.

Simulating the Ship

When we evaluate our controllers, we will use the nonlinear model in simulation. Note that to do this we need to convert the n^{th}-order nonlinear ordinary differential equations representing the ship to n first-order ordinary differential equations; for convenience, let

$$a = \left(\frac{1}{\tau_1} + \frac{1}{\tau_2}\right)$$

$$b = \left(\frac{1}{\tau_1\tau_2}\right)$$

$$c = \frac{K\tau_3}{\tau_1\tau_2}$$

and

$$d = \frac{K}{\tau_1\tau_2}$$

(this notation is not to be confused with the d-step delay of Section 6.2.4). We would like the model in the form

$$\dot{x}(t) = F(x(t), \delta(t))$$

$$y(t) = G(x(t), \delta(t))$$

where $x(t) = [x_1(t), x_2(t), x_3(t)]^\top$ and $F = [F_1, F_2, F_3]^\top$ for use in a nonlinear simulation program. We need to choose $\dot{x}_i$ so that F_i depends only on x_i and δ for $i = 1, 2, 3$. We have

$$\dddot{\psi}(t) = -a\ddot{\psi}(t) - bH(\dot{\psi}(t)) + c\dot{\delta}(t) + d\delta(t) \tag{6.8}$$

Choose

$$\dot{x}_3(t) = \dddot{\psi}(t) - c\dot{\delta}(t)$$

so that F_3 will not depend on $c\dot{\delta}(t)$ and

$$x_3(t) = \ddot{\psi}(t) - c\delta(t)$$

Choose $\dot{x}_2(t) = \ddot{\psi}(t)$ so that $x_2(t) = \dot{\psi}(t)$. Finally, choose $x_1(t) = \psi$. This gives us

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t) = F_1(x(t), \delta(t)) \\
\dot{x}_2(t) &= x_3(t) + c\delta(t) = F_2(x(t), \delta(t)) \\
\dot{x}_3(t) &= -a\ddot{\psi}(t) - bH(\dot{\psi}(t)) + d\delta(t)
\end{aligned}$$

But, $\ddot{\psi}(t) = x_3(t) + c\delta(t)$, $\dot{\psi}(t) = x_2(t)$, and $H(x_2) = x_2^3(t) + x_2(t)$ so

$$\dot{x}_3(t) = -a\left(x_3(t) + c\delta(t)\right) - b\left(x_2^3(t) + x_2(t)\right) + d\delta(t) = F_3(x(t), \delta(t))$$

This provides the proper equations for the simulation. Next, suppose that the initial conditions are $\psi(0) = \dot{\psi}(0) = \ddot{\psi}(0) = 0$. This implies that $x_1(0) = x_2(0) = 0$ and $x_3(0) = \ddot{\psi}(0) - c\delta(0)$ or $x_3(0) = -c\delta(0)$. For a discrete-time implementation, we simply discretize the differential equations.

FMRLC Design

In this section we explain how to design an FMRLC for controlling the directional heading of the cargo ship. The inputs to the fuzzy controller are the heading error and change in heading error expressed as

$$e(kT) = \psi_r(kT) - \psi(kT)$$

and

$$c(kT) = \frac{e(kT) - e(kT - T)}{T}$$

respectively, where $\psi_r(kT)$ is the desired ship heading ($T = 50$ milliseconds). The controller output is the rudder angle $\delta(kT)$ of the ship. For our fuzzy controller design, 11 uniformly spaced triangular membership functions are defined for each controller input, as shown in Figure 6.4 on page 322.

The scaling controller gains for the error, change in error, and the controller output are chosen via the design procedure to be $g_e = \frac{1}{\pi}$ (since the error $e(kT)$ 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/sec), and $g_u = \frac{8\pi}{18}$ (since we want to limit δ between ±80°, we have $g_u = \frac{80\pi}{180} = \frac{8\pi}{18}$). The fuzzy sets for the fuzzy controller output are assumed to be symmetric and triangular-shaped with a base width of 0.4, and all centered at zero on the normalized universe of discourse (i.e., 121 output membership functions all centered at zero).

The reference model was chosen so as to represent somewhat realistic performance requirements as

$$\ddot{\psi}_m(t) + 0.1\ \dot{\psi}_m(t) + 0.0025\ \psi_m(t) = 0.0025\ \psi_r(t)$$

where $\psi_m(t)$ specifies the desired system performance for the ship heading $\psi(t)$.

The input to the fuzzy inverse model includes the error and change in error between the reference model and the ship heading expressed as

$$\psi_e(kT) = \psi_m(kT) - \psi(kT)$$

and

$$\psi_c(kT) = \frac{\psi_e(kT) - \psi_e(kT - T)}{T}$$

respectively. For each of these inputs, 11 symmetric and triangular-shaped membership functions are defined that are evenly distributed on the appropriate universes of discourse (the same as shown in Figure 6.4 on page 322). The normalizing controller gains associated with $\psi_e(kT)$, $\psi_c(kT)$, and $p(kT)$ are chosen to be $g_{\psi_e} = \frac{1}{\pi}$, $g_{\psi_c} = 5$, and $g_p = \frac{8\pi}{18}$, respectively, according to design procedure 1 in Section 6.2.5.

For a cargo ship, an increase in the rudder angle $\delta(kT)$ will generally result in a *decrease* in the ship heading angle (see Figure 6.5). This is the information about the inverse dynamics of the plant that we use in the fuzzy inverse model rules. Specifically, we will use rules of the form

$$\textbf{If } \tilde{\psi}_e \text{ is } \tilde{\Psi}_e^i \textbf{ and } \tilde{\psi}_c \text{ is } \tilde{\Psi}_c^j \textbf{ Then } \tilde{p} \text{ is } \tilde{P}^m$$

Suppose that we name the center of the output membership function for this rule $c_{i,j}$ to emphasize that it is the center associated with the output membership function that has the i^{th} membership function for the $\tilde{\psi}_e$ universe of discourse and the j^{th} membership function for the $\tilde{\psi}_c$ universe of discourse. The rule-base array shown in Table 6.1 is employed for the fuzzy inverse model for the cargo ship. In Table 6.1, Ψ_e^i denotes the i^{th} fuzzy set associated with the error signal ψ_e, and Ψ_c^j denotes the j^{th} fuzzy set associated with the change in error signal ψ_c. The entries of the table represent the center values of symmetric triangular-shaped membership functions $c_{i,j}$ with base widths 0.4 for output fuzzy sets P^m on the normalized universe of discourse.

TABLE 6.1 Knowledge-Base Array Table for the Cargo Ship Fuzzy Inverse Model

$c_{i,j}$		Ψ_c^j										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
Ψ_e^i	-5	1	1	1	1	1	1	.8	.6	.4	.2	0
	-4	1	1	1	1	1	.8	.6	.4	.2	0	$-.2$
	-3	1	1	1	1	.8	.6	.4	.2	0	$-.2$	$-.4$
	-2	1	1	1	.8	.6	.4	.2	0	$-.2$	$-.4$	$-.6$
	-1	1	1	.8	.6	.4	.2	0	$-.2$	$-.4$	$-.6$	$-.8$
	0	1	.8	.6	.4	.2	**0**	$-.2$	$-.4$	$-.6$	$-.8$	-1
	1	.8	.6	.4	.2	0	$-.2$	$-.4$	**$-.6$**	$-.8$	-1	-1
	2	.6	.4	.2	0	$-.2$	$-.4$	$-.6$	$-.8$	-1	-1	-1
	3	.4	.2	0	$-.2$	$-.4$	$-.6$	$-.8$	-1	-1	-1	-1
	4	.2	0	$-.2$	$-.4$	$-.6$	$-.8$	-1	-1	-1	-1	-1
	5	0	$-.2$	$-.4$	$-.6$	$-.8$	-1	-1	-1	-1	-1	-1

The meaning of the rules in Table 6.1 is best explained by studying the meaning of a few specific rules. For instance, if $i = j = 0$ then we see from the table that $c_{i,j} = c_{0,0} = 0$ (this is the center of the table). This cell in the table represents the rule that says "if $\psi_e = 0$ and $\psi_c = 0$ then y is tracking y_m perfectly so you should not update the fuzzy controller." Hence, the output of the fuzzy inverse model will be zero. If, on the other hand $i = 2$ and $j = 1$ then $c_{i,j} = c_{2,1} = -0.6$. This rule indicates that "if ψ_e is positive (i.e., ψ_m is greater than ψ) and ψ_c is positive (i.e., $\psi_m - \psi$ is increasing) then change the input to the fuzzy controller that is generated to produce these values of ψ_e and ψ_c by decreasing it." Basically, this is because we want ψ to increase, so we want to decrease δ to achieve this (see Figure 6.5). We see that the inverse model indicates that whatever the input was in this situation, it should have been less so it subtracts some amount (the amount affected by the scaling gain g_p). It is a good idea for you to convince yourself that other rules in Table 6.1 make sense. For instance, consider the case where $i = -2$ and $j = -4$ so that $c_{-2,-4} = 1$: explain why this rule makes sense and how it represents information about the inverse behavior of the plant.

It is interesting to note that we can often pick a form for the fuzzy inverse model that is similar *in form* to that shown in Table 6.1 (at least, the pattern of the consequent membership function centers often has this type of symmetry, or has the sequence of zeros along the other diagonal). At other times we will need to incorporate additional inputs to the fuzzy inverse model or we may need to use a nonlinear mapping for the output centers. For example, a cubic mapping of the centers is sometimes useful, so if y is close to y_m we will slow adaptation, but if they are far apart we will speed up adaptation significantly.

Gradient-Based Model Reference Adaptive Control

The controller parameter adjustment mechanism for the gradient approach to MRAC can be implemented via the "MIT rule." For this, the cost function

$$J(\underline{\theta}) = \frac{1}{2}\psi_e^2(t)$$

where $\underline{\theta}$ holds the parameters of the controller that will be tuned, $\psi_e(t) = \psi_m(t) - \psi(t)$, and

$$\frac{d\underline{\theta}}{dt} = -\gamma \ \frac{\partial J}{\partial \underline{\theta}}$$

so that

$$\frac{d\underline{\theta}}{dt} = -\gamma \ \psi_e(t) \ \frac{\partial \psi_e(t)}{\partial \underline{\theta}}$$

For developing the MIT rule for the ship we assume that the ship may be modeled by a second-order linear differential equation. This model is obtained by eliminating the process pole resulting from τ_2 in Equation (6.5) since its associated dynamics are significantly faster than those resulting from τ_1. Also, for small heading variations the rudder angle derivative $\dot{\delta}$ is likely to be small and may be neglected. Therefore, we obtain the following reduced-order model for the ship:

$$\ddot{\psi}(t) + \left(\frac{1}{\tau_1}\right)\dot{\psi}(t) = \left(\frac{K}{\tau_1}\right)\delta(t) \tag{6.9}$$

The PD-type control law that will be employed for this process may be expressed by

$$\delta(t) = k_p \ (\psi_r(t) - \psi(t)) - k_d \ \dot{\psi}(t) \tag{6.10}$$

where k_p and k_d are the proportional and derivative gains, respectively, and $\psi_r(t)$ is the desired process output. Substituting Equation (6.10) into Equation (6.9), we obtain

$$\ddot{\psi}(t) + \left(\frac{1 + K \ k_d}{\tau_1}\right)\dot{\psi}(t) + \left(\frac{K \ k_p}{\tau_1}\right)\psi(t) = \left(\frac{K \ k_p}{\tau_1}\right)\psi_r(t) \tag{6.11}$$

It follows from Equation (6.11) that

$$\psi(t) = \frac{\frac{K \ k_p}{\tau_1}}{p^2 + \left(\frac{1+K \ k_d}{\tau_1}\right) \ p + \left(\frac{K \ k_p}{\tau_1}\right)}\psi_r(t) \tag{6.12}$$

where p is the differential operator.

The reference model for this process is chosen to be

$$\psi_m(t) = \frac{\omega_n^2}{p^2 + 2\zeta\omega_n\ p + \omega_n^2}\psi_r(t) \tag{6.13}$$

where to be consistent with the FMRLC design we choose $\zeta = 1$ and $\omega_n = 0.05$. Combining Equations (6.13) and (6.12) and finding the partial derivatives with respect to the proportional gain k_p and the derivative gain k_d, we find that

$$\frac{\partial\ \psi_e}{\partial k_p} = \left(\frac{\frac{K}{\tau_1}}{p^2 + \left(\frac{1+K\ k_d}{\tau_1}\right)\ p + \left(\frac{K\ k_p}{\tau_1}\right)}\right)(\psi - \psi_r) \tag{6.14}$$

and

$$\frac{\partial\ \psi_e}{\partial k_d} = \left(\frac{\frac{K}{\tau_1}\ p}{p^2 + \left(\frac{1+K\ k_d}{\tau_1}\right)\ p + \left(\frac{K\ k_p}{\tau_1}\right)}\right)\psi \tag{6.15}$$

In general, Equations (6.14) and (6.15) cannot be used because the controller parameters k_p and k_d are not known. Observe that for the "optimal values" of k_p and k_d, we have

$$p^2 + \left(\frac{1 + K\ k_d}{\tau_1}\right)\ p + \left(\frac{K\ k_p}{\tau_1}\right) = p^2 + 2\zeta\omega_n\ p + \omega_n^2$$

Furthermore, the term $\frac{K}{\tau_1}$ may be absorbed into the adaptation gain γ. However, this requires that the sign of $\frac{K}{\tau_1}$ be known since, in general, γ should be positive to ensure that the controller updates are made in the direction of the negative gradient. For a forward-moving cargo ship the sign of $\frac{K}{\tau_1}$ happens to be negative, which implies that the term γ with $\frac{K}{\tau_1}$ absorbed into it must be negative to achieve the appropriate negative gradient. After making the above approximations, we obtain the following differential equations for updating the PD controller gains:

$$\frac{d\ k_p}{dt} = -\gamma_1\left(\frac{1}{p^2 + 2\zeta\omega_n p + \omega_n^2}\ (\psi - \psi_r)\right)\psi_e$$
$$\frac{d\ k_d}{dt} = -\gamma_2\left(\frac{p}{p^2 + 2\zeta\omega_n p + \omega_n^2}\ \psi\right)\psi_e$$

where γ_1 and γ_2 are negative real numbers. After many simulations, the best values that we could find for the γ_i are $\gamma_1 = -0.005$ and $\gamma_2 = -0.1$.

Lyapunov-Based Model Reference Adaptive Control

In this section we present a Lyapunov-based approach to MRAC that tunes a proportional derivative (PD) control law. Recall that the ship dynamics may be

approximated by a second-order linear time-invariant differential equation given by Equation (6.9). We use a PD control law

$$\delta(t) = k_p\ (\psi_r(t) - \psi(t)) - k_d\ \dot{\psi}(t)$$

where k_p and k_d are the proportional and derivative gains, respectively, and $\psi_r(t)$ is the desired process output. The dynamic equation that describes the compensated system is $\dot{\underline{\psi}} = A_c\underline{\psi} + B_c\psi_r$ where $\underline{\psi} = [\psi, \dot{\psi}]^\top$ and

$$A_c = \begin{bmatrix} 0 & 1 \\ -\frac{Kk_p}{\tau_1} & -\frac{(1+Kk_d)}{\tau_1} \end{bmatrix}$$
$$B_c = \begin{bmatrix} 0 \\ \frac{Kk_p}{\tau_1} \end{bmatrix}$$

The reference model is given by

$$\dot{\underline{\psi}}_m = A_m\underline{\psi}_m + B_m\underline{\psi}_r \tag{6.16}$$

where $\underline{\psi}_m = [\psi_m, \dot{\psi}_m]^T$ and

$$A_m = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix}$$
$$B_m = \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix}$$

and where to be consistent with the FMRLC design we choose $\zeta = 1$ and $\omega_n = 0.05$.

The differential equation that describes the error $\underline{\psi}_e(t) = \underline{\psi}_m(t) - \underline{\psi}(t)$ may be expressed by

$$\dot{\underline{\psi}}_e = A_m(t)\underline{\psi}_e + (A_m(t) - A_c(t))\underline{\psi} + (B_m(t) - B_c(t))\underline{\psi}_r \tag{6.17}$$

The equilibrium point $\underline{\psi}_e = 0$ in Equation (6.17) is asymptotically stable if we choose the adaptation laws to be

$$\dot{A}_c(t) = \gamma P\underline{\psi}_e\underline{\psi}^\top \tag{6.18}$$
$$\dot{B}_c(t) = \gamma P\underline{\psi}_e\underline{\psi}_r^\top \tag{6.19}$$

where $P \in \Re^{n\times n}$ is a symmetric, positive definite matrix that is a solution of the Lyapunov equation

$$A_m^\top P + PA_m = -Q\ < 0$$

Assuming that Q is a 2×2 identity matrix and solving for P, we find that

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} = \begin{bmatrix} 25.0125 & 200.000 \\ 200.000 & 2005.00 \end{bmatrix}$$

Solving for $\dot{k}_p$ and $\dot{k}_d$ in Equations (6.18) and (6.19), respectively, the adaptation law in Equations (6.18) and (6.19) may be implemented as

$$\dot{k}_p = -\gamma_1(p_{21}\psi_e + p_{22}\dot{\psi}_e)(\psi - \psi_r), \tag{6.20}$$
$$\dot{k}_d = -\gamma_2(p_{21}\psi_e + p_{22}\dot{\psi}_e)\dot{\psi}. \tag{6.21}$$

Equations (6.20) and (6.21) assume that the plant parameters and disturbance are varying slowly. In obtaining Equations (6.20) and (6.21), we absorbed the term $\frac{K}{\tau_1}$ into the adaptation gains γ_1 and γ_2. Recall that for the cargo ship, $\frac{K}{\tau_1}$ happens to be a negative quantity. Therefore, both γ_1 and γ_2 must be negative. We found that $\gamma_1 = -0.005$ and $\gamma_2 = -0.1$ were suitable adaptation gains.

Comparative Analysis of FMRLC and MRAC

For the simulations for both the FMRLC and the MRACs, we use the nonlinear process model given in Equation (6.7) to emulate the ship's dynamics. Figure 6.6 shows the results for the FMRLC, and we see that it was quite successful in generating the appropriate control rules for a good response since the ship heading tracks the reference model almost perfectly. In fact, the maximum deviation between the two signals was observed to be less than 1° over the entire simulation. This is the case even though initially the right-hand sides of the control rules have membership functions with centers all at zero (i.e., initially, the controller knows little about how to control the plant), so we see that the FMRLC learns very fast.

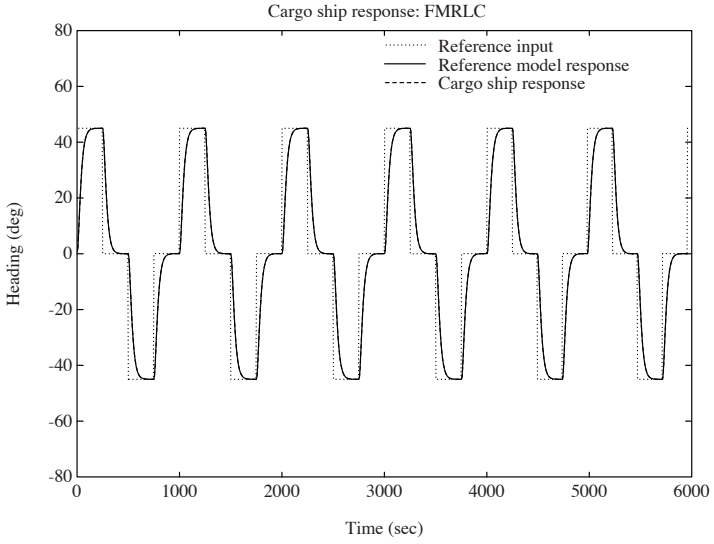


FIGURE 6.6 FMRLC simulation results (figure taken from [112], © IEEE).

Compare the results for the FMRLC with those obtained for the gradient-based and Lyapunov-based approaches to MRAC, which are shown in Figures 6.7 and 6.8. For the gradient-based and Lyapunov-based approaches, both system responses converged to track the reference model. However, the convergence rate of both algorithms was significantly slower than that of the FMRLC method (and comparable control energy was used by the FMRLC and the MRACs). The controller gains k_p and k_d for both MRACs were initially chosen to be 5. This choice of initial controller gains happens to be an unstable case for the second-order linear process model (in the case where the adaptation mechanism is disconnected). However, we felt this to be a fair comparison since the fuzzy controller is initially chosen so that it would put in zero degrees of rudder no matter what the controller input values were. We would have chosen both controller gains to be zero, but this choice resulted in a very slow convergence rate for the MRACs.

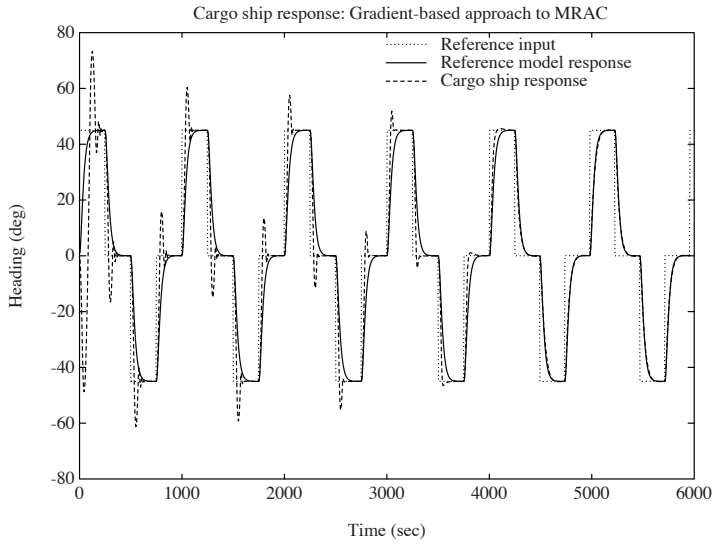


FIGURE 6.7 Gradient-based MRAC simulation results (figure taken from [112], © IEEE).

The final set of simulations for the ship was designed to illustrate the ability of the learning and adaptive controllers to compensate for disturbances at the process input. A disturbance is injected by adding it to the rudder command δ then putting this signal into the plant as the control signal. Specifically, the disturbance was chosen to be be a sinusoid with a frequency of one cycle per minute and a magnitude of 2° with a bias of 1° (see the bottom plot in Figure 6.9). The effect of this disturbance is similar to that of a gusting wind acting upon the ship.

Figure 6.9 illustrates the results obtained for this simulation. To provide an

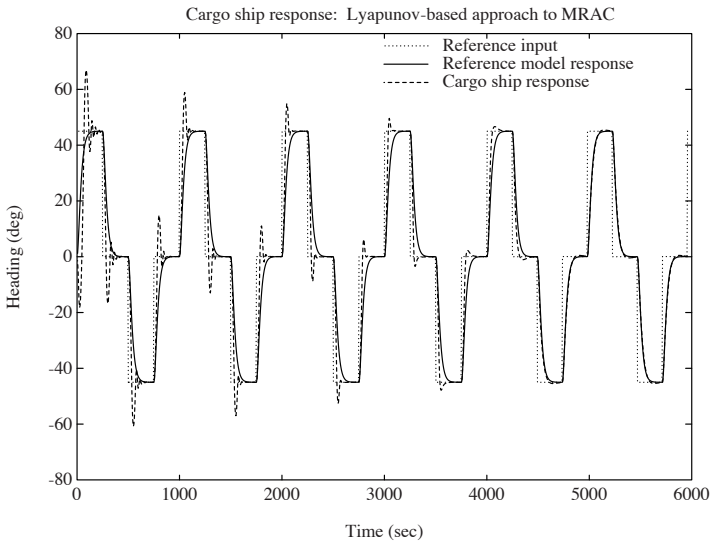


FIGURE 6.8 Lyapunov-based MRAC simulation results (figure taken from [112], © IEEE).

especially fair comparison with the FMRLC algorithm, we initially loaded the PD controllers in both MRAC algorithms with the controller gains that resulted at the end of their 6000-second simulations in Figures 6.7 and 6.8. However, the centers of the right-hand sides of the membership functions for the knowledge-base of the fuzzy controller in the FMRLC algorithm were initialized with all zeros as before (hence, we are giving the MRACs an advantage). Notice that the FMRLC algorithm was nearly able to completely cancel the effects of the disturbance input (there is still a very small magnitude oscillation). However, the gradient- and Lyapunov-based approaches to MRAC were not nearly as successful.

Discussion: A Control-Engineering Perspective

In this section we summarize and more carefully discuss the conclusions from our simulation studies. The results in the previous section seem to indicate that the FMRLC has the following advantages:

- It achieves fast convergence compared to the MRACs.
- No additional control energy is needed to achieve this faster convergence.
- It has good disturbance rejection properties compared to the MRACs.
- Its design is independent of the particular form of the mathematical model of the underlying process (whereas in the MRAC designs, we need an explicit mathematical model of a particular form)

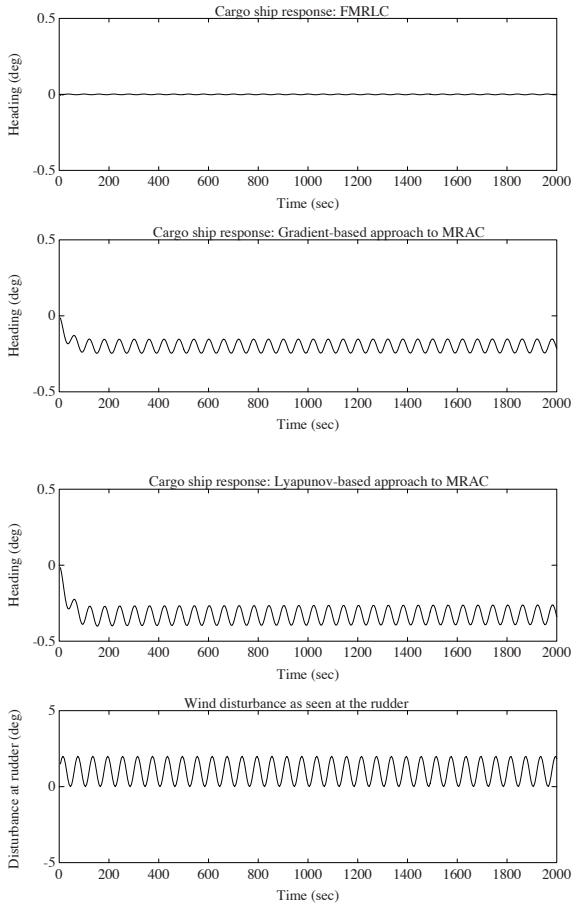


FIGURE 6.9 Simulation results comparing disturbance rejection for the FMRLC, the gradient-based approach to MRAC, and the Lyapunov-based approach to MRAC (figure taken from [112], © IEEE).

Overall, the FMRLC provides a method to synthesize (i.e., automatically design) and tune the knowledge-base for a direct fuzzy controller. As the direct fuzzy controller is a nonlinear controller, some of the above advantages may be attributed to the fact that the underlying controller that is tuned inherently has more significant functional capabilities than the PD controllers used in the MRAC designs.

While our application may indicate that FMRLC is a promising alternative to conventional MRAC, we must also emphasize the following:

- We have compared the FMRLC to only two types of MRACs, for only one appli-

cation, for a limited class of reference inputs, and only in simulation. There is a wide variety of other adaptive control approaches that also deserve consideration.

- There are no guarantees of stability or convergence; hence, we can simply pick a different reference input, and the system may then be unstable (indeed, for some applications, we have been able to destabilize the FMRLC, especially if we pick the adaptation gain g_p large).

- "Persistency of excitation"[77] is related to the learning controller's ability to always generate an appropriate plant input and to *generalize* the results of what it has learned earlier and apply this to new situations. In this context, for the ship we ask the following questions: (1) What if we need to turn the ship in a different direction? Will the rule-base be "filled in" for this direction? (2) Or will it have to learn for each new direction? (3) If it learns for the new directions, will it forget how to control for the old ones?

- In terms of control energy, we may have just gotten lucky for this application and for the chosen reference input (although with additional tests, this does not seem to be the case). There seem to be no analytical results that guarantee that the FMRLC or any other fuzzy learning control technique minimizes the use of control energy for a wide class of plants.

- This is a very limited investigation of the disturbance rejection properties (i.e., only one type of wind disturbance is considered).

- The design approach for the FMRLC, although it did not depend on a mathematical model, is somewhat ad hoc. What fundamental limitations will, for example, nonminimum phase systems present? Certainly there will be limitations for classes of nonlinear systems. What will these limitations be? It is important to note that the use of a mathematical model helps to show what these limitations will be (hence, it cannot *always* be considered an advantage that many fuzzy control techniques do *not* depend on the specification of a mathematical model). Also, note that due to our avoidance of using a mathematical model of the plant, we have also ignored the "model matching problem" in adaptive control [77].

- There may be gains in performance, but are these gains being made by paying a high price in computational complexity for the FMRLC? The FMRLC is somewhat computationally intensive (as are many neural and fuzzy learning control approaches), but we have shown implementation tricks in Chapter 2 that can significantly reduce problems with computation time. The FMRLC can, however, require significant memory since we need to store each of the centers of the output membership functions of the rules, and with increased numbers of inputs to the fuzzy controller, there is an exponential increase in the numbers of these centers (assuming you use all possible combinations of rules). The FMRLC in this section required us to store and update 121 output membership function centers.

6.3.2 Fault-Tolerant Aircraft Control

There are virtually an unlimited number of possible failures that can occur on a sophisticated modern aircraft such as the F-16 that we consider in this case study. While preplanned pilot-executed response procedures have been developed for certain anticipated failures, especially catastrophic and high-probability failures, certain unanticipated events can occur that complicate successful failure accommodation. Indeed, aircraft accident investigations sometimes find that even with some of the most severe unanticipated failures, there was a way in which the aircraft could have been saved if the pilot had taken proper actions in a timely fashion. Because the time frame during a catastrophic event is typically short, given the level of stress and confusion during these incidents, it is understandable that a pilot may not find the solution in time to save the aircraft.

With the recent advances in computing technology and control theory, it appears that the potential exists to implement a computer control strategy that can assist (or replace) the pilot in helping to mitigate the consequences of severe failures in aircraft. In this case study we will investigate the use of the FMRLC for failure accommodation; however, we must emphasize that our study is somewhat academic. The reader should be aware that there currently exists no completely satisfactory solution to the fault-tolerant aircraft control problem. Indeed, it is an important area of current research. For instance, in this case study we will only study a certain class of actuator failures where in some aircraft sensor failures are of concern. We do not compare and contrast the fuzzy control approach to other conventional approaches to fault tolerant control (e.g., conventional adaptive control approaches). We do not study stability and robustness of the resulting control system. These, and many other issues, are interesting areas for future research.

Aircraft Model

The F-16 aircraft model used in this case study is based on a set of five linear perturbation models (that are extracted from a nonlinear model at the five operating conditions[1]) (A_i, B_i, C_i, D_i), $i \in \{1, 2, 3, 4, 5\}$:

$$\begin{aligned} \dot{\underline{x}} &= A_i \underline{x} + B_i \underline{u} \\ \underline{y} &= C_i \underline{x} + D_i \underline{u} \end{aligned} \tag{6.22}$$

where the variables are defined as follows (see Figure 6.10):

- Inputs $\underline{u} = [\delta_e, \delta_{de}, \delta_a, \delta_r]^\top$:
  1. δ_e = elevator deflection (in degrees)
  2. δ_{de} = differential elevator deflection (in degrees)
  3. δ_a = aileron deflection (in degrees)
  4. δ_r = rudder deflection (in degrees)

1. All information about the F-16 aircraft models was provided by Wright Laboratories, WPAFB, OH.

- System state $\underline{x} = [\alpha, q, \phi, \beta, p, r]^\top$:
  1. α = angle of attack (in degrees)
  2. q = body axis pitch rate (in degrees/second)
  3. ϕ = Euler roll angle (in degrees)
  4. β = sideslip angle (in degrees)
  5. p = body axis roll rate (in degrees/second)
  6. r = body axis yaw rate (in degrees/second)

The output is $\underline{y} = [x^\top, A_z]^\top$ where A_z is the normal acceleration (in g).

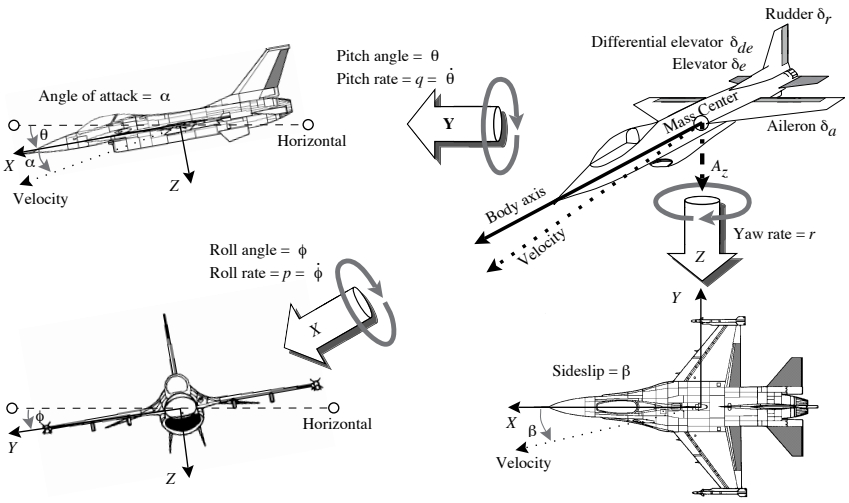


FIGURE 6.10 The F-16 aircraft (figure taken from [103], © IEEE).

Nominal Control Laws

The nominal control laws for the F-16 aircraft used in this study consist of two parts, one for the lateral channel as shown in Figure 6.11, and the other for the longitudinal channel. The inputs to the controller are the pilot commands and the F-16 system feedback signals. For the longitudinal channel, the pilot command is the desired pitch A_{zd}, and the system feedback signals are normal acceleration A_z, angle of attack α, and pitch rate q. For the lateral channel, the pilot commands are the desired roll rate p_d as well as the desired sideslip β_d, and the system feedback signals are the roll rate p, yaw angle r, and sideslip β. The controller gains for the longitudinal channel and $K(\bar{q})$ for the lateral channel in Figure 6.11 are scheduled as a function of different dynamic pressures $\bar{q}$. The dynamic pressure for all five perturbation models is fixed at 499.24 psf, which is based on an assumption that

the F-16 aircraft will operate with constant speed and altitude. For the lateral channel we have

$$K(499.24) = \begin{bmatrix} 0.47 & 0.14 & 0.14 & -0.56 & -0.38 \\ -0.08 & -0.056 & 0.78 & -1.33 & -4.46 \end{bmatrix} \tag{6.23}$$

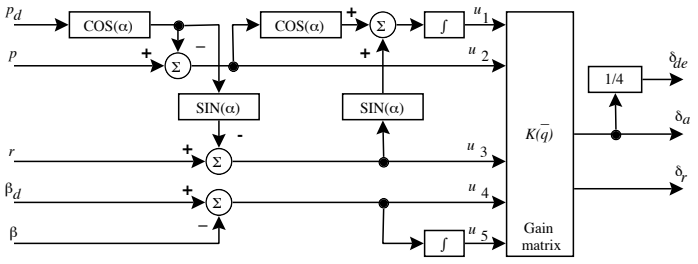


FIGURE 6.11 Nominal lateral control system (figure taken from [103], © IEEE).

The transfer function $\frac{20}{s+20}$ is used to represent the actuator dynamics for each of the aircraft control surfaces, and the actuators have physical saturation limits so that $-21^\circ \leq \delta_e \leq 21^\circ$, $-21^\circ \leq \delta_{de} \leq 21^\circ$, $-23^\circ \leq \delta_a \leq 20^\circ$, and $-30^\circ \leq \delta_r \leq 30^\circ$. The actuator rate saturation is $\pm 60^\circ$/sec for all the actuators.

To simulate the closed-loop system, we interpolate between the five perturbation models based on the value of α, which produces a nonlinear simulation of the F-16. For all the simulations, a special "loaded roll command sequence" is used. This command sequence is as follows: At time $t = 0.0$, a 60°/sec roll rate command (p_d) is held for 1 second. At time $t = 1.0$, a 3 g pitch command (A_{zd}) is held for 9 seconds. At time $t = 4.5$, a -60°/sec roll rate command (p_d) is held for 1.8 seconds. Finally, at time $t = 11.5$, a 60°/sec roll rate command (p_d) is held for 1 second. The sideslip command β_d is held at zero throughout the sequence.

Failure Scenarios

Many different failures can occur on a high-performance aircraft such as the F-16. For instance, there are two major types of actuator failures:

1. *Actuator malfunction:* Two main types are possible:
   (a) Actuator performance degradation (e.g., a bandwidth decrease).
   (b) Actuator stuck at a certain angle (e.g., an arbitrary angle during a motion, or at the maximum deflection).
2. *Actuator damage:* Again, two main types are possible:

(a) Actuator damaged so that the control surface oscillates in an uncontrollable fashion.

(b) Control surface loss due to severe structural damage.

Here we focus on actuator malfunctions for the F-16.

FMRLC for the F-16

In this section we develop a MIMO FMRLC for the fault-tolerant aircraft control problem. We use the same basic structure for the FMRLC as in Figure 6.3 on page 321 with a slightly different notation for the variables. In particular, we use underlines for vector quantities so that $\underline{y}_r(kT)$ is the vector of reference inputs, $\underline{y}_f(kT)$ is the vector of outputs from the MIMO fuzzy inverse model, $\underline{e}(kT)$ is the vector of error inputs to the fuzzy controller, $\underline{y}_e(kT)$ is the vector of error inputs to the inverse model, and $\underline{c}(kT)$ and $\underline{y}_c(kT)$ are the change-in-error vectors to the fuzzy controller and inverse model, respectively. The scaling gains are denoted as, for example, $\underline{g}_e = [g_{e_1}, \ldots, g_{e_s}]^\top$ if there are s inputs to the fuzzy controller—similarly for the other scaling gains (the gains on the inverse model output $\underline{y}_f(kT)$ will be denoted with $\underline{g}_f$) so that $g_{e_i}e_i(kT)$ is an input to the fuzzy controller. The gains $\underline{g}_e$ are chosen so that the range of values of $g_{e_i}e_i(kT)$ lies on $[-1, 1]$, and $\underline{g}_u$ is chosen by using the allowed range of inputs to the plant in a similar way. The gains $\underline{g}_c$ are determined by experimenting with various inputs to the system to determine the normal range of values that $\underline{c}(kT)$ will take on; then $\underline{g}_c$ is chosen so that this range of values is scaled to $[-1, 1]$. We utilize r MISO fuzzy controllers, one for each process input u_n (equivalent to using one MIMO controller). Each of the fuzzy controllers and fuzzy inverse models has the form explained in Section 6.2.

To begin the design of the FMRLC, it is important to try to use some intuition that we have about how to achieve fault-tolerant control. For instance, generally it is not necessary to utilize all the control effectors to compensate for the effects of the failure of a single actuator on the F-16. If the ailerons in the lateral channel fail, the differential elevators can often be used for compensation, or vice versa. However, the elevators may not aid in reconfiguration for an aileron failure unless they are specially designed to induce moments in the lateral channel. Hence, it is sufficient to redesign only part of the nominal controller to facilitate control reconfiguration. Here, we will replace the $K(\bar{q})$ portion of the lateral nominal control laws (see Figure 6.11) with a fuzzy controller and let the learning mechanism of the FMRLC tune the fuzzy controller to perform control reconfiguration for an aileron failure (see Figure 6.12).

To apply the FMRLC in the F-16 reconfigurable control application, it is of fundamental importance that for an unimpaired aircraft, the FMRLC must behave at least as good as (indeed, the same as) the nominal control laws. In normal operation, the learning mechanism is inactive or used only to maintain the aircraft performance at the level of specified reference models. In the presence of failures, where the performance becomes different from the specified reference model, the

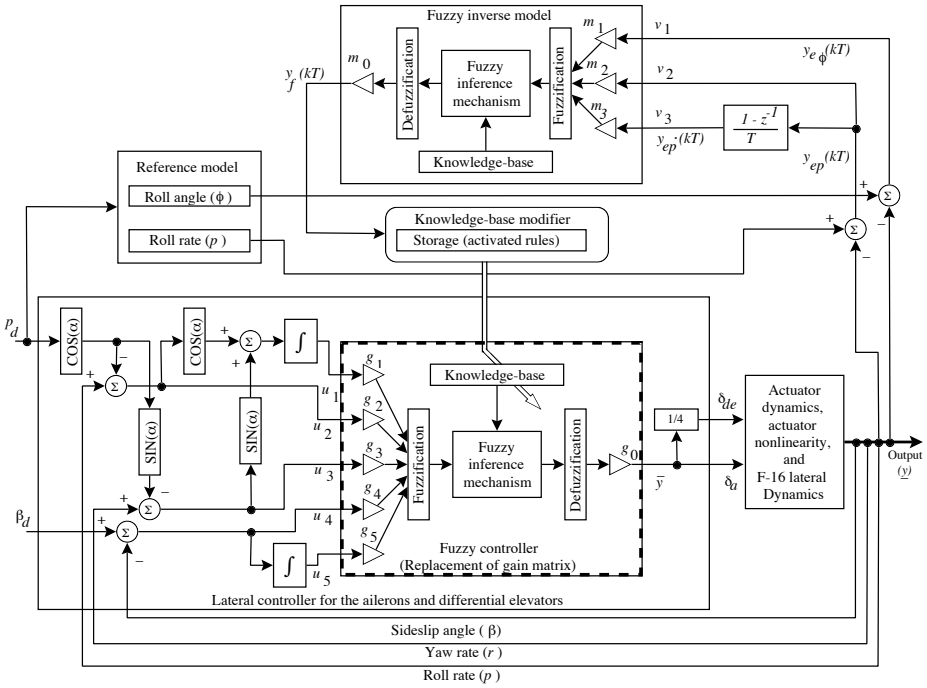


FIGURE 6.12 FMRLC for reconfigurable control in case of aileron or differential elevator failures (figure taken from [103], © IEEE).

learning mechanism can then tune the fuzzy controller to achieve controller reconfiguration. In the next section, we explain how to pick the initial fuzzy controller shown in Figure 6.12 so that it will perform the same as the nominal controller when there is no failure (the procedure is different from the initialization approach where you simply choose all output membership functions to be centered at zero). Following this we introduce the reference model and learning mechanism.

The Fuzzy/Nominal Controller

Notice that the gain matrix block $K(\bar{q})$ in Figure 6.11 is replaced by a fuzzy controller in Figure 6.12, which will be adjusted by the FMRLC to reconfigure part of the control laws in case there is a failure. Therefore, to copy the nominal control laws, all that is necessary is for the fuzzy controller to simulate the effects of the portion of the gain matrix $K(\bar{q})$ that affects the aileron and differential elevator outputs. In this way, the FMRLC is provided with the very good initial guess of the control strategies (i.e., nominal control laws resulting from years of experience of the designer). We have shown how to make a fuzzy controller form a weighted sum of its inputs for the rotational inverted pendulum in Chapter 3. A similar approach is used here to produce the fuzzy controller that approximates the gain

matrix $K(\bar{q})$. If we name the gains $g_0 - g_5$ for the five inputs to the fuzzy controller, then using the procedure from Chapter 3 we get

$$[g_0, g_1, g_2, g_3, g_4, g_5] = \left[14, \frac{1}{29.79}, \frac{1}{100}, \frac{1}{100}, -\frac{1}{25}, -\frac{1}{36.84}\right] \tag{6.24}$$

With this choice the direct fuzzy controller (i.e., with the adaptation mechanism turned off) performs similarly to the nominal control laws.

F-16 Reference Model Design

As discussed in the previous subsection, the reference model is used to characterize the closed-loop specifications such as rise-time, overshoot, and settling time. The performance of the overall system is computed with respect to the reference model by generating error signals between the reference model output and the plant outputs–that is, $y_{e\phi}(kT)$, $y_{ep}(kT)$, and $y_{e\dot{p}}(kT)$ in Figure 6.12. (Note that we use the notation $y_{e\dot{p}}$ to denote the signal that is the approximate derivative of the change in error of the roll rate p. The use of "$\dot{p}$" in the subscript does *not* denote the use of a continuous-time signal.) To achieve the desired performance, the learning mechanism must force $y_{e\phi}(kT) \approx 0$, $y_{ep}(kT) \approx 0$, and $y_{e\dot{p}}(kT) \approx 0$ for all $k \geq 0$. For the aircraft, the reference model must be chosen so that the closed-loop system will behave similarly to the unimpaired aircraft when the nominal control laws are used, and so that unreasonable performance requirements are not requested. With these two constraints in mind, we choose a second-order transfer function

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

where $\omega_n = \sqrt{200}$ and $\zeta = 0.85$ for the reference models for the roll rate and $H(s)/s$ for the reference model of the roll angle. An alternative choice for the reference model would be to use the actual nominal closed-loop system with a plant model since the objective of this control problem is to design an adaptive controller that will try to make a failed aircraft behave like the nominal unfailed aircraft.

Learning Mechanism Design Procedure

The learning mechanism consists of two parts: (1) a fuzzy inverse model, which performs the function of mapping the necessary changes in the process output error $y_{e\phi}(kT)$, $y_{ep}(kT)$, and $y_{e\dot{p}}(kT)$, to the relative changes in the process inputs $y_f(kT)$, so that the process outputs will match the reference model outputs, and (2) a knowledge-base modifier that updates the fuzzy controller's knowledge-base. As discussed earlier, from one perspective the fuzzy inverse model represents information that the control engineer has about what changes in the plant inputs are needed so that the plant outputs track the reference model outputs. From another point of view that we use here, the fuzzy inverse model can be considered as another fuzzy controller in the adaptation loop that is used to monitor the error

signals $y_{e\phi}(kT)$, $y_{ep}(kT)$, and $y_{e\dot{p}}(kT)$, and then choose the controller parameters in the main loop (i.e., the lower portion of Figure 6.12) in such a way that these errors go to zero. With this concept in mind, we introduce the following design procedure for the FMRLC, which we have found to be very useful for the fault-tolerant control application (it is based on the two procedures in Section 6.2.5):

1. Initialize the fuzzy controller by designing its rule-base to achieve the highest performance possible when the learning mechanism is disconnected. (If you wish to initialize the fuzzy controller rule-base so that all the output membership functions are located at zero (as in Section 6.2), then this design procedure should be applied iteratively where for each pass through the design procedure the trained fuzzy controller from steps 5–6 is used to initialize the fuzzy controller in step 1.)

2. Choose a reference model that represents the desired closed-loop system behavior (you must be careful to avoid requesting unreasonable performance).

3. Choose the rule-base for the fuzzy inverse model in a manner similar to how you would design a standard fuzzy controller (if there are many inputs to the fuzzy inverse model, then follow the approach taken in the application of this procedure below).

4. Find the range in which the i^{th} input to the fuzzy inverse model lies for a typical reference input and denote this by $[-\bar{R}_i, \bar{R}_i]$ ($i = 1, 2, \ldots, n$ where n denotes the number of inputs).

5. Construct the FMRLC with the domain interval of the output universe of discourse $[-\bar{R}_0, \bar{R}_0]$ to be $[0, 0]$, which is represented by the output gain m_0 of the fuzzy inverse model set at zero. Then, excite the system with a reference input that would be used in normal operation (such as a series of step changes, but note that simulations must be run long enough so that possible instabilities are detected). Then, increase the gain m_0 and observe the process response until the desired overall performance is achieved (i.e., the errors between the reference models and system outputs are minimum).

6. If there are difficulties in finding a value of m_0 that improves performance, then check the following three cases:

   (a) If there exist unacceptable oscillations in a given process output response about the reference model response, then choose the domain intervals of the input universes of discourse for the fuzzy inverse model to be $[-a_i\bar{R}_i, a_i\bar{R}_i]$ where a_i is a scaling factor that must be selected (typically, it lies in the range $0 < a_i \leq 10$), and go back to step 5. Note that the value of a_i should not be chosen too small, nor too large, such that the resulting domain interval $[-a_i\bar{R}_i, a_i\bar{R}_i]$ is out of the operating range of the system output; often you would choose to enlarge the input universes of discourse by decreasing a_i.

(b) If a process response is acceptable but there exist unacceptable oscillations in the command input to the plant, then adjust the rule-base of the fuzzy inverse model and go back to step 4.

(c) If the process output is unable to follow the reference model response, then choose a different reference model (typically at this point, you would want to choose a "slower" (i.e., less demanding) reference model), and go back to step 3.

It is important to note that for step 5, investigations have shown that the choice of m_0 significantly affects the learning capabilities and stability of the system. Generally, the size of m_0 is proportional to the learning rate, and with $m_0 = 0$ learning capabilities are turned off completely. Hence, for applications where a good initial guess for the controller is known and only minor variations occur in the plant, you may want to choose a relatively small value of m_0 to ensure stability yet allow for some learning capabilities. For other applications where significant variations in the plant are expected (e.g., failures), you may want to choose a larger value for m_0 so that the system can quickly learn to accommodate for the variation. In such a situation there is, however, a danger that the larger value of m_0 could lead to an instability. Hence, you generally want to pick m_0 large enough so that the system can quickly adapt to variations, yet small enough to ensure a stable operation. Moreover, we would like to emphasize that if a single step response is used as an evaluation during the tuning procedure, there exists the danger that the resulting system may not be stable for other inputs. Thus, a long enough reference input sequence must be used to show whether using a specific m_0 will result in a stable overall system. Next, we finish the design of the FMRLC by using the above design procedure to choose the learning mechanism.

In the F-16 aircraft application, step 1 of the design procedure was presented earlier where the fuzzy controller was picked so that it emulated the gain matrix of the nominal controller. After the equivalent fuzzy controller was constructed, the reference models were picked as described in step 2. Following step 3, the rule-base of the fuzzy inverse model is constructed. To ensure smooth performance at all times, we would like the fuzzy inverse model (viewed as a controller) to provide the capability to correct a big error quickly and adjust more slowly for minor errors; that is indicated in the input-output map for the fuzzy inverse model in Figure 6.13. To realize the map in Figure 6.13 we use (1) a similar rule-base initialization procedure to the one discussed in the fuzzy controller design, where we picked a set of uniformly spaced input membership functions for each of the three input universes of discourse, and (2) the centers of the output membership functions given by a nonlinear function of the input membership function centers.

According to step 4, the difference between the reference model responses and the system outputs are measured when $m_0 = 0$. Based on this information, the ranges of all the three inputs to the fuzzy inverse model $y_{e\phi}(kT)$, $y_{ep}(kT)$, and $y_{e\dot{p}}(kT)$ are found to be $[-4.4, 4.4]$, $[-8.4, 8.4]$, and $[-97.6, 97.6]$. For the first iteration, we will choose $a_i = 1$ (where $i = 1, 2, 3$).

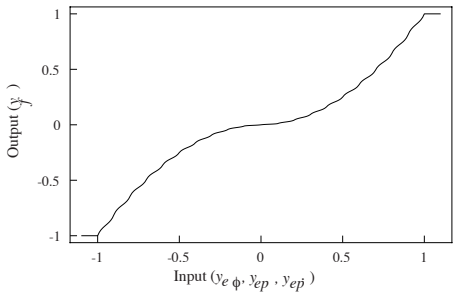


FIGURE 6.13 Input-output relationships for $y_{e\phi}$, y_{ep}, and $y_{ep\dot{}}$ to y_f maps (figure taken from [103], © IEEE).

In order to apply step 5, the loaded roll sequence is repeated several times. In this first iteration of the design procedure, the gain m_0 is found to be 0.02, which is a relatively small value that will not give significant learning capabilities. Therefore, we will proceed to step 6, and apply condition (a) where the scaling factors a_i $(i = 1, 2, 3)$ are selected to obtain a higher m_0. After a few iterations, the scaling factors are found to be $a_1 = 2.273$, $a_2 = 5.952$, and $a_3 = 2.049$ such that the domain intervals for the input universes of discourse for the fuzzy inverse model are $[-10, 10]$, $[-50, 50]$, and $[-200, -200]$, which correspond to $y_{e\phi}(kT)$, $y_{ep}(kT)$, and $y_{ep\dot{}}(kT)$. Then, m_0 is found to be 0.1, and the tuning procedure is completed.

Notice that the actual acceptable m_0, where the difference between the reference models and the system outputs is deemed small enough, is found to be in the range [0.05, 0.11] (i.e., a range of m_0 values worked equally well). Due to the fact that we would like the largest possible value of m_0 (i.e., higher learning capabilities) to adapt to failures in the aircraft, and we would like to ensure stability of the overall system, we picked the value of $m_0 = 0.1$. Moreover, we will not consider conditions (b) and (c) under step 6 because we assumed that the rule-base of the fuzzy inverse model represents good knowledge about how to minimize the errors between the reference model and the aircraft, and the reference models are indeed the design specifications for the aircraft that must be met in all cases.

Simulation Results

In this section, the F-16 aircraft with the FMRLC is simulated using the sampling time T of 0.02 seconds, and tested with an aileron failure at 1 second. We found that the FMRLC performed equally well when failures were induced at other times. The $t = 1$ sec failure time was chosen as it represents the first time maximum aileron deflection is achieved. Figure 6.14 compares the performance of the FMRLC to the nominal control laws for the case where there is no failure. All six plots show that the FMRLC performs as good as, if not better than, the nominal control laws. Notice that the FMRLC achieves its goal of following the reference models of the roll angle and the roll rate, except for slight steady-state errors (see the portions of the

response indicated by the arrows in Figure 6.14) where the responses of the FMRLC do not exactly match that of the nominal control laws. These errors are due to the fact that simple, second- or third-order, zero steady-state error reference models (roll rate/roll angle) are picked for the closed-loop multiple perturbation models of the aircraft. This discrepancy between the nominal controller and FMRLC responses is due to the difficulties you encounter in defining the reference models that can accurately emulate the nominal behavior of a given closed-loop system.

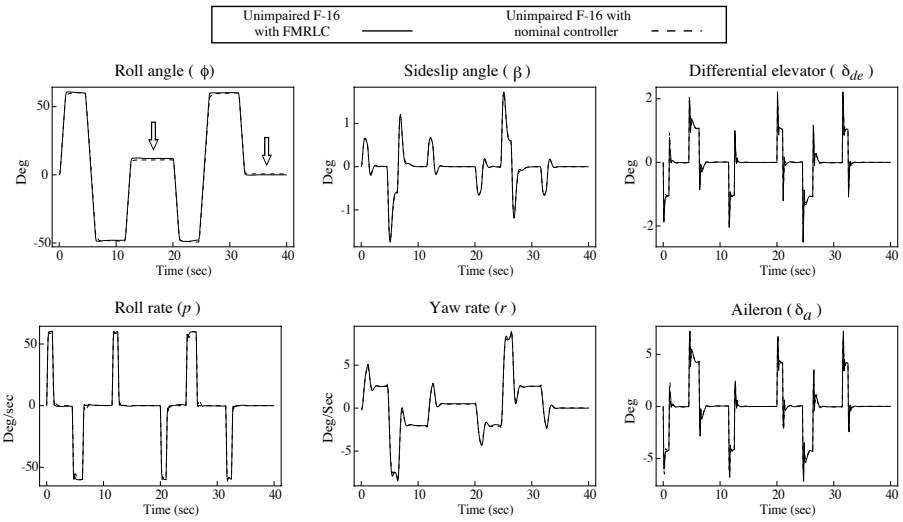


FIGURE 6.14 Unimpaired F-16 system outputs with FMRLC (figure taken from [103], © IEEE).

In case of failure, when the ailerons stick at 1 sec, the responses are shown in Figure 6.15. The FMRLC system responses are acceptable since all the responses eventually match that of the unimpaired aircraft. However, the performance in the first 9 seconds of the command sequence is obviously degraded as compared to the unimpaired responses (the portions of the roll angle and roll rate responses highlighted with arrows in Figure 6.15) but improves as time goes on. The performance degradation precipitates from the actuator failure. As shown in the actuator responses in Figure 6.15, the differential elevator (δ_{de}) swings between -1.30 and 10.00 with a bias of about 4.5 degrees for the impaired aircraft with FMRLC. The actuation of the differential elevator replaces the original function of the aileron with the bias so that the effect of the failure is canceled. We see that via reconfigurable control, the differential elevator can be made to have the same effectiveness as the ailerons as a roll effector for this particular failure. This application will be revisited in Chapter 7 when we study supervision of adaptive fuzzy controllers.

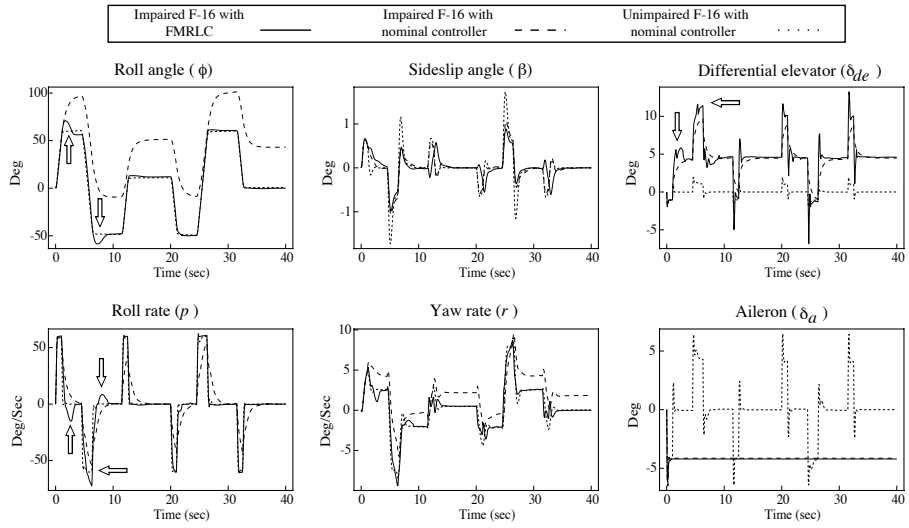


FIGURE 6.15 Impaired F-16 with FMRLC (aileron stuck at 1 sec) (figure taken from [103], © IEEE).

6.3.3 Vibration Damping for a Flexible Robot

For the two-link flexible robot considered here and in Chapter 3, our goal of achieving fast slews over the entire workspace with a minimum amount of endpoint vibration is complicated by two factors:

1. The manner in which varying the inertial configuration of the links has an effect on structural parameters (e.g., its effects on the modes of vibration).

2. Unknown payload variations (i.e., what the robot picks up), which significantly affect the plant dynamics.

Using several years of experience in developing conventional controllers for the robot mechanism, coupled with our intuitive understanding of the dynamics of the robot, in Chapter 3 we developed a fuzzy controller that achieves adequate performance for a variety of slews. However, even though we were able to tune the fuzzy controller to achieve such performance for varying configurations, its performance generally degrades when there is a payload variation at the endpoint.

While some would argue that the solution to such a performance degradation problem is to "load more expertise into the rule-base," there are several limitations to such a philosophy including the following:

1. The difficulties in developing (and characterizing in a rule-base) an accurate intuition about how to best compensate for the unpredictable and significant payload variations that can occur while the robot is in any position in its workspace.

2. The complexities of constructing a fuzzy controller that potentially has a large number of membership functions and rules.

Moreover, our experience has shown that it is possible to tune fuzzy controllers to perform very well if the payload is known. Hence, the problem does not result from a lack of basic expertise in the rule-base, but from the fact that there is no facility for automatically redesigning (i.e., retuning) the fuzzy controller so that it can appropriately react to unforeseen situations as they occur.

In this case study, we develop an FMRLC for automatically synthesizing and tuning a fuzzy controller for the flexible robot. We use the FMRLC structure shown in Figure 6.16, which tunes the coupled direct fuzzy controller from Chapter 3 and is simply a MIMO version of the one shown in Figure 6.3 on page 321. Next, we will describe each component of the FMRLC for the two-link flexible robot.

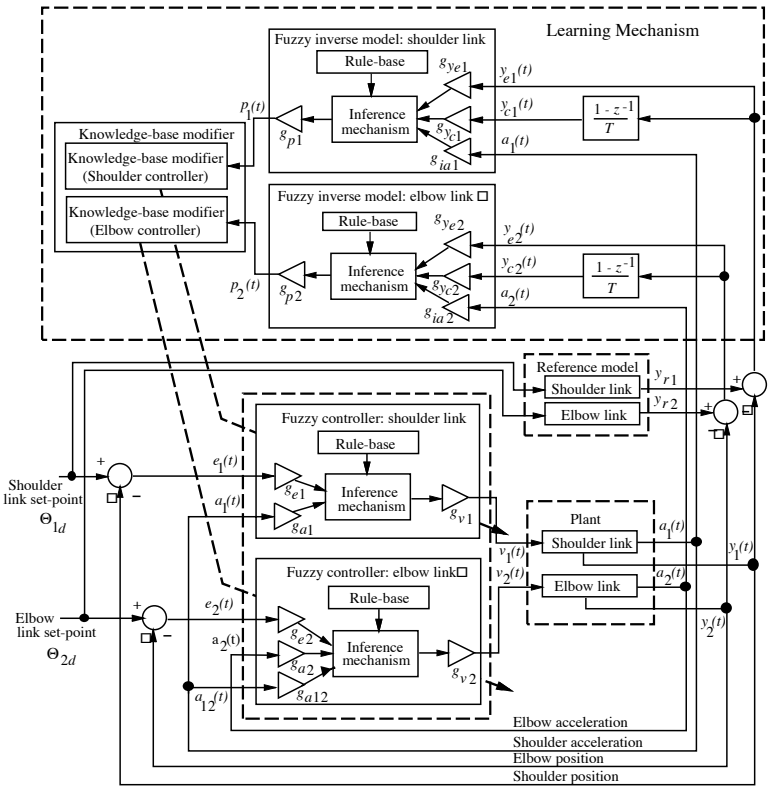


FIGURE 6.16 FMRLC for the flexible-link robot (figure taken from [144], © IEEE).

The Fuzzy Controller and Reference Model

We use the same basic structure for the fuzzy controller as was used in Chapter 3 with the same input fuzzy sets as shown in Figure 3.4 on page 131 and Figure 3.8 on page 136, but the difference here is that the output fuzzy sets for both controllers are all initially centered at zero, resulting in rule-bases filled with zeros (we tried to initialize the fuzzy controller with the one from Chapter 3 but it works best if it is initialized with zeros and constructs its own controller). This implies that the fuzzy controller by itself has little knowledge about how to control the plant. As the algorithm executes, the output membership functions are rearranged by the learning mechanism, filling up the rule-base. For instance, once a slew is commanded, the learning mechanism described below will move the centers of the output membership functions of the activated rules away from zero and begin to synthesize the fuzzy controller.

The universe of discourse for the position error input e_1 to the shoulder link controller was chosen to be $[-100, +100]$ degrees, and the universe of discourse for the endpoint acceleration a_1 is $[-10, +10]$ g. For the elbow link controller, the universe of discourse for the position error e_2 is $[-80, +80]$ degrees, and the universe of discourse for the acceleration input a_2 is $[-10, +10]$ g. The universe of discourse for the shoulder link acceleration input a_{12} to the elbow link controller is $[-8, +8]$ g. We choose the output universe of discourse for v_1 and v_2 by letting $g_{v1} = 0.125$ and $g_{v2} = 1.0$. We determined all these values from our experiences experimenting with the fuzzy controller in Chapter 3 and from our experiments with the FMRLC.

The desired performance is achieved if the learning mechanism forces

$$y_{e1}(kT) \approx 0, \quad y_{e2}(kT) \approx 0$$

for all $k \geq 0$. It is important to make a proper choice for a reference model so that the desired response does not dictate unreasonable performance requirements for the plant to be controlled. Through experimentation, we determined that

$$\frac{3}{s+3}$$

is a good choice for the reference models for both the shoulder and the elbow links.

The Fuzzy Inverse Models

There are several steps involved in specifying the fuzzy inverse models and these are outlined next.

Choice of Inputs: For our robot there are two fuzzy inverse models, each with three inputs $y_{ej}(t)$, $y_{cj}(t)$, and $a_j(t)$ ($j = 1$ corresponding to the shoulder link and $j = 2$ corresponding to the elbow link, as shown in Figure 6.16). Several issues dictated the choice of these inputs: (1) we found it easy to specify reference models for the shoulder and elbow link position trajectories (as we discussed above) and hence the position error signal is readily available; (2) we found via experimentation that the rates of change of position errors, $y_{cj}(t)$, $j = 1, 2$, and acceleration signals

$a_j(t)$, $j = 1, 2$, were very useful in deciding how to adjust the fuzzy controller; and (3) we sought to minimize the number of inputs to the fuzzy inverse models to ensure that we could implement the FMRLC with a short enough sampling interval (in our case, 15 milliseconds). The direct use of the acceleration signals $a_j(t)$, $j = 1, 2$, for the inverse models actually corresponds to choosing reference models for the acceleration signals that say "no matter what slew is commanded, the desired accelerations of the links should be zero" (why?). While it is clear that the links cannot move without accelerating, with this choice the FMRLC will attempt to accelerate the links as little as possible to achieve the command slews, thereby minimizing the amount of energy injected into the modes of vibration. Next, we discuss rule-base design for the fuzzy inverse models.

Choice of Rule-Base: For the rule-bases of the fuzzy inverse models, we use rules similar to those described in Tables 3.3–3.9 beginning on page 137, for both the shoulder and elbow links except that the cubical block of zeros is eliminated by making the pattern of consequents uniform. These rules have premises that quantify the position error, the rate of change of the position error, and the amount of acceleration in the link. The consequents of the rules represent the amount of change that should be made to the direct fuzzy controller by the knowledge-base modifier. For example, fuzzy inverse model rules capture knowledge such as (1) if the position error is large and the acceleration is moderate, but the link is moving in the correct direction to reduce this error, then a smaller change (or no change) is made to the direct fuzzy controller than if the link were moving to increase the position error; and (2) if the position error is small but there is a large change in position error and a large acceleration, then the fuzzy controller must be adjusted to avoid overshoot. Similar interpretations can be made for the remaining portions of the rule-bases used for both the shoulder and elbow link fuzzy inverse models.

Choice of Membership Functions: The membership functions for both the shoulder and elbow link fuzzy inverse models are similar to those used for the elbow link controller shown in Figure 3.8 on page 136 except that the membership functions on the output universe of discourse are uniformly distributed and there are different widths for the universes of discourse, as we explain next (these widths define the gains $g_{y_{ej}}$, $g_{y_{cj}}$, g_{aj}, and g_{pj} for $j = 1, 2$). We choose the universe of discourse for y_{ei} to be $[-80, +80]$ degrees for the shoulder link and $[-50, +50]$ for the elbow link. We have chosen a larger universe of discourse for the shoulder link inverse model than for the elbow link inverse model because we need to keep the change of speed of the shoulder link gradual so as not to induce oscillations in the elbow link (the elbow link is mounted on the shoulder link and is affected by the oscillations in the shoulder link). The universe of discourse for y_{c1} is chosen to be $[-400, +400]$ degrees/second for the shoulder link and $[-150, +150]$ degrees/second for y_{c2} of the elbow link. These universes of discourse were picked after experimental determination of the angular velocities of the links. The output universe of discourse for the fuzzy inverse model outputs (p_1 and p_2) is chosen to be relatively small to keep the size of the changes to the fuzzy controller small, which helps ensure smooth movement of the robot links. In particular, we choose the output universe of dis-

course to be $[-0.125, +0.125]$ for the shoulder link inverse model, and $[-0.05, +0.05]$ for the elbow link inverse model. Choosing the output universe of discourse for the inverse models to be $[-1, +1]$ causes the learning mechanism to continually make the changes in the rule-base of the controller so that the actual output is exactly equal to the reference model output, making the actual plant follow the reference model closely. This will cause significant amounts of speed variations in the motors as they try to track the reference models exactly, resulting in chattering along a reference model path. The choice of a smaller width for the universe of discourse keeps the actual output below the output of the reference model until it reaches the set-point. This increases the settling time slightly, but the response is much less oscillatory. This completes the definition of two fuzzy inverse models in Figure 6.16.

The Knowledge-Base Modifier

Given the information (from the inverse models) about the necessary changes in the input needed to make $y_{e1} \approx 0$ and $y_{e2} \approx 0$, the knowledge-base modifier changes the knowledge-base of the fuzzy controller so that the previously applied control action will be modified by the amount specified by the inverse model outputs $p_i, i = 1, 2$. To modify the knowledge-base, the knowledge-base modifier shifts the centers of the output membership functions (initialized at zero) of the rules that were "on" during the previous control action by the amount $p_1(t)$ for the shoulder controller and $p_2(t)$ for the elbow controller.

Note that to achieve good performance, we found via experimentation that certain enhancements to the FMRLC knowledge-base modification procedure were needed. In particular, based on the physics of the flexible robot, we know that if the errors e_1 and e_2 are near zero, the fuzzy controller *should* choose $v_1 = v_2 = 0.0$. Hence, using this knowledge about how to control the plant, we use the same FMRLC knowledge-base modification procedure as in Section 6.2.3 except that *we never modify the rules at the center of the rule-base* so the fuzzy controller will always output zero when there is zero error. Essentially, we make this adjustment to the knowledge-base modification procedure to overcome a high gain effect near zero that we observed in previous experiments.

Experimental Results

The total number of rules used by the FMRLC is 121 for the shoulder controller, plus 343 for the elbow controller, plus 343 for the shoulder fuzzy inverse model, plus 343 for the elbow fuzzy inverse model, for a total of 1150 rules. Even with this number of rules, we were able to keep the same sampling time of $T = 15$ milliseconds that was used for the direct fuzzy controller in Chapter 3.

Experimental results obtained from the use of the FMRLC are shown in Figure 6.17 for a slew of 90° for each link (see inset for inertial configuration). The rise-time for the response is about 1.0 sec and the settling time is approximately 1.8 sec. Comparing this response to the direct fuzzy control response (Figure 3.9 on page 140), we see an improvement in the endpoint oscillation and the settling time. Notice that the settling time for the robot is slightly larger than that of the reference

model (1.5 sec). This is because of the way the learning mechanism modified the rule-base of the controller to keep the response below that of the reference model. For counterrelative or small-angle slews, we get good results that are comparable to the direct fuzzy control case.

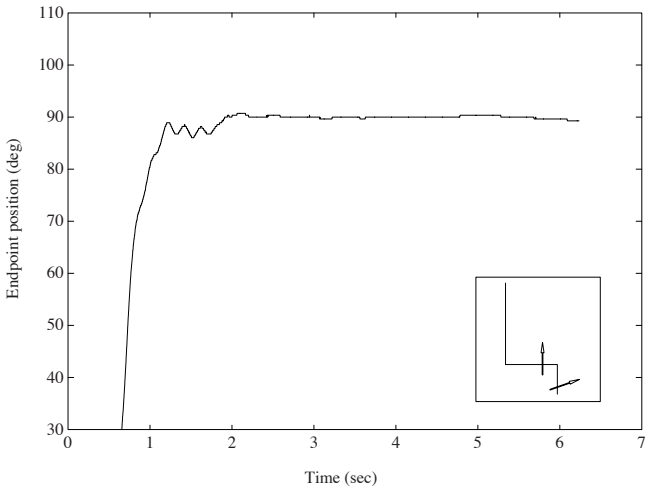


FIGURE 6.17 Endpoint position for FMRLC controller design (figure taken from [144], © IEEE).

Figure 6.18 shows the robot response for the loaded endpoint case. The elbow link endpoint is loaded with a 30-gram mass of aluminum and is commanded to slew 90° in each joint. The response with the payload here is superior to that of the direct fuzzy controller (see Figure 3.10 on page 141). To achieve the improved performance shown in Figure 6.18, the FMRLC exploits (1) the information that we have about how to control the flexible robot that is represented in the fuzzy inverse model and (2) data gathered during the slewing operation, as we discuss next. During the slew, the FMRLC observes how well the fuzzy controller is performing (using data from the reference model and robot) and seeks to adjust it so that the performance specified in the reference model is achieved and vibrations are reduced. For instance, in the initial part of the slew the position errors are large, the change in errors are zero, the accelerations are zero, and the fuzzy controller has all its consequent membership functions centered at zero. For this case, the fuzzy inverse model will indicate that the fuzzy controller should generate voltage inputs to the robot links that will get them moving in the right direction. As the position errors begin to change and the change in errors and accelerations vary from zero, the fuzzy inverse model will cause the knowledge-base modifier to fill in appropriate changes to the fuzzy controller consequent membership functions until the position trajectories match the ones specified by the reference models (notice that the fuzzy

inverse model was designed so that it will *continually* adjust the fuzzy controller until the reference model behavior is achieved). Near the end of the slew (i.e., when the links are near their commanded positions), the FMRLC is particularly good at vibration damping since in this case the plant behavior will repeatedly return the system to the portion of the fuzzy controller rule-base that was learned the last time a similar oscillation occurred (i.e., the learning capabilities of the FMRLC enable it to develop, remember, and reapply a learned response to plant behaviors).

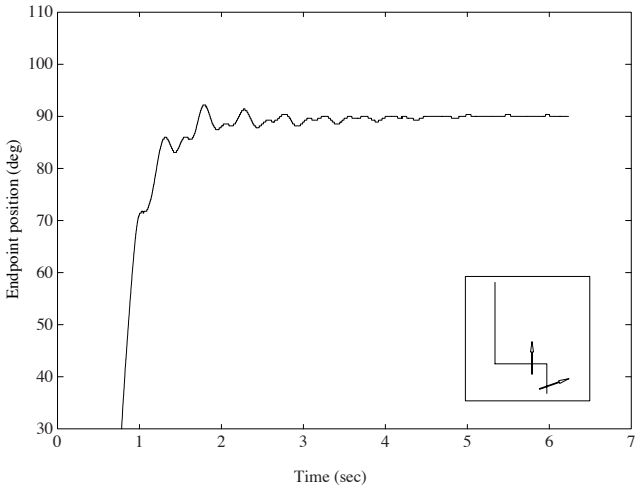


FIGURE 6.18 Endpoint position for loaded elbow link for FMRLC (figure taken from [144], © IEEE).

Different payloads change the modal frequencies in the link/payload combination (e.g., heavier loads tend to reduce the frequencies of the modes of oscillation) and the shapes of the error and acceleration signals $e_1(t)$, $e_2(t)$, and $a_1(t)$ (e.g., heavier loads tend to slow the plant responses). Hence, changing the payload simply results in the FMRLC developing, remembering, and applying different responses depending on the type of the payload variation that occurred. Essentially, the FMRLC uses data from the closed-loop system that is generated during on-line operation of the robot to specially tailor the manner in which it designs/tunes the fuzzy controller. This enables it to achieve better performance than the direct fuzzy controller described in Chapter 3.

Finally, we note that if a series of slews is made we do not use the fuzzy controller that is learned by the end of one slew to initialize the one that is tuned in the next slew. We found experimentally that it is a bit better to simply zero all the elements of the rule-base for each slew and have it re-learn the rule-base each time. The reason for this is that the fuzzy controller that is learned for a slew is in a sense optimized for the vibration damping near the end of the slew and not the

large angle movements necessary in the initial part of the slew (note that most of the time, the rule-base modifications are being made during the vibration damping phase). This shows that there is some room for improvement of this FMRLC where additional inputs to the fuzzy controller may allow it to learn and remember the appropriate controllers in different operating regions so they do not have to be re-learned.

6.4 Dynamically Focused Learning (DFL)

As we pointed out at the beginning of Section 6.2, a learning system possesses the capability to improve its performance over time by interacting with its environment. A learning control system is designed so that its learning controller has the ability to improve the performance of the closed-loop system by generating command inputs to the plant and utilizing feedback information from the plant. Learning controllers are often designed to mimic the manner in which a human in the control loop would learn how to control a system while it operates. Some characteristics of this human learning process may include the following:

1. A natural tendency for the human to *focus* her or his learning by paying particular attention to the current operating conditions of the system since these may be most relevant to determining how to enhance performance.

2. After the human has learned how to control the plant for some operating condition, if the operating conditions change, then the best way to control the system may have to be relearned.

3. A human with a significant amount of experience at controlling the system in one operating region should not forget this experience if the operating condition changes.

To mimic these types of human learning behavior, in this section we introduce three strategies that can be used to dynamically focus a learning controller onto the current operating region of the system. We show how the subsequent "dynamically focused learning" (DFL) can be used to enhance the performance of the FMRLC that was introduced and applied in the last two sections, and we also perform comparative analysis with a conventional adaptive control technique.

Ultimately, the same overall objectives exist as for the FMRLC. That is, we seek to provide a way to automatically synthesize or tune a direct fuzzy controller since it may be hard to do so manually or it may become "detuned" while in operation. With DFL, however, we will be tuning not only the centers of the output membership functions, but also the input membership functions of the rules. A magnetic ball suspension system is used throughout this section to perform the comparative analyses, and to illustrate the concept of dynamically focused fuzzy learning control.

6.4.1 Magnetic Ball Suspension System: Motivation for DFL

In this section we develop a conventional adaptive controller and an FMRLC for a magnetic ball suspension system and perform a comparative analysis to assess the advantages and disadvantages of each approach. At the end of this section, we highlight certain problems that can arise with the FMRLC and use these as motivation for the dynamically focused learning enhancement to the FMRLC.

Magnetic Ball Suspension System

The model of the magnetic ball suspension system shown in Figure 6.19 is given by [102]

$$\begin{aligned} M\frac{d^2y(t)}{dt^2} &= Mg - \frac{i^2(t)}{y(t)} \\ v(t) &= Ri(t) + L\frac{di(t)}{dt} \end{aligned} \tag{6.25}$$

where $y(t)$ is the ball position in meters, $M = 0.1$ kg is the ball mass, $g = 9.8$ m/s^2 is the gravitational acceleration, $R = 50\ \Omega$ is the winding resistance, $L = 0.5$ Henrys is the winding inductance, $v(t)$ is the input voltage, and $i(t)$ is the winding current. The position of the ball is detected by a position sensor (e.g., an infrared, microwave, or photoresistive sensor) and is assumed to be fully detectable over the entire range between the magnetic coil and the ground level. We assume that the ball will stay between the coil and the ground level (and simulate the system this way). In state-space form, Equation (6.25) becomes

$$\begin{aligned} \frac{dx_1(t)}{dt} &= x_2(t) \\ \frac{dx_2(t)}{dt} &= g - \frac{x_3^2(t)}{Mx_1(t)} \\ \frac{dx_3(t)}{dt} &= -\frac{R}{L}x_3(t) + \frac{1}{L}v(t) \end{aligned} \tag{6.26}$$

where $[x_1(t), x_2(t), x_3(t)]^\top = [y(t), \frac{dy(t)}{dt}, i(t)]^\top$. Notice that the nonlinearities are induced by the $x_3^2(t)$ and $\frac{1}{x_1(t)}$ terms in the $\frac{dx_2(t)}{dt}$ equation. By linearizing the plant model in Equation (6.26), assuming that the ball is initially located at $x_1(0) = y(0)$, we can find a linear system by calculating the Jacobian matrix at $y(0)$. The linear state-space form of the magnetic ball suspension system is given as

$$\begin{aligned} \frac{dx_1(t)}{dt} &= x_2(t) \\ \frac{dx_2(t)}{dt} &= \frac{g}{y(0)}x_1(t) - 2\sqrt{\frac{g}{My(0)}}x_3(t) \\ \frac{dx_3(t)}{dt} &= -\frac{R}{L}x_3(t) + \frac{1}{L}v(t) \end{aligned} \tag{6.27}$$

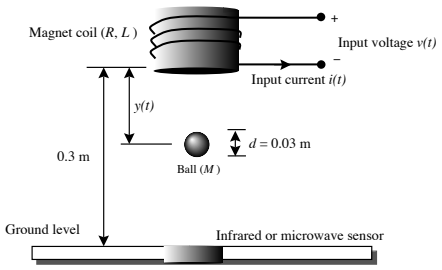


FIGURE 6.19 Magnetic ball suspension system (figure taken from [103], © IEEE).

Since the ball position $y(t)$ is the only physical output of the plant, by assuming that all initial conditions are zero for the linear perturbation model, we can rewrite the model as a transfer function

$$\frac{Y(s)}{V(s)} = \frac{-\frac{2}{L}\sqrt{\frac{g}{My(0)}}}{(s^2 - \frac{g}{y(0)})(s + \frac{R}{L})} \tag{6.28}$$

Note that there are three poles (two stable and one unstable) and no zeros in the transfer function in Equation (6.28). Two poles (one stable and one unstable) and the DC gain change based on the initial position of the ball so that the system dynamics will vary significantly depending on the location of the ball. From Figure 6.19, the total distance between the magnetic coil and the ground level is 0.3 m, and the diameter of the ball is 0.03 m. Thus, the total allowable travel is 0.27 m, and the initial position of the ball $y(0)$ can be anywhere between 0.015 m (touching the coil) and 0.285 m (touching the ground). For this range the numerator of the transfer function

$$-\frac{2}{L}\sqrt{\frac{g}{My(0)}}$$

varies from -323.3 (ball at 0.015 m) to -74.17 (ball at 0.285 m), while the two poles move from ± 25.56 to ± 5.864.

Clearly, then, the position of the ball will affect our ability to control it. If it is close to the coil it may be difficult to control since the unstable pole moves farther out into the right-half plane, while if it is near the ground level it is easier to control. The effect of the ball position on the plant dynamics can cause problems with the application of fixed linear controllers (e.g., ones designed with root locus or Bode techniques that assume the plant parameters are fixed). It is for this reason that we investigate the use of a conventional adaptive controller and the FMRLC for this control problem. We emphasize, however, that our primary concern is not with the determination of the best control approach for the magnetic ball suspension system; we simply use this system as an example to compare control approaches

and to illustrate the ideas in this section.

Conventional Adaptive Control

In this section a model reference adaptive controller (MRAC) is designed for the magnetic ball suspension system. The particular type of MRAC we use is described in [180] (on p. 125); it uses the so-called "indirect" approach to adaptive control where the updates to the controller are made by first identifying the plant parameters. To design the MRAC, a linear model is required. To make the linear model most representative of the range of dynamics of the nonlinear plant, we assume that the ball is initialized at the middle between the magnetic coil and the ground level where $y(0) = 0.15$ m to perform our linearization. In order to simplify the MRAC design, we will assume that the plant is second-order by neglecting the pole at -100 since its dynamics are much faster than the remaining roots in the plant. We found via simulation that the use of this second-order linear model has no significant effect on the low-frequency responses compared to the original third-order linear model. Hence, the transfer function of the system is rewritten as (note that the DC gain term k_p is changed accordingly)

$$\frac{Y(s)}{V(s)} = \frac{-1.022}{s^2 - 65.33} \tag{6.29}$$

A reference model is used to specify the desired closed-loop system behavior. Here, the reference model is chosen to be

$$\frac{-25}{s^2 + 10s + 25} \tag{6.30}$$

This choice reflects our desire to have the closed-loop response with minimal overshoot, zero steady-state error, and yet a stable, fast response to a reference input. Moreover, to ensure that the "matching equality" is achieved (i.e., that there will exist a set of controller parameters that can achieve the behavior specified in the reference model [180]) we choose the order of the reference model to be the same as that of the plant.

We use a "normalized gradient algorithm with projection" [180] to update the parameters of a controller. Since the plant is assumed to be second-order, based on the theory of persistency of excitation [180], the identifier parameters will converge to their true values if an input that is "sufficiently rich" of an order that is at least twice the order of the system is used. Therefore, an input composed of the sum of two sinusoids will be used to obtain richness of order four according to the theory. In order to pick the two sinusoids as the input, it would be beneficial to study the frequency response of the plant model. The way to pick the inputs is that the frequency selected should be able to excite most of the frequency range we are interested in. A Bode plot of the third-order linear system suggested that the cutoff frequency (3dB cutoff) of the plant is about 6 rad/sec. Hence, we picked two sinusoids (1 rad/sec and 10 rad/sec) to cover the most critical frequency range.

The amplitude of the input is chosen to force the system, as well as the reference model, to swing approximately between ±0.05 m around the initial ball position (i.e., at 0.15 m, where the total length of the system is 0.3 m); hence, we choose $r(t) = 0.05(\sin(1t) + \sin(10t))$. Note that this input will drive the system into different operating conditions where the plant behavior will change due to the nonlinearities.

Next, the adaptive controller will be simulated with two different plant models to demonstrate the closed-loop performance. The two plant models used are (1) the second-order linear model, and (2) the original nonlinear system (i.e, Equation (6.26)).

Second-Order Linear Plant: With an appropriate choice for the adaptation gains, we get the responses shown in Figure 6.20, where the identifier error $e_i = y_i - y$ (where y_i is the output of the identifier model) is approaching zero in 0.5 sec, while the plant output error $e_o = y_m - y$ is still slowly converging after 20 sec (i.e., swinging between ±0.005 m) but $y(t)$ is capable of matching the response specified by the reference model in about 15 sec. Notice that there is a fairly large transient period in the first 2 seconds when both the identifier and the controller parameters are varying widely. We see that the system response is approaching the one specified by the reference model, but the convergence rate is quite slow (even with relatively large adaptation gains). Note that the voltage input v in Figure 6.20 is of acceptable magnitude compared with the implementation in [15]; in fact, all control strategies studied in this case study produced acceptable voltage control inputs to the plant compared to [15].

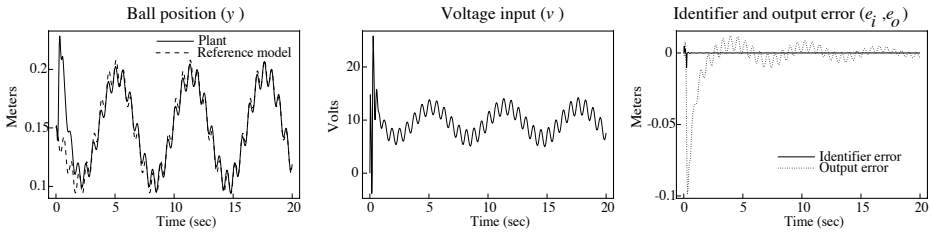


FIGURE 6.20 Responses using MRAC design (reduced order linear model, sinusoidal input sequence) (figure taken from [103], © IEEE).

Nonlinear Plant: In this section, the adaptive controller will be simulated with the nonlinear model of the ball suspension system with the same controller and initial conditions so that the ball starts at 0.15 m. Figure 6.21 shows the responses for the nonlinear model. It is observed that the ball first drops to the ground level since the adaptation mechanism is slow and it cannot keep up with the fast-moving system. After about 2.5 sec, the system starts to recover and tries to keep up with the plant. The identifier error e_i dies down after about 5 sec to where it swings between ±0.001 m; however, the plant output error swings between ±0.03 m and

appears to maintain at the same level (i.e., the plant output never perfectly matches that of the reference model). The plant output is not capable of matching the one specified by the reference model mainly because the indirect adaptive controller is not designed for the nonlinear model. We also observe that the ball position reacts better in the range where the ball is close to the ground level (0.3 m), whereas the response gets worse in the range where the ball is close to the magnetic coil (0 m) (i.e., the nonzero identifier error is found and the control input is more oscillatory in the instants when the ball position is closer to 0 m). This behavior is due to the nature of the nonlinear plant, where the system dynamics vary significantly with the ball position, and the adaptive mechanism is not fast enough to adapt the controller parameters with respect to the system dynamics.

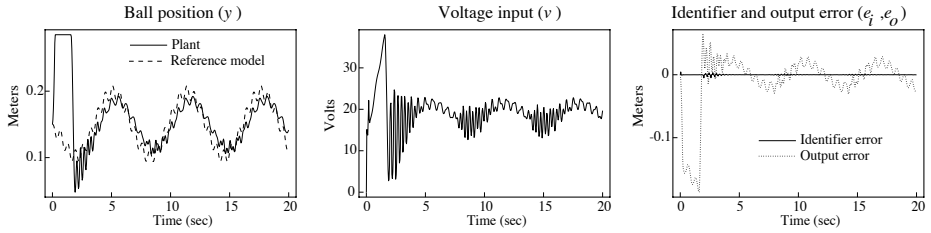


FIGURE 6.21 Responses for MRAC design (nonlinear model, sinusoidal input sequence) (figure taken from [103], © IEEE).

In order to keep the ball from falling to the ground level or lifting up to the coil, one approach is to apply the previously adapted controller parameters to initialize the adaptive controller. It is hoped that this initialization process would help the adaptation mechanism to keep up with the plant dynamics at the beginning of the simulation. As shown in Figure 6.22, when this approach is employed, the ball does not fall to the ground level (compared to Figure 6.21). Despite the fact that the system appears to be stable, the identifier error does not approach zero and swings between ± 0.001 m and the plant output error swings between ± 0.03 m (i.e., the closed-loop response of the plant is still not matching that of the reference model).

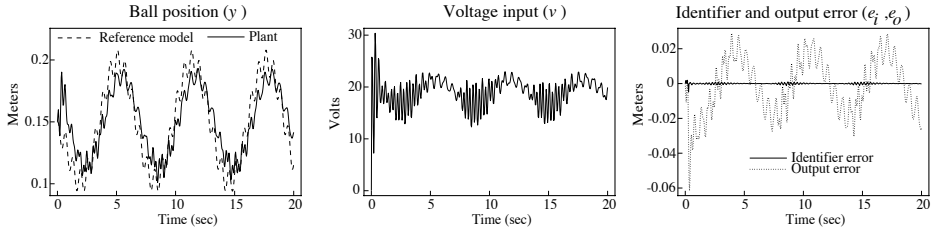


FIGURE 6.22 Responses for MRAC design after "training" (nonlinear model, sinusoidal input sequence) (figure taken from [103], © IEEE).

Unfortunately, if a step input sequence is used as the reference input to the nonlinear plant, as shown in Figure 6.23, the MRAC does a very poor job of following the reference model. However, the DFL strategies will significantly improve on this performance.

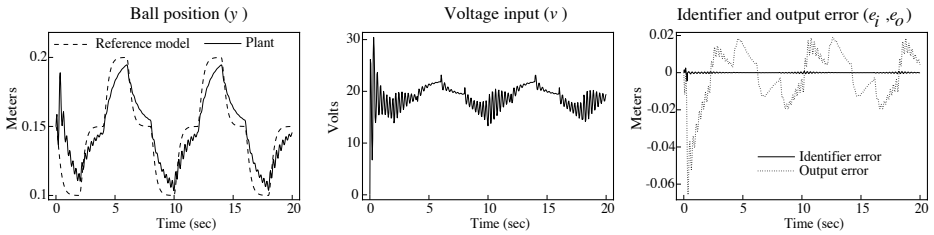


FIGURE 6.23 Responses for MRAC (nonlinear model, step input sequence) (figure taken from [103], © IEEE).

Fuzzy Model Reference Learning Control

In this section, the FMRLC will be designed for the magnetic ball suspension system. Note that the design of the FMRLC does not require the use of a linear plant model, and thus from now on we will always use the nonlinear model of the magnetic ball suspension system. The fuzzy controller uses the error signal $e(kT) = r(kT) - y(kT)$ and the change in error of the ball position $c(kT) = \frac{e(kT)-e(kT-T)}{T}$ to decide what voltage to apply so that $y(kT) \to r(kT)$ as $k \to \infty$. For our fuzzy controller design, the gains g_e, g_c, and g_v were employed to normalize the universe of discourse for the error $e(kT)$, change in error $c(kT)$, and controller output $v(kT)$, respectively. The gain g_e is chosen so that the range of values of $g_e e(kT)$ lies on $[-1, 1]$, and g_v is chosen by using the allowed range of inputs to the plant in a similar way. The gain g_c is determined by experimenting with various inputs to the system to determine the normal range of values that $c(kT)$ will take on; then g_c is chosen so that this range of values is scaled to $[-1, 1]$. According to this procedure, the universes of discourse of the inputs to the fuzzy controller $e(t)$ and $c(t)$ are chosen to be $[-0.275, 0.275]$ and $[-2.0, 2.0]$, respectively. This choice is made based on the distance between the coil and ground level of the magnetic ball suspension system and an estimate of the maximum attainable velocity of the ball that we obtain via simulations. Thus, the gains g_e and g_c are $\frac{1}{0.275}$ and $\frac{1}{2}$, respectively. The output gain g_v is then chosen to be 30, which is the maximum voltage we typically would like to apply to the plant.

We utilize one MISO fuzzy controller, which has a rule-base of If-Then control rules of the form

$$\textbf{If } \tilde{e} \text{ is } \tilde{E}^a \textbf{ and } \tilde{c} \text{ is } \tilde{C}^b \textbf{ Then } \tilde{v} \text{ is } \tilde{V}^{a,b}$$

where $\tilde{e}$ and $\tilde{c}$ denote the linguistic variables associated with controller inputs $e(kT)$ and $c(kT)$, respectively; $\tilde{v}$ denotes the linguistic variable associated with the controller output v; $\tilde{E}^a$ denotes the a^{th} linguistic value associated with $\tilde{e}$; $\tilde{C}^b$ denotes the b^{th} linguistic value associated with $\tilde{c}$; and $\tilde{V}^{a,b}$ denotes the consequent linguistic value associated with $\tilde{v}$. We use 11 fuzzy sets (triangular-shaped membership function with base widths of 0.4) on the normalized universes of discourse for $e(kT)$ and $c(kT)$, as shown in Figure 6.24(a).

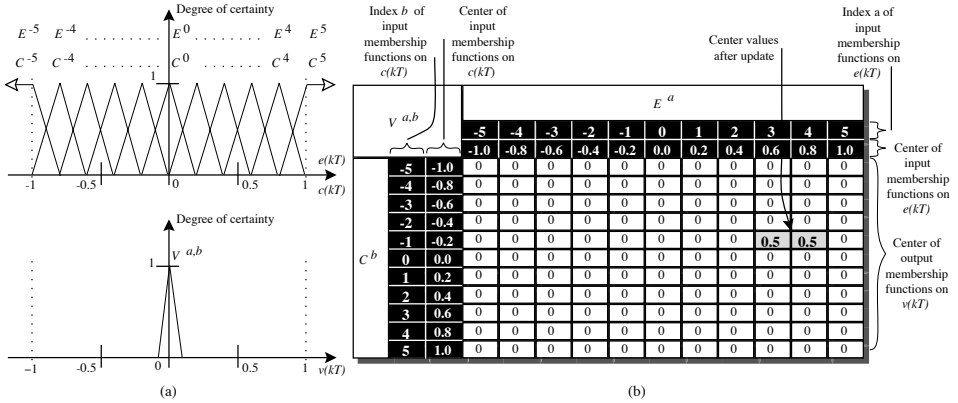


FIGURE 6.24 Input-output universes of discourse and rule-base for the fuzzy controller (figure taken from [103], © IEEE).

Assume that we use the same fuzzy sets on the $c(kT)$ normalized universes of discourse (i.e., $C^b = E^a$). As shown in Figure 6.24(a), we initialize the fuzzy controller knowledge-base with 121 rules (using all possible combinations of rules) where all the right-hand-side membership functions are triangular with base widths of 0.2 and centers at zero.

We use a discretized version of the same reference model as was used for the MRAC of the previous section for the conventional MRAC. The performance of the overall system is computed with respect to the reference model by generating error signals

$$y_e(kT) = y_m(kT) - y(kT)$$

and

$$y_c(kT) = \frac{y_e(kT) - y_e(kT - T)}{T}$$

The fuzzy inverse model is set up similar to the fuzzy controller with the same input membership functions as in Figure 6.24 for $y_e(kT)$ and $y_c(kT)$, but there are 21 triangular output membership functions that are uniformly spaced across a

$[-1, 1]$ effective universe of discourse. The rule-base is chosen so that it represents the knowledge of how to update the controller when the error, the change of error between the reference model, and the plant output are given. In particular, the centers of the membership functions for $Y_f^{a,b}$ (the inverse model output membership functions) are given by

$$-\frac{(a+b)}{10}$$

so that the rule-base has a similar form to the one in Table 6.1 on page 338 but has different off-diagonal terms. The gains of the fuzzy inverse model are then initially chosen to be $g_{y_e} = \frac{1}{0.275}$, $g_{y_c} = 0.5$, and $g_f = 30$. Note that all the gains are chosen based on the physical properties of the plant, so that $g_{y_e} = g_e$, $g_{y_c} = g_c$, and $g_f = g_v$ (more details on the rationale and justification for this choice for the gains is provided in Section 6.2).

According to the second design procedure in Section 6.2.3, a step input can be used to tune the gains g_c and g_{y_c} of the FMRLC. Here, we chose a step response sequence. Notice in the ball position plot in Figure 6.25 that the FMRLC design was quite successful in generating the control rules such that the ball position tracks the reference model almost perfectly. It is important to note that the FMRLC design here required no iteration on the design process. However, this is not necessarily true in general, and some tuning is most often needed for different applications.

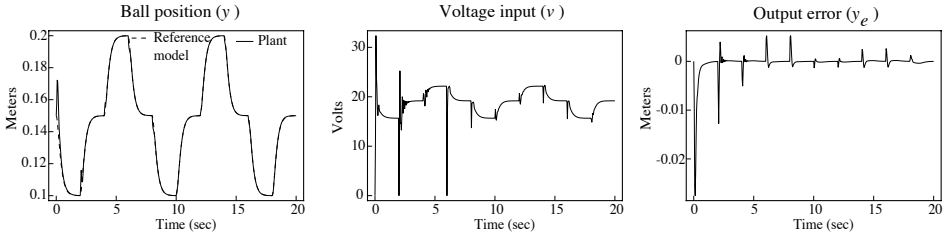


FIGURE 6.25 Responses for FMRLC (step input sequence) (figure taken from [103], © IEEE).

While the FMRLC seems quite successful, it is possible that there exists an input sequence that will cause the FMRLC to fail since stability of the FMRLC depends on the input (as it does for all nonlinear systems). For example, if the sinusoidal input sequence $r(t) = 0.05(\sin(1t)+\sin(10t))$ is used (as it was used in the adaptive controller design), the plant response is unstable, as shown in Figure 6.26, in the sense that the ball hits the coil and stays there. Notice that the ball hits the coil and even with a small (or zero) voltage is held there; this is a characteristic of the somewhat academic plant model, the saturations due to restricting the ball movements between the coil and ground level, and the way that the system is simulated. Although exhaustive tuning of the gains (except g_e, g_v, g_{y_e}, and g_f since

we artificially consider these to be set by the physical system) are performed to improve the FMRLC, Figure 6.26 indeed shows one of the best responses we can obtain.

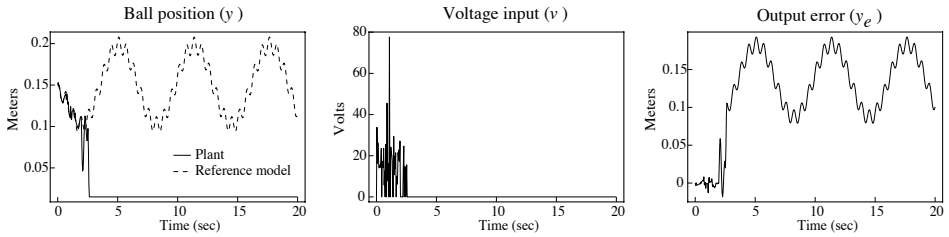


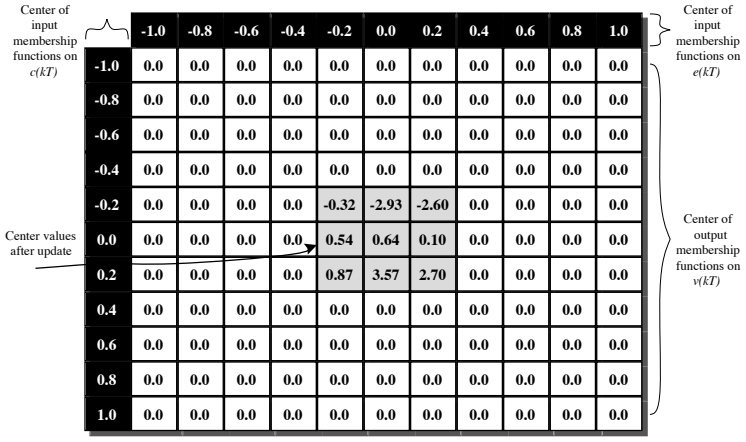
FIGURE 6.26 Responses for FMRLC (sinusoidal input sequence) (figure taken from [103], © IEEE).

Motivation for Dynamically Focused Learning

To gain better insight into why the FMRLC fails, in Figure 6.27 we show the learned rule-base of the fuzzy controller in the FMRLC (after the step input sequence in Figure 6.25 is applied to the system for 20 sec). This shows that the fuzzy controller actually utilized only 9 of the 121 possible rules. In fact, the 9 rules that are learned lie within the center section. With such a small number of rules, the learning mechanism of the FMRLC performed inadequately because the resulting control surface can capture only very approximate control actions. In other words, for more complicated control actions, such a rule-base may not be able to force the plant to follow the reference model closely.

To improve FMRLC performance, one possible solution is to redesign the controller so that the rule-base has enough membership functions at the center, where the most learning is needed. Yet, we will not consider this approach because the resulting controller will then be limited to a specific range of the inputs that happen to have been generated for the particular reference input sequence. Another possible solution is to increase the number of rules (by increasing the number of membership functions on each input universe of discourse) used by the fuzzy controller. Therefore, the total number of rules (for all combinations) is also increased, and we enhance the capability of the rule-base to memorize more distinct control actions (i.e., to achieve "fine control").

For instance, if we increase the number of membership functions on each input universe of discourse from 11 to, say 101 (but keeping all other parameters, such as the scaling gains, the same), the total number of rules will increase from 121 to 10,201—that is, there is a two order of magnitude increase in the number of rules (we chose this number of membership functions by trial and error and found that further increases in the number of membership functions had very little effect on performance), and we get the responses shown in Figure 6.28 for the FMRLC.



c(kT) \ e(kT)	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
-1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.2	0.0	0.0	0.0	0.0	-0.32	-2.93	-2.60	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.54	0.64	0.10	0.0	0.0	0.0	0.0
0.2	0.0	0.0	0.0	0.0	0.87	3.57	2.70	0.0	0.0	0.0	0.0
0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

FIGURE 6.27 Rule-base of the learned fuzzy controller (step input sequence) (figure taken from [103], © IEEE).

Clearly, as compared to Figure 6.26, we have drastically improved the performance of the FMRLC to the extent that it performs similarly to the MRAC for the nonlinear model (see Figure 6.22). Notice that in Figure 6.28 the output error swings between ± 0.027 m even after 15 sec of simulation, and the plant output is oscillatory. Longer simulations have shown that this FMRLC appears to be stable, but the plant cannot perfectly follow the response of the reference model.

Even though we were able to significantly improve performance, enlarging the rule-base has many disadvantages: (1) the number of rules increases exponentially for an increase in membership functions and inputs to the fuzzy controller, (2) the computational efficiency decreases as the number of rules increases, and (3) a rule-base with a large number of rules will require a long time period for the learning mechanism to fill in the correct control laws since smaller portions of the rule-base map in Figure 6.27 will be updated by the FMRLC for a higher-granularity rule-base (unless, of course, you raise the adaptation gain). Hence, the advantages of increasing the number of rules will soon be offset by practical implementation considerations and possible degradations in performance.

This motivates the need for special enhancements to the FMRLC so that we can (1) minimize the number of membership functions and therefore rules used, and (2) at the same time, maximize the granularity of the rule-base near the point where the system is operating (e.g., the center region of the rule-base map in Figure 6.27) so that very effective learning can take place.

FMRLC Learning Dynamics

Before introducing the DFL strategies that will try to more effectively use the rules, we clarify several issues in FMRLC learning dynamics including the following: (1) the effects of gains on linguistic values, and (2) characteristics of the rule-base such

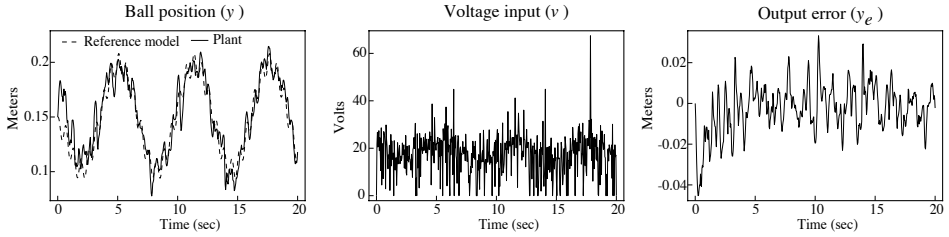


FIGURE 6.28 Responses for FMRLC (nonlinear model, sinusoidal input sequence) (figure taken from [103], © IEEE).

as granularity, coverage, and the control surface.

Effects on Linguistic Values: The fuzzy controller in the FMRLC used for the magnetic ball suspension system has 11 membership functions for each input ($e(kT)$ and $c(kT)$). There are a total of 121 rules, with all the output membership function centers initialized at zero. The universes of discourse for each process input are normalized to the interval $[-1, 1]$ by means of constant scaling factors. For our fuzzy controller design, the gains g_e, g_c, and g_v were employed to normalize the universe of discourse for the error $e(kT)$, change in error $c(kT)$, and controller output $v(kT)$, respectively. The gains g_e and g_c then act as the scaling factors of the physical range of the inputs. By changing these gains, the meanings of the premises of the linguistic rules will also be changed. An off-line tuning procedure for selecting these gains (such as the one described in Section 6.2) is essentially picking the appropriate meaning for each of the linguistic variables (recall our discussion in Chapter 2 on tuning scaling gains). For instance, one of the membership functions E^4 on $e(kT)$ is defined as "PositiveBig" (see Figure 6.24), and it covers the region $[0.6, 1.0]$ on $e(kT)$. With the gain $g_e = \frac{1}{0.275}$, the linguistic term "PositiveBig" quantifies the position errors in the interval $[0.165, 0.275]$. If the gain is increased to $g_e = \frac{1}{0.05}$ (i.e., reducing the domain interval of the universe of discourse from $[-0.275, 0.275]$ to $[-0.05, 0.05]$), then the linguistic term "PositiveBig" quantifies position errors in the interval $[0.03, 0.05]$. Note that the range covered by the linguistic term is reduced by increasing the scaling factor (decreasing the domain interval of the universe of discourse), and thus the true meanings of a membership function can be varied by the gains applied. The reader should keep this in mind when studying the DFL strategies in subsequent sections.

Rule-Base Coverage: As explained in Chapter 2, the fuzzy controller rule-base can be seen as a control surface. Then, a two-input single-output fuzzy controller can be viewed as a functional map that maps the inputs to the output of the fuzzy controller. Therefore, the FMRLC algorithm that constructs the fuzzy controller is essentially identifying this control surface for the specified reference model. With the "granularity" chosen by the number of membership functions and the gain, this control surface is normally most effective on the domain interval of the input universes of discourse (at the outer edges, the inputs and output of the fuzzy

controller saturate). For example, the gain $g_e = \frac{1}{0.275}$ is chosen to scale the input $e(kT)$ onto a normalized universe of discourse $[-1, 1]$. The domain interval of the input universe of discourse on $e(kT)$ is then bounded on $[-0.275, 0.275]$. Hence, a tuning procedure that changes the gains g_e and g_c is altering the "coverage" of the control surface. Note that for a rule-base with a fixed number of rules, when the domain interval of the input universes of discourse are large (i.e., small g_e and g_c), it represents a "coarse control" action; and when the input universes of discourse are small (i.e., large g_e and g_c), it represents a "fine control" action. Hence, we can vary the "granularity" of a control surface by varying the gains g_e and g_c.

Based on the above intuition about the gains and the resulting fuzzy controller, it is possible to develop different strategies to adjust the gains g_e and g_c so that a smaller rule-base can be used on the input range needed the most. This is done by adjusting the meaning of the linguistic values based on the most recent input signals to the fuzzy controller so that the control surface is properly *focused* on the region that describes the system activity. In the next section, we will give details on three techniques that we will be able to scale (i.e., "auto-tune"), to move (i.e., "auto-attentive"), and to move and remember (i.e., "auto-attentive with memory") the rule-base to achieve dynamically focused learning for FMRLC.

For comparison purposes, all the fuzzy controllers in the following sections have 121 rules, where each of the input universes of discourse have 11 uniformly spaced membership functions (the same ones that were used in Figure 6.24). The initial gains g_e and g_c are chosen to be $\frac{1}{0.05}$ and $\frac{1}{0.5}$, respectively (this choice will make the initial rule-base of the fuzzy controller much smaller than the center learned region in Figure 6.27), in order to ensure various DFL approaches for FMRLC are activated so that we can study their behavior. It is interesting to note that with this choice of gains, the FMRLC (without dynamically focused learning) will produce the unstable responses shown in Figure 6.29 (see the discussion on Figure 6.26 where similar behavior is observed). In the following sections we will introduce techniques that will focus the rule-base so that such poor behavior (i.e., where the ball is lifted to hit the coil) will be avoided.

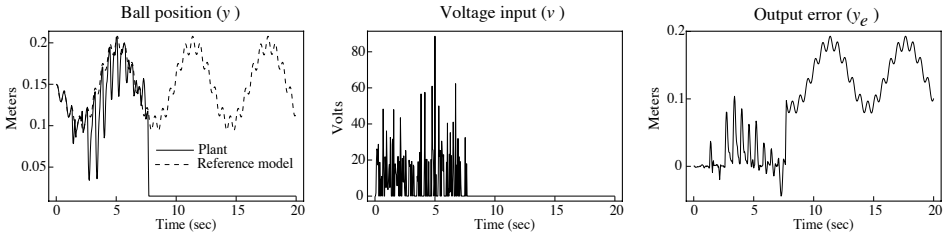


FIGURE 6.29 Responses for FMRLC with reduced rule-base and no DFL (sinusoidal input sequence) (figure taken from [103], © IEEE).

6.4.2 Auto-Tuning Mechanism

In the standard FMRLC design for the magnetic ball suspension system, the input sequence does not excite the whole range of the designated input universes of discourse (see Figure 6.27). Instead, the rule-base learned for the input sequence only covered the center part of the rule-base. Hence, to achieve an adequate number of rules to enhance the granularity of the rule-base near the center, it would be necessary to design the rule-base so that it is located at exactly where most of the rules are needed. However, we would like to ensure that we can adapt the fuzzy rule-base should a different input sequence drive the operation of the system out of this center region.

Auto-Tuning

Based on our experience in tuning the FMRLC, it is often observed that the gains g_e and g_c are chosen as bounds on the inputs to the controller so that the rule-base represents the active region of the control actions (e.g., see the cargo ship FMRLC design example in Section 6.3.1 on page 333). We base our on-line auto-tuning strategy for the input scaling gains on this idea. Let the maximum of each fuzzy controller input over a time interval (window) of the last T_A seconds be denoted by $\max_{T_A}\{e(kT)\}$ and $\max_{T_A}\{c(kT)\}$. Then this maximum value is defined as the gain of each input $e(kT)$ and $c(kT)$ so that

$$g_e = \frac{1}{\max_{T_A}\{e(kT)\}}$$

and

$$g_c = \frac{1}{\max_{T_A}\{c(kT)\}}$$

For the magnetic ball suspension system, after some experimentation, we chose $T_A = 0.1$ sec (it was found via simulations that any $T_A \in [0.05, 0.3]$ sec can be used equally effectively). Longer time windows tend to slow down the auto-tuning action; while a shorter window often speeds up the auto-tuning, but the resulting control is more oscillatory. Once the gains are changed, it is expected that the learning mechanism of the FMRLC will adjust the rules accordingly when they are reactivated, because the scaling will alter all the rules in the rule-base.

Note that the learning process now involves two individual, distinct components: (1) the FMRLC learning mechanism that fills in the appropriate consequents for the rules, and (2) the auto-tuning mechanism (i.e., an adaptation mechanism) that scales the gains that actually redefine the premise membership functions. Normally, we make the learning mechanism operate "at a higher rate" than the auto-tuning mechanism for the premise membership functions in order to try to assure stability. If the auto-tuning mechanism is designed to be "faster" than the FMRLC learning mechanism, the learning mechanism will not be able to keep up with the changes made by the auto-tuning mechanism so it will never be able to learn the

rule-base correctly. The different rates in learning and adaptation can be achieved by adjusting the sampling period T of the FMRLC and the window length T_A of the auto-tuning mechanism.

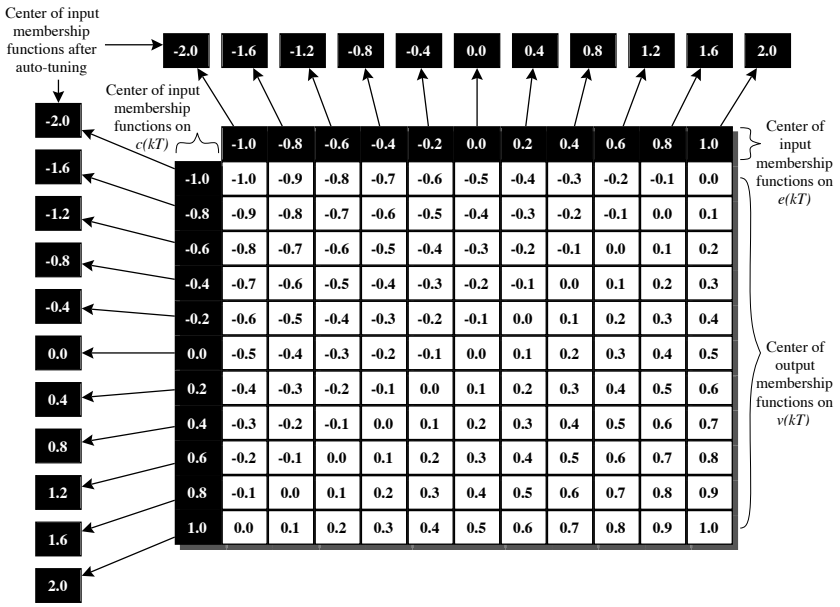


FIGURE 6.30 Dynamics of auto-tuning for FMRLC (figure taken from [103], © IEEE).

Figure 6.30 illustrates how the gain scaling implemented by auto-tuning affects the input membership functions. Note that the center of the output membership functions defined on the $v(kT)$ universe of discourse in Figure 6.30 are filled with a "standard" set of rules such that they represent a typical choice (for illustration purposes) from a control engineer's experience for the fuzzy controller. For example, at the beginning, the centers of each of the input membership functions are shown in the rule-base in Figure 6.30. In the next time instant, if the values $\max_{T_A}\{e(kT)\}$ and $\max_{T_A}\{c(kT)\}$ are halved, the gains

$$g_e = \frac{1}{\max_{T_A}\{e(kT)\}}$$

and

$$g_c = \frac{1}{\max_{T_A}\{c(kT)\}}$$

are now doubled. Then, the overall effect is that each of the membership functions

on the input universes of discourse is given a new linguistic meaning, and the domain of the control surface is expanded as shown by the centers of each input membership function after the auto-tuning action (see Figure 6.30).

Notice that we will require a maximum gain value; otherwise, each input universe of discourse for the fuzzy system may be reduced to zero (where the gains g_e and g_c go to infinity) so that controller stability is not maintained. For the magnetic ball suspension system, the maximum gain is chosen to be the same as the initial value (i.e., $g_e = \frac{1}{0.05}$ and $g_c = \frac{1}{0.5}$). Other gains g_v, g_{y_e}, g_{y_c} and g_f (the gain on the output of the model) are the same as those used in the standard FMRLC.

Auto-Tuning Results

For FMRLC with auto-tuning, Figure 6.31 shows that the ball position can follow the sinusoidal input sequence very closely, although perfect tracking of the reference response is not achieved. However, this result is better than the case where conventional adaptive control is used (see Figure 6.22), and definitely better than the standard FMRLC design (see Figure 6.26). Notice that the results shown in Figure 6.31 are similar to those shown in Figure 6.28, where 10,201 rules are used; however, the auto-tuning approach used only 121 rules. There are extra computations needed to implement the auto-tuning strategy to take the maximum over a time interval in computing the gains. Figure 6.32 shows excellent responses for the same auto-tuned FMRLC with the step input sequence where the ball position follows the reference model without noticeable difference (compare to Figures 6.23 and 6.25 for the MRAC and FMRLC, respectively).

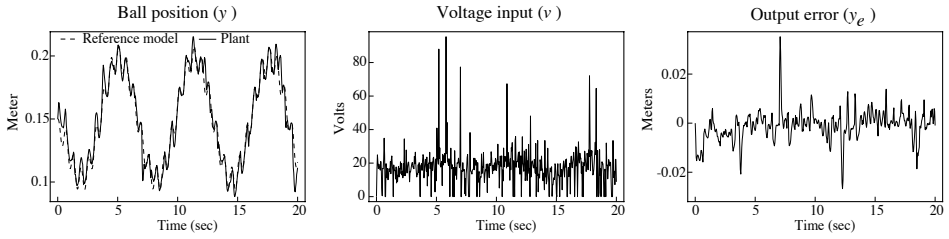


FIGURE 6.31 Responses for FMRLC with auto-tuning (sinusoidal input sequence) (figure taken from [103], © IEEE).

6.4.3 Auto-Attentive Mechanism

One of the disadvantages of auto-tuning the FMRLC is that all the rules in the rule-base are changed by the scaling of the gains, which may cause distortions in the rule-base and require the learning mechanism to relearn the appropriate control laws. Hence, instead of scaling, in this section we will consider moving the entire rule-base with respect to a fixed coordinate system so that the fuzzy controller can automatically "pay attention" to the current inputs.

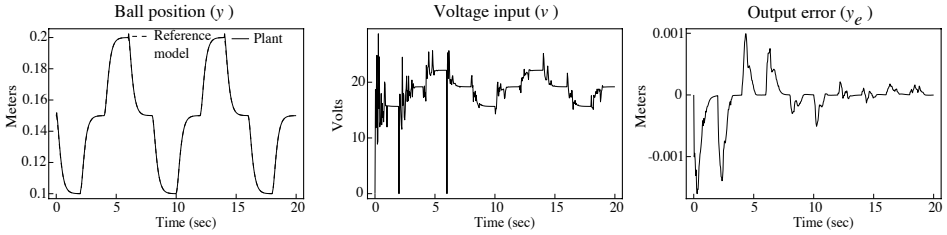


FIGURE 6.32 Responses for FMRLC with auto-tuning (step input sequence) (figure taken from [103], © IEEE).

Auto-Attentive Approach

To explain the auto-attentive mechanism, it is convenient to define some new terms that are depicted in Figure 6.33. First of all, the rule-base of the fuzzy controller is considered to be a single cell called the "auto-attentive active region," and it represents a fixed-size rule-base that is chosen by the initial scaling gains (i.e., g_e and g_c must be selected a priori). The outermost lightly shaded region of the rule-base is defined as the "attention boundary." The four lightly shaded rules (note that there are at most four rules "on" at one time due to our choice for membership functions shown in Figure 6.24) in the lower right portion of the rule-base are referred as the FMRLC "active learning region"; this is where the rules are updated by the learning mechanism of the FMRLC. Finally, the white arrow in Figure 6.33 indicates the direction of movement of the active learning region.

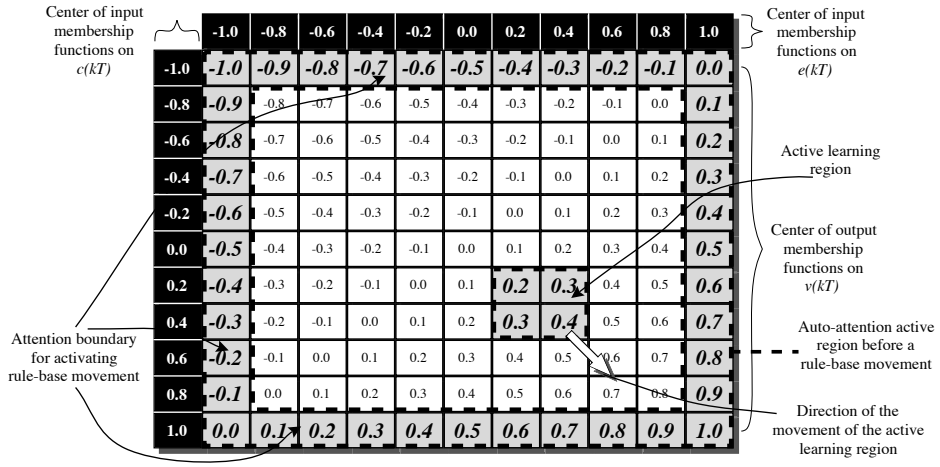


FIGURE 6.33 Auto-attentive mechanism for FMRLC (before shifting) (figure taken from [103], © IEEE).

For the auto-attentive mechanism, if the FMRLC active learning region moves

into the attention boundary, a "rule-base shift" is activated. For example, if the active learning region hits the lower-right attention boundary, as shown in Figure 6.34, the result is that the rule-base will be shifted down one unit and to the right one unit (i.e., the width of a membership function). We chose the convention that shifting the rule-base to the right and downward both correspond to *positive* offsets and shifting the rule-base to the left and upward both correspond to *negative* offsets. This choice is made to be compatible with the convention used in the input universes of discourse in the rule-base (as shown in Figures 6.33 and 6.34). Hence, the shift in the rule-base is represented by the "offset" of the rule-base from its initial position, which is $(E_{offset}, C_{offset}) = (1, 1)$ as shown in Figure 6.34 for this example. With the offset values, the shift of the rule-base is obtained simply by adding the offset values to each of the premise membership functions. After the rule-base is shifted, the active attention region is moved to the region in the large dashed box shown in Figure 6.34. In the new unexplored region (i.e., the darkly shaded row and column), the consequents of the rules will be filled with zeros since this represents that there is no knowledge of how to control in the new region. Another approach would be to extrapolate the values from the adjacent cells since this may provide a more accurate guess at the shape of the controller surface.

Conceptually, the rule-base is moving and following the FMRLC active learning region. We emphasize, however, that if the active learning region never hits the attention boundary, there will never be a rule-base shift and the controller will behave exactly the same as the standard FMRLC. Overall, we see that the auto-attentive mechanism seeks to keep the controller rule-base focused on the region where the FMRLC is learning how to control the system (one could think of this as we did with the auto-tuning mechanism as adapting the meaning of the linguistic values). If the rule-base shifts frequently, the system will "forget" how to control in the regions where it used to be, yet learn how to control in the new regions where adaptation is needed most. Note that we can consider the width of the attention boundary to be a design parameter, but we found that it is best to set the attention boundary as shown in Figure 6.33 since this choice minimizes oscillations and unnecessary shifting of the rule-base for this example.

Similar to the auto-tuning DFL strategy, there are two distinct processes here: (1) the FMRLC learning mechanism that fills in appropriate consequents for the rules, and (2) the auto-attentive mechanism (another adaptation mechanism) that moves the entire rule-base. Moreover, we think of the FMRLC learning mechanism as running at a higher rate compared to the auto-attentive mechanism (in order to try to assure stability), since we only allow a shift of the entire rule-base by a single unit in any direction in any time instant. The rate of adaptation can be controlled by using a different attention boundary to activate the rule-base movement. For example, if the attention boundary shown in Figure 6.33 is in the inner part of the rule-base (say, the second outermost region of the rule-base instead of the outermost region), then the rule-base will be shifted more often and thus increase the adaptation rate of the auto-attentive mechanism.

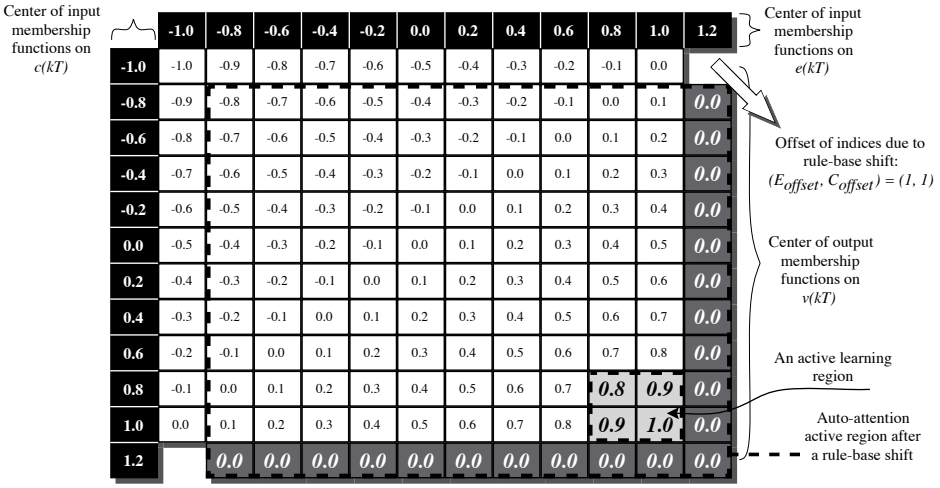


FIGURE 6.34 Auto-attentive mechanism for FMRLC (after shifting) (figure taken from [103], © IEEE).

Auto-Attentive Mechanism Results

For the magnetic ball suspension system, the input universes of discourse are chosen as $[-0.05, 0.05]$ and $[-0.5, 0.5]$ (i.e., the gains g_e and g_c are $\frac{1}{0.05}$ and $\frac{1}{0.5}$, respectively), while all the other gains are the same as the ones used in the standard FMRLC design in Section 6.4.1. Figure 6.35 illustrates the performance of the FMRLC with the auto-attentive mechanism. We see that the ball position can follow the input sequence very closely, although perfect tracking of the reference model cannot be achieved (with maximum output error y_e within ± 0.0078 m), but this result is better than the case with the conventional adaptive controller (see Figure 6.22), the standard FMRLC with 10201 rules (see Figure 6.28) and the auto-tuning FMRLC (see Figure 6.31); and definitely better than the case with the unstable standard FMRLC (see Figure 6.26, where the ball is lifted to the coil).

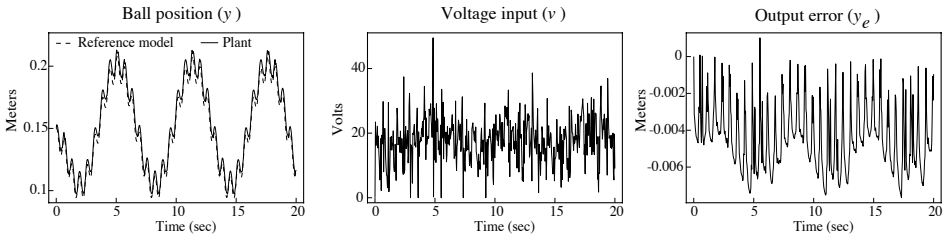


FIGURE 6.35 Responses for FMRLC with auto-attentive mechanism (sinusoidal input sequence) (figure taken from [103], © IEEE).

To gain insight into the dynamics of the auto-attentive mechanism, Figures 6.36(a) and (b) show the E_{offset} and C_{offset} values throughout the simulation, and Figure 6.36(c) depicts the first five movements of the rule-base. The double arrows in Figure 6.36(c) denote the movement of the rule-base from the initial position (shown as an empty box) to an outer region (shown as a shaded box), while the number next to the shaded box is the rule-base at the next time instant where the rule-base moved (the shades also change to deeper gray as time progresses). Hence, the rule-base is actually oscillating about to the (E_{offset}, C_{offset}) origin as time progresses, and it also moves around the initial position in a counterclockwise circular motion (this motion is induced by the sinusoids that the rule-base is trying to track). Note that we have done simulation tests for different sizes of the active attention region for improving the responses from the auto-attentive FMRLC. However, we found that smaller active attention regions result in excessive motion for the rule-base, while larger auto-attention active regions will have the same low rule-base "granularity" problem as the standard FMRLC. Figure 6.37 shows excellent responses for the same auto-attentive FMRLC design with a step input sequence, which is basically the same as in the case of standard FMRLC (see Figure 6.25).

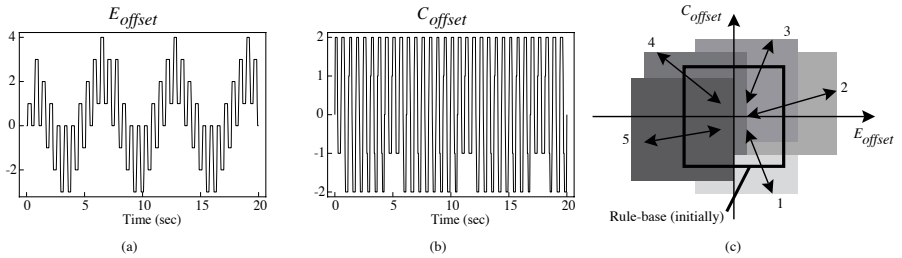


FIGURE 6.36 Movement of the rule-base for the auto-attentive mechanism (sinusoidal input sequence) (figure taken from [103], © IEEE).

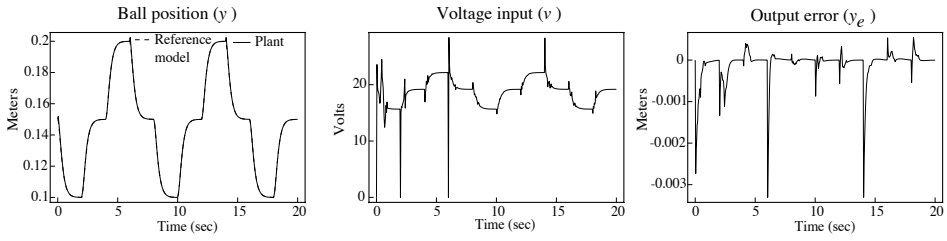


FIGURE 6.37 Responses for FMRLC with auto-attentive mechanism (step input sequence) (figure taken from [103], © IEEE).

6.4.4 Auto-Attentive Mechanism with Memory

Note that in the auto-attentive DFL strategy, every shift of the rule-base will create a new unexplored region (i.e., the darkly shaded row and column in Figure 6.34). This region will be filled with zeros since this represents that we have no knowledge of how to control when we move into a new operating condition. Having to learn the new regions from scratch after every movement of the rule-base can cause degradations in the performance of the auto-attentive FMRLC since it will require the learning mechanism to fill in the unknown rules (i.e., additional time for learning will be needed). For example, if an auto-attentive FMRLC has been operating for a long time on an input sequence, then at some time instant a disturbance affected the controller inputs and forced the rule-base to leave its current position, some of the rules are lost and replaced by new rules that will accommodate the disturbance. When the temporary disturbance is stopped and the rule-base returns to its initial position again, its previous experience is lost and it has to "relearn" everything about how to control in a region where it actually has gained a significant amount of experience.

Auto-Attentive Approach with Memory

There are two main components to add to the auto-attentive mechanism to obtain the auto-attentive mechanism with memory. These are the fuzzy experience model and its update mechanism.

Fuzzy Experience Model: To better reflect the "experience" that a controller gathers, we will introduce a third fuzzy system, which we call the "fuzzy experience model" for the FMRLC, as the memory to record an abstraction of the control laws that are in the region previously reached through the auto-attentive mechanism. The rule-base of this fuzzy experience model (i.e., the "experience rule-base") is used to represent the "global knowledge" of the fuzzy controller. In this case, no matter how far off the auto-attentive mechanism has offset the rule-base, there is a rough knowledge of how to control in any region the controller has visited before. In other words, this fuzzy controller not only possesses learning capabilities from the learning mechanism and adaptation abilities from the auto-attentive algorithm; it also maintains a representation of the "experience" it has gathered on how to control in an additional fuzzy system (an added level of memory and hence learning—with some imagination you can envision how to add successive nested learning/auto-attentive mechanisms and memory models for the FMRLC).

As shown in Figure 6.38, the fuzzy experience model has two inputs $e_{center}(kT)$ and $c_{center}(kT)$, which represent the center of the auto-attentive active region that is defined on $e(kT)$ and $c(kT)$. For our example, these inputs have five symmetric, uniformly spaced membership functions, and there are a total of 25 rules (i.e., 25 output membership functions that are initialized at zero). The universes of discourse for each of these inputs are normalized to the interval $[-1, 1]$ by means of constant

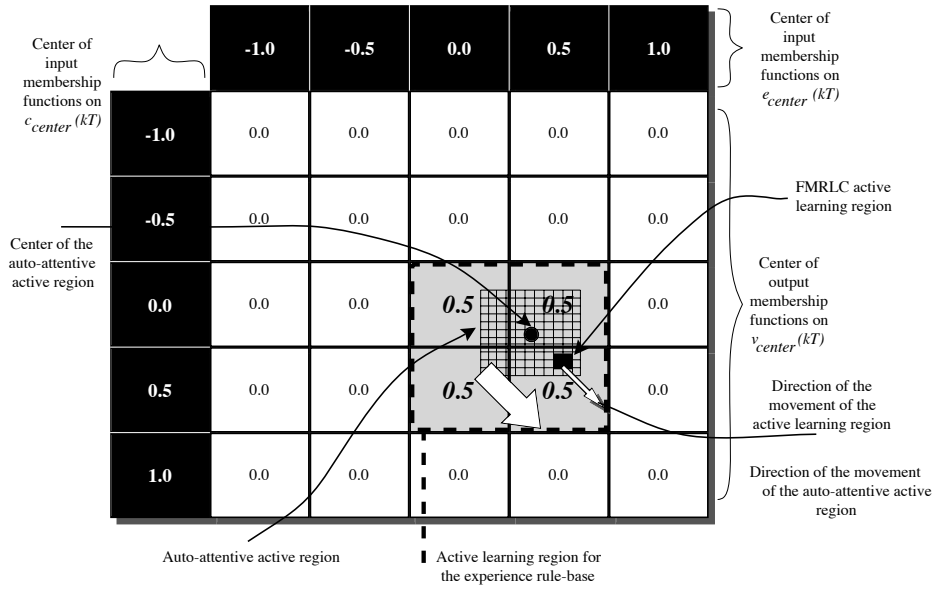


FIGURE 6.38 The fuzzy experience model for the auto-attentive mechanism with memory for FMRLC (figure taken from [103], © IEEE).

scaling factors. To represent the global knowledge, the gains

$$g_{e_{center}} = \frac{1}{0.275}$$

and

$$g_{c_{center}} = \frac{1}{2.0}$$

were employed to normalize the universe of discourse for the error $e_{center}(kT)$ and change in error $c_{center}(kT)$. The same gains used in the standard FMRLC design are employed here since these are assumed to represent the complete universes of discourse (determined by the physical limits) for the magnetic ball suspension system. The output universe of discourse is selected to be $[-1, 1]$ with gain $g_{v_{center}} = 1$, which preserves the original information from the fuzzy experience model without scaling.

Learning Mechanism for Fuzzy Experience Model: The learning mechanism for this fuzzy experience model is similar to the learning mechanism for the FMRLC except that the fuzzy inverse model is not needed for the fuzzy experience model. The two inputs $e_{center}(kT)$ and $c_{center}(kT)$ are used to calculate the "experience" value $v_{center}(kT)$ for the current auto-attentive active region, and the "activation level" of all the rules, while only the rules with activation levels larger

than zero will be updated (i.e., the same as the method used in the FMRLC learning mechanism in Section 6.2.3). Each time after the fuzzy controller rule-base (i.e., the auto-attentive active region) is updated, the numerical average value of the auto-attentive rule-base consequent centers, denoted by $v_{center(avg)}(kT)$, will be taken for the corresponding fuzzy experience model. Hence, the change of the consequent fuzzy sets of the experience rule-base that have premises with nonzero activation levels, can be computed as

$$v_{center(chg)} = v_{center(avg)}(kT) - v_{center}(kT)$$

and $v_{center(chg)}$ is used to update the fuzzy experience model exactly the same way as the fuzzy controller is updated in Section 6.2.3.

For example, the shaded area in Figure 6.38 (the active learning region for the experience rule-base) is activated by the inputs $e_{center}(kT)$ and $c_{center}(kT)$ (i.e., these are the rules that have nonzero activation level). First, assume that the centers of all membership functions on $v_{center}(kT)$ are zero at the beginning, and thus the output of the fuzzy experience model $v_{center}(kT)$ is zero. Then, assume we found $v_{center(avg)}(kT) = 0.5$ to be the average value of the control surface (i.e., average value of the centers of the output membership functions) for the auto-attentive active region; hence, the update of the fuzzy experience model $v_{center(chg)} = 0.5$ can be found. Hence, the consequent membership functions of the fuzzy experience model will be shifted to 0.5, as shown in the shaded region of Figure 6.38.

It is obvious that there are numerous other methods to obtain an abstract representation of the rule-base in the auto-attentive active region besides using the average. In fact, more complicated methods, such as using least squares to find a linear surface that best fits the control surface, could be used. However, we have found that such a method significantly increases the computational complexity without major performance improvements (at least, for the magnetic ball suspension system example). Our approach here uses a simple method to represent experience and hence provides a rough estimate of the unknown control laws.

Using Information from the Fuzzy Experience Model: As the auto-attentive active region moves, the shaded region at the boundary of the auto-attentive active region in Figure 6.34 can be filled in using the information recorded in the fuzzy experience model, instead of filling with zeros in the consequent of the rules. To do this we will need to perform a type of interpolation that pulls information off the experience model and puts it in the shaded region on the boundary. The interpolation is achieved by finding the consequent fuzzy sets for the unexplored region (see the shaded region in Figure 6.39) given the centers of each of the premise fuzzy sets. The enlarged active learning region for the experience rule-base shown in Figure 6.39 illustrates that there are 21 unexplored rules in the auto-attentive active region that needed to be estimated. We could simply compute the output of the experience model for each of the 21 cells in the shaded region and put these values in the shaded region. However, these computations are expensive for obtaining the guesses for the unexplored region, and thus we choose to compute only the consequent fuzzy sets for the center of the column (i.e., with input at

$(e_{center(column)}(kT), c_{center}(kT))$ as shown in Figure 6.39) and the row (i.e., with input at $(e_{center}(kT), c_{center(row)}(kT))$ as shown in Figure 6.39) of the unexplored region, and then fill the entire column or row with their center values.

Note that the auto-attentive mechanism that uses the fuzzy experience model for memory essentially performs a multidimensional interpolation, where a coarse rule-base is used to store the general shape of the global control surface and this information is used to fill in guesses for the auto-attentive active region as it shifts into regions that it has visited before. There are many other ways to store information in the experience rule-base and load information from it. For instance, we could use some of the alternatives to knowledge-base modification, or we could simply use the center value of the auto-attentive active region rule-base in place of the average. Sometimes you may know how to specify the experience rule-base a priori. Then you could omit the updates to the experience model and simply pull information off it as the auto-attentive active region moves.

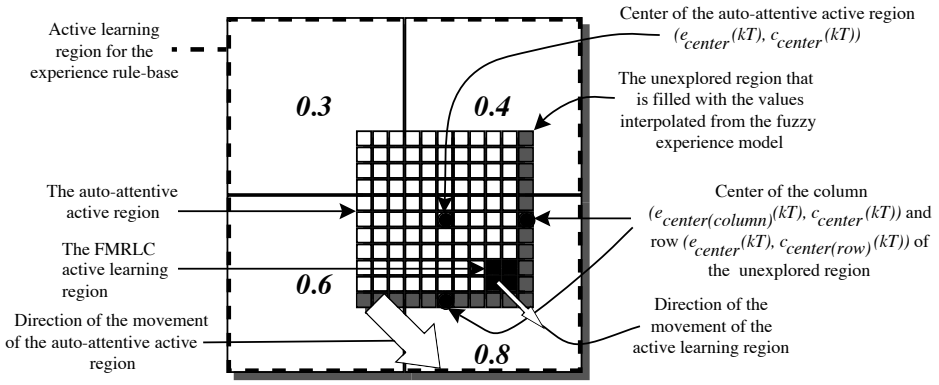


FIGURE 6.39 The enlargement of the active learning region for the experience rule-base (figure taken from [103], © IEEE).

Auto-Attentive Mechanism with Memory Results

As shown in Figure 6.40, when the auto-attentive mechanism with memory is used, the ball position can follow the input sequence almost perfectly with maximum output error y_e within ± 0.0022 m (i.e., about 3.5 times smaller than for the auto-attentive FMRLC without memory shown in Figure 6.35). Figure 6.41 shows the results for the same technique when we use a step input sequence. Notice that in terms of output error, these are the best results that we obtained (compared to the results from MRAC and the two other dynamic focusing techniques).

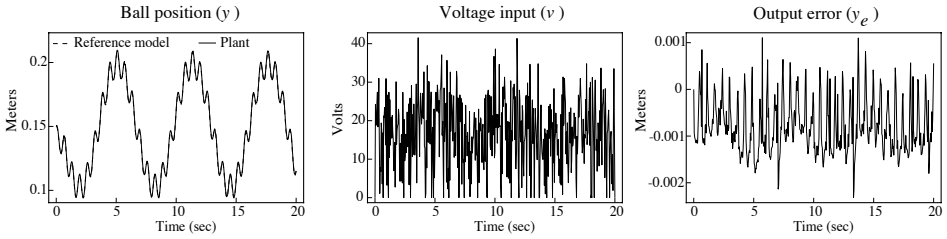


FIGURE 6.40 Responses for FMRLC with auto-attentive mechanism with memory (sinusoidal input sequence) (figure taken from [103], © IEEE).

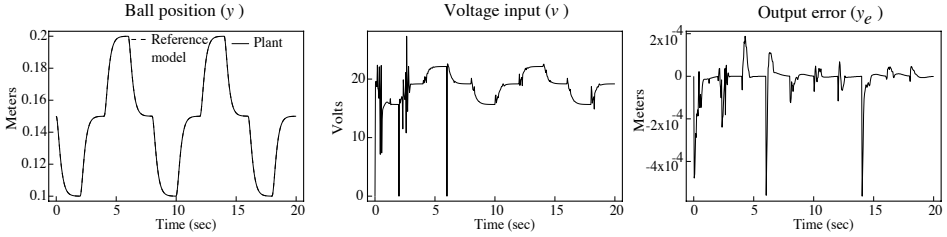


FIGURE 6.41 Responses for FMRLC with auto-attentive mechanism with memory (step input sequence) (figure taken from [103], © IEEE).

6.5 DFL: Design and Implementation Case Studies

In this section we provide two case studies in the design and implementation of DFL strategies. In the first, we will simply use the auto-tuning strategy on a direct fuzzy controller (not an FMRLC) for the rotational inverted pendulum case study of Chapter 3. Next, we will use similar auto-tuning strategy for the machine scheduling problem in Chapter 3. We refer the reader to the appropriate sections in Chapter 3 for background on the problem formulations and studies in direct fuzzy control for these problems.

6.5.1 Rotational Inverted Pendulum

While the direct fuzzy controller synthesized in Chapter 3 using the LQR gains performed adequately for the nominal case, its performance degraded significantly when a bottle of sloshing liquid was added to the endpoint. It is the objective of this section to use the auto-tuning method to try to achieve good performance for both the nominal and perturbed conditions without any manual tuning in between the two experiments. We will not auto-tune the FMRLC as we did in the last section. Here, we simply tune a direct fuzzy controller. This helps show the versatility of the dynamically focused learning concepts.

Auto-Tuning Strategy

The auto-tuning method for a direct fuzzy controller essentially expands on the idea of increasing the "resolution" of the fuzzy controller by dynamically increasing or decreasing the density of the input membership functions. For the rotational inverted pendulum, if we increase the number of membership functions on each input to 25, improved performance (and smoother control action) can be obtained. To increase the resolution of the direct fuzzy controller with a limited number of membership functions (as before, we will impose a limit of seven), we propose to use auto-tuning to dynamically focus the membership functions to regions where they are most useful.

Ideally, the auto-tuning algorithm should not alter the nominal control algorithm near the center; we therefore do not adjust each input gain independently. We can, however, tune the most significant input gain, and then adjust the rest of the gains based on this gain as we did in Chapter 3. For the inverted pendulum system, the most significant controller input is the position error of the pendulum, $e_3 = \theta_1$. The input-output gains are updated every n_s samples in the following manner:

1. Find the maximum e_3 over the most recent n_s samples and denote it by e_3^{max}.
2. Set the input gain $g_3 = \frac{1}{|e_3^{max}|}$.
3. Recalculate the remaining gains using the technique discussed in Chapter 3 so as to preserve the nominal control action near the center.

We note that the larger n_s is, the slower the updating rate is, and that too fast an updating rate may cause instability.

Auto-Tuning Results

Simulation tests with $n_s = 50$ and $g_3 = 2$ reveal that when the fuzzy controller is activated after swing-up, the input gains gradually increase while the output gain decreases, as the pendulum moves closer to its inverted position. As a result, the input and output universes of discourses contract, and the resolution of the fuzzy system increases. In practice, it is important to constrain the maximum value for g_3 (for our system, to a value of 10) because disturbances and inaccuracies in measurements could have adverse effects. As g_3 reaches its maximum value, the control action near $\theta_1 = 0$ is smoother than that of direct fuzzy control with 25 membership functions, and very good balancing performance is achieved.

When turning to actual implementation on the laboratory apparatus described in Chapter 3, some adjustments were done in order to optimize the performance of the auto-tuning controller. As with the direct fuzzy controller, the value of g_3 was adjusted upward, and the tuning (window) length was increased to $n_s = 75$ samples. In the first experiment we applied the scheme to the nominal system. In this case, the auto-tuning mechanism improved the response of the direct fuzzy controller (see Figure 3.15 on page 151) by varying the controller resolution on-line. That

is, as the resolution of the fuzzy controller increased over time, the high-frequency effects diminished.

However, the key issue with the adaptive (auto-tuning) mechanism is whether it can adapt its controller parameters as the process dynamics change. Once again we investigate the performance when the "sloshing liquid" dynamics (and additional weight) are added to the endpoint of the pendulum. As expected from simulation exercises, the auto-tuning mechanism effectively suppressed the disturbances caused by the sloshing liquid, as clearly shown in Figure 6.42. Overall, we see that the auto-tuning method provides a very effective controller for this application.

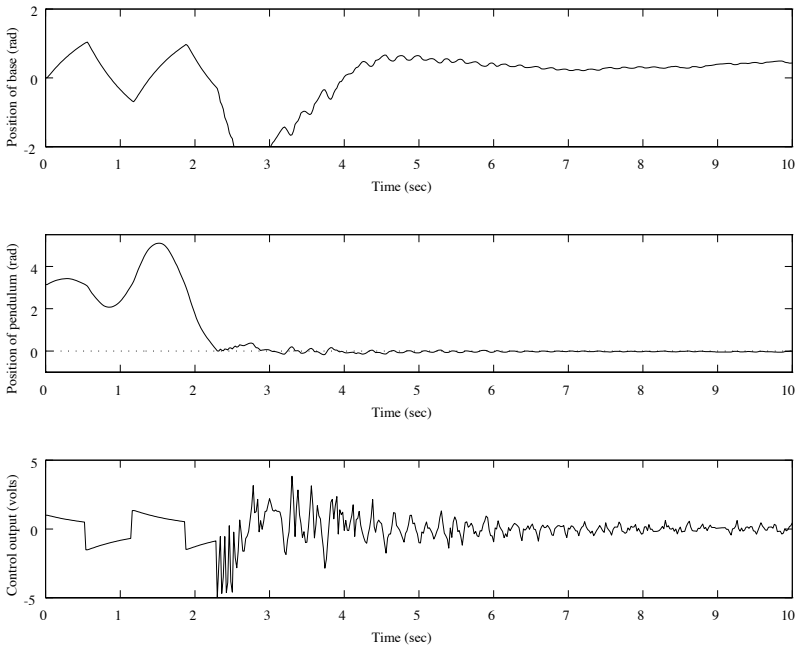


FIGURE 6.42 Experimental results: Auto-tuned fuzzy control on the pendulum with sloshing liquid at its endpoint (figure taken from [244], © IEEE).

6.5.2 Adaptive Machine Scheduling

In this section we develop an adaptive fuzzy scheduler (AFS) for scheduling the single machine by augmenting the fuzzy scheduler in Chapter 3 with an auto-tuning mechanism. In the AFS there is an adaptation mechanism that can automatically synthesize a fuzzy scheduler, independent of the machine parameters. Moreover, if there are machine parameter changes during operation that still satisfy the necessary conditions for stability (see Chapter 3), the AFS will tune the parameters of the

fuzzy scheduler so that high-performance operation is maintained. The universally stabilizing supervisory mechanism (USSM) that is described in Chapter 3 governs the AFS to ensure that it is stable. Therefore, the complete scheduler consists of three layers: The bottom layer is simply the fuzzy scheduler itself, the middle layer is the adaptation mechanism to be introduced here, and the top layer is the USSM that supervises the lower two layers to ensure stable operation.

If the parameters of the machine change, the USSM may not guarantee stability anymore since it assumes that the machine parameters stay constant. The parameter γ of the USSM is dependent on the parameters of the machine, whereas the parameters z_i are not. If the parameter γ is chosen large enough, the USSM may still provide stability over a large class of machine parameters. However, since the USSM assumes constant machine parameters, stability is not guaranteed when the machine parameters change even if γ is large enough for the new machine parameters. It is for this reason that we split the adaptation problem into controller synthesis (i.e., determining the positioning of a fixed number of fuzzy sets by automatically picking M_p) and controller tuning (i.e., tuning the positioning of the fuzzy sets by changing M_p to react to machine parameter changes). In synthesis we are guaranteed stability, while in tuning we have no proof that the policy is stable.

Automatic Scheduler Synthesis

In this section, we introduce the AFS that has an adaptive mechanism that observes x_p, $p \in \{1, 2, 3, \ldots, P\}$ and automatically tunes the values of M_p (see Chapter 3). This adaptation mechanism, shown in Figure 6.43, adjusts the parameters M_p of the fuzzy scheduler by using a moving window. The size of the window is not fixed but is equal to the length of time for a fixed number of production runs. In this section we will use a window size of 10 production runs, while in the next section we will use a larger window size. Throughout this window the buffer levels x_p are recorded. The window slides forward at the end of each production run, and the values of M_p are updated to the maximum values of x_p over the last window frame. As M_p is updated, the fuzzy sets on the universe of discourse for x_p are shifted so that they remain symmetric and uniformly distributed. The fuzzification and defuzzification strategies, the output fuzzy sets, and the rule-base remain constant so that the adaptation mechanism adjusts only the input membership functions to improve machine performance. Basically, the AFS tunes the M_p values in search of a lower η. It does this by automatically adjusting the premise membership functions of the rules of the fuzzy scheduler so that they appropriately fit the machine.

Next, we show how the automatic tuning method can be used to synthesize the fuzzy scheduler for the single machine. In particular, we will show how without any knowledge of the machine parameters, our adaptation mechanism can synthesize a fuzzy scheduler that can perform as well as the CPK policy (see Chapter 3). We shall first consider the same machines used in Chapter 3. For each of the following machines, the number of fuzzy sets is set to 5. The parameter M_p is initially set to 1. The adaptation mechanism will adjust M_p at the end of each production run.

When simulating the AFS for machine 1 (see Chapter 3 for its description),

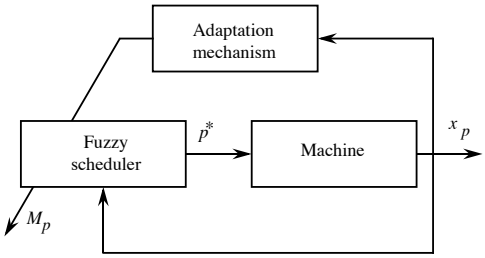


FIGURE 6.43 Adaptive fuzzy scheduler (AFS) for a single machine.

by the end of 15 production runs $M_1 = 30.993$, $M_2 = 34.605$, and $M_3 = 12.4519$ (and for later production runs, the M_p values stay near these values). After 10,000 production runs, we find that $\eta = 1.0263$, which is the same as that produced by CPK after 10,000 production runs. But the AFS automatically constructed its scheduler without knowledge of the machine parameters. The CPK policy uses machine parameters to help specify the policy and is therefore tailored specifically to the machine.

When simulating the AFS for machine 2 (see Chapter 3 for its description), we find that the values of M_p converge slowly compared to the previous machine (it took until about 700 production runs to get convergence). After 10,000 production runs, $\eta = 1.0993$ which is worse than the $\eta = 1.0017$ produced by CPK after 10,000 production runs. However, when M_p is initially 10,000 instead of 1 and the adaptation mechanism updates the M_p every other 10 production runs, η from fuzzy scheduler improves to 1.0018. This highlights an inherent problem with the adaptation mechanism: the window size and M_p update strategy must be chosen in an ad hoc manner with no guarantees on performance levels.

Automatic Scheduler Tuning

In this section we investigate whether the AFS and CPK can adjust themselves to disturbances or failures that may occur during the operation of a machine. The disturbance or failure may be in the form of changes in arrival rates, processing times, or setup times. In order to observe how the fuzzy scheduler and CPK adjust to machine parameter changes, first we use the same machine parameters and part-types, and switch the part-types to arrive at different buffers. Following this, we will investigate the tuning capabilities of the adaptation mechanism by examining the effects of changing the machine load. In the simulations, the machine parameters stay constant for the first 10,000 production runs, then the machine parameters are changed and remain constant at different values for the next 10,000 production runs. When the parameters are changed, the parameters M_p of the fuzzy scheduler are continued from the last production run. For the last 10,000 production runs, the CPK schedules based on the former machine parameters, while the AFS adjusts itself to improve performance.

1. Switching buffers:

   (a) Case 1:
   Old machine: $d_1 = 7$, $d_2 = 9$, $d_3 = 3$, $\tau_1 = 1/100$, $\tau_2 = 1/51$, $\tau_3 = 1/27$.
   New machine: $d_2 = 7$, $d_3 = 9$, $d_1 = 3$, $\tau_2 = 1/100$, $\tau_3 = 1/51$, $\tau_1 = 1/27$.
   The AFS maintains the same η at 1.026, whereas η of CPK degrades from 1.027 to 1.237.

   (b) Case 2:
   Old Machine: $d_1 = 18$, $d_2 = 3$, $d_3 = 1$, $\tau_1 = 1/35$, $\tau_2 = 1/7$, $\tau_3 = 1/20$.
   New Machine: $d_2 = 18$, $d_3 = 3$, $d_1 = 1$, $\tau_1 = 1/35$, $\tau_2 = 1/7$, $\tau_3 = 1/20$.
   The value of η of the AFS improves from 1.0993 to 1.0018, whereas η of CPK degrades from 1.0017 to 1.1965.

   The AFS is expected to perform similarly since the parameters of the machines are similar. However, as you can see from the rule-base for the fuzzy scheduler, there are some rules in the rule-base of the fuzzy scheduler that are biased toward some part-types. Therefore, when we switch the order of indexing the part-types, the performance of the fuzzy scheduler can be different.

2. Machine load variations:

   (a) Case 3:
   Old machine ($\rho = 0.99286$): $d_1 = 18$, $d_2 = 3$, $d_3 = 1$, $\tau_1 = 1/35$, $\tau_2 = 1/7$, $\tau_3 = 1/20$.
   New machine ($\rho = 0.35758$): $d_1 = 7$, $d_2 = 9$, $d_3 = 3$, $\tau_1 = 1/100$, $\tau_2 = 1/51$, $\tau_3 = 1/27$.
   This is a transition from a high to a low machine load. The value of η of the AFS changes from 1.0993 to 1.0263, as expected. On the other hand, η of CPK changes from 1.0017 to 1.0477 instead of 1.0263. Note that CPK can still perform reasonably well as the machine parameters change from a highly loaded to a lightly loaded machine.

   (b) Case 4:
   Old machine ($\rho = 0.35758$): $d_1 = 7$, $d_2 = 9$, $d_3 = 3$, $\tau_1 = 1/100$, $\tau_2 = 1/51$, $\tau_3 = 1/27$.
   New machine ($\rho = 0.99286$): $d_1 = 18$, $d_2 = 3$, $d_3 = 1$, $\tau_1 = 1/35$, $\tau_2 = 1/7$, $\tau_3 = 1/20$.
   This is a transition from a low to a high machine load. The value of η of the AFS changes from 1.0263 to 1.0993, as expected. On the other hand, η of CPK changes from 1.0263 to 1.106 instead of 1.0017—that is, its performance degrades.

The results show that the AFS we have developed has the capability to maintain good performance even if there were significant changes in the underlying machine parameters (representing, e.g., machine failures). CPK, on the other hand, is dependent on the exact specification of the machine parameters, and hence its performance can degrade if the parameters change. We found similar improvements

in performance as compared to the CLB and CAF policies that are described in Chapter 3.

6.6 Indirect Adaptive Fuzzy Control

In this section we take the "indirect" approach to adaptive fuzzy control where we use an on-line identification method to estimate the parameters of a model of the plant. The estimated model of the plant is then used by a "controller designer" that specifies the controller parameters. See Figure 6.2 on page 319. There is an inherent assumption by the controller designer that the model parameter estimates provided at each time instant represent the plant perfectly. Then the controller designer specifies a controller assuming that this is a perfect model. The resulting control law is called a "certainty equivalence controller" since it was specified by assuming that we were certain that the plant model estimates were equivalent to those of the actual plant.

One strength of the indirect approach is that it is modular in the sense that the design of the plant model identifier can be somewhat independent of the way that we specify the controller designer. In this section we will first introduce two methods from Chapter 5 that can be used for on-line plant model estimation; these are the gradient and least squares methods. Following this we discuss an approach based on feedback linearization, then we introduce the "adaptive parallel distributed compensator" that builds on the parallel distributed compensator from Chapter 4. We close the section with a simple example of how to design an indirect adaptive fuzzy controller.

6.6.1 On-Line Identification Methods

Several of the methods for identification in Chapter 5 were inherently batch approaches so they cannot be used in indirect adaptive control since an on-line adjustment method is needed. For instance, in the learning from examples approaches, there are only methods for adding rules to the system so they do not provide the appropriate adjustment capabilities for achieving the on-line tracking of dynamic changes in the plant. Similar comments can be made about the batch least squares method and the clustering with optimal output predefuzzification methods.

Two methods from Chapter 5 do lend themselves to on-line implementation; these are the recursive least squares and gradient training methods. For each of these we can easily establish an "identifier structure" (i.e., the structure for the model that has its parameters tuned). Then we use the RLS or gradient method to tune the parameters of the model (e.g., the membership function centers). We should note that RLS only allows for tuning parameters that enter the model in a linear fashion (e.g., the output membership function centers), while the gradient method allows for tuning parameters that enter in a nonlinear fashion (e.g., the input membership function widths). It is for this reason that the gradient method may have an enhanced ability to tune the fuzzy system to act like the plant. However, we must emphasize that we will not be providing stability or convergence

results showing that either method will succeed in their tasks. Ours is simply a heuristic construction procedure for the adaptive fuzzy controllers; we cannot say a priori which tuning method to choose.

6.6.2 Adaptive Control for Feedback Linearizable Systems

In the case of conventional indirect adaptive control for linear systems with constant but unknown parameters, the certainty equivalence control law is used and the controller designer may use, for example, "model reference control" [77] or pole placement methods (e.g., LQR or polynomial methods). As the adaptive control problem for linear systems is well studied and many methods exist for that case, we briefly focus here on the use of adaptive control for nonlinear (feedback linearizable) systems.

Following the approach in [189], we assume that our plant is in the form

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t)$$
$$y(t) = h(x(t))$$

where x is the state, u is the input, and y is the output of the plant. Under certain assumptions by differentiating the plant output it is possible to transform the plant model into the form

$$y^{(r)} = \alpha(x(t)) + \beta(x(t))u(t) \tag{6.31}$$

where $y^{(d)}$ denotes the d^{th} derivative of y and d denotes the "relative degree" of the nonlinear plant. If $d < n$ then there can be "zero dynamics" [223], and normally you must assume that these are stable, as we do here. We assume that $\beta(x(t)) \geq \beta_0 > 0$ for all $x(t)$ for some given β_0. Note that if at some $x(t)$ we have $\beta(x(t)) = 0$, then u is not able to affect the system at this state.

For the plant in Equation (6.31) we can use the controller

$$u(t) = \frac{1}{\beta(x(t))}\left(-\alpha(x(t)) + \nu(t)\right) \tag{6.32}$$

where ν will be specified below. Now, if we substitute this control law into Equation (6.31), the closed-loop system will become

$$y^{(d)} = \nu(t)$$

(which is a linear system with input $\nu(t)$). We see that the control law uses feedback to cancel the nonlinear dynamics of the plant and replaces them with $\nu(t)$. Hence, all we have to do is choose $\nu(t)$ so that it represents the kind of dynamics that we would like to have in our closed-loop system. For example, suppose that the relative degree is the same as the order of the plant and is equal to two (i.e., $d = n = 2$). In this case we could choose

$$\nu(t) = ae_o^{(1)}(t) + be_o(t)$$

where $e_o(t) = r(t) - y(t)$, $e_o^{(i)}$ is the i^{th} derivative of $e_o(t)$, $r(t)$ is the reference input, and a and b are design parameters. Notice that with this choice

$$y^{(2)} = ae_o^{(1)}(t) + be_o(t)$$

or

$$y^{(2)} + ay^{(1)} + by(t) = ar^{(1)}(t) + br(t)$$

to make the closed-loop system linear. Hence,

$$\frac{Y(s)}{R(s)} = \frac{as + b}{s^2 + as + b}$$

so that if we pick $a > 0$ and $b > 0$, we will have a stable closed-loop system (you can use the quadratic formula to show this). Also, we see that we can pick a and b to specify the type of closed-loop response we want (i.e., fast, slow, with a specific amount of overshoot, etc.). The same general approach works for higher-order systems (show how by repeating the above analysis for an arbitrary value of $d = n$).

Now, the above design procedure for nonlinear controllers for the nonlinear system in Equation (6.31) assumes that we have perfect knowledge of the plant dynamics (i.e., that we know $\alpha(x(t))$ and $\beta(x(t))$ and the order of the plant). Here, we will assume that we do not know $\alpha(x(t))$ or $\beta(x(t))$ but that we do know that $\beta(x(t)) \geq \beta_0 > 0$ for all $x(t)$ for some given β_0. Then, we will use on-line identifiers to estimate the plant dynamics $\alpha(x(t))$ and $\beta(x(t))$ with $\hat{\alpha}(x(t))$ and $\hat{\beta}(x(t))$, which will be fuzzy systems. With this we use the certainty equivalence control law for the plant in Equation (6.31), which, based on Equation (6.32), would be

$$u(t) = \frac{1}{\hat{\beta}(x(t))} \left(-\hat{\alpha}(x(t)) + \nu(t)\right) \tag{6.33}$$

We will choose $\nu(t)$ the same way as we did above. This control law specifies the "controller designer." That is, it is the recipe for specifying the control law given the estimates of the plant dynamics. Intuitively, we know that if our identifier can do a good job at identifying the dynamics of the plant then it will be possible to achieve the closed-loop behavior (which we characterize above via a and b).

The complete indirect adaptive fuzzy controller consists of an on-line estimator for the fuzzy systems $\hat{\alpha}(x(t))$ and $\hat{\beta}(x(t))$. If recursive least squares is used with a standard fuzzy system, then the output membership function centers of the fuzzy systems $\hat{\alpha}(x(t))$ and $\hat{\beta}(x(t))$ are tuned. If recursive least squares is used with $\hat{\alpha}(x(t))$ and $\hat{\beta}(x(t))$ defined as Takagi-Sugeno fuzzy systems, then the parameters of the output functions are tuned. If a gradient method is used, then all the parameters of the fuzzy systems $\hat{\alpha}(x(t))$ and $\hat{\beta}(x(t))$ (either standard or Takagi-Sugeno) can be tuned to try to make $\hat{\alpha}(x(t)) \to \alpha(x(t))$ and $\hat{\beta}(x(t)) \to \beta(x(t))$ so that a feedback linearizing control law is found and the nonlinear dynamics are replaced by the

designed dynamics specified in $\nu(t)$ (note that even if we do not get this convergence we can often still obtain a successful adaptive controller).

As a final note we must emphasize, however, that there are no guarantees that you will achieve good performance or stable operation with this approach. If you want guarantees, you should study the methods that are shown to achieve stable operation (see For Further Study at the end of this chapter). There, work is described that explains the full details on how to define the fuzzy system parameter update methods and the entire indirect adaptive fuzzy controller that will ensure stable operation.

6.6.3 Adaptive Parallel Distributed Compensation

In Chapter 4 we studied the parallel distributed compensator for Takagi-Sugeno fuzzy systems. For that controller we assumed that either via system identification or modeling we have a Takagi-Sugeno model of the nonlinear plant. From this model we constructed a parallel distributed compensator that could provide a global asymptotically stable equilibrium for the closed-loop system.

Here, we do not assume that the Takagi-Sugeno model of the plant is known a priori. Instead, we use an on-line identification method to adjust the parameters of a Takagi-Sugeno "identifier model" to try to make it match the behavior of the plant. Then, using the certainty equivalence principle, we employ the parameters of the Takagi-Sugeno identifier model in a standard control design method for the standard parallel distributed compensator. In this way, as the identifier becomes more accurate, the controller parameters of the parallel distributed compensator will be adjusted, and if the identifier succeeds in its task, the controller should too.

Suppose that the identifier model is specified by R rules

If $y(k)$ is $\tilde{A}_1^j$ **and**, $\ldots$, **and** $y(k-n+1)$ is $\tilde{A}_n^\ell$
Then $\hat{y}_i(k+1) = \alpha_{i,1}y(k) + \cdots + \alpha_{i,n}y(k-n+1) + \beta_{i,1}u(k) + \cdots + \beta_{i,m}u(k-m+1)$

which have as consequents discrete-time linear systems (see Exercise 4.6 on page 227 for stability results for the discrete-time case). Here, $u(k)$ and $y(k)$ are the plant input and output, respectively; $\tilde{A}_i^j$ is the linguistic value; $\alpha_{i,j}$, $\beta_{i,p}$, $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$, and $p = 1, 2, \ldots, m$ are the parameters of the consequents; and $\hat{y}_i(k+1)$ is the identifier model output considering only rule i. Suppose that μ_i denotes the premise certainty for rule i. Using center-average defuzzification, we get the identifier model output

$$\hat{y}(k+1) = \frac{\sum_{i=1}^{R} \hat{y}_i(k+1)\mu_i}{\sum_{i=1}^{R} \mu_i}$$

Let

$$\xi_i = \frac{\mu_i}{\sum_{i=1}^{R} \mu_i} \tag{6.34}$$

$$\xi = \begin{bmatrix} y(k)\xi_1 \\ y(k)\xi_2 \\ \vdots \\ y(k)\xi_R \\ \vdots \\ u(k-m+1)\xi_1 \\ u(k-m+1)\xi_2 \\ \vdots \\ u(k-m+1)\xi_R \end{bmatrix}, \quad \theta = \begin{bmatrix} \alpha_{1,1} \\ \alpha_{2,1} \\ \vdots \\ \alpha_{R,1} \\ \vdots \\ \beta_{1,m} \\ \vdots \\ \beta_{R,m} \end{bmatrix}$$

so that

$$\hat{y}(k+1) = \theta^\top \xi$$

is the identifier model output. An on-line method such as RLS could adjust the $\alpha_{i,j}$ and $\beta_{i,p}$ parameters since they enter linearly. Gradient methods could be used to adjust the $\alpha_{i,j}$ and $\beta_{i,p}$ parameters and the parameters of the premises (e.g., the input membership function centers and spreads if Gaussian input membership functions are used).

For the controller, we can use Takagi-Sugeno rules of the form

$$\textbf{If } y(k) \text{ is } \tilde{A}_1^j \textbf{ and}, \ \ldots, \textbf{ and } y(k-n+1) \text{ is } \tilde{A}_n^\ell \textbf{ Then } u_i(k) = L_i(\cdot)$$

where $L_i(\cdot)$ is a linear function of its arguments that can depend on past plant inputs and outputs and the reference input, and $u_i(k)$ is the controller output considering only rule i. For example, for some applications it may be appropriate to choose

$$L_i(r(k), y(k)) = k_{i,0} r(k) - k_{i,1} y(k)$$

which is simply a proportional controller with gains $k_{i,0}$ and $k_{i,1}$. For other applications you may need a more complex linear mapping in the consequents of the rules. In any case, the identifier will adjust the gains of the L_i functions using a certainty equivalence approach. For example, the gains $k_{i,0}$ and $k_{i,1}$ could be chosen at each time step to try to meet some stability or performance specifications. In the next section we give an example of how to do this.

6.6.4 Example: Level Control in a Surge Tank

In this section we use a level control problem for a surge tank to show how to design an indirect adaptive fuzzy controller using the adaptive parallel distributed compensation approach. In particular, suppose that you are given the "surge tank" that is shown in Figure 6.44.

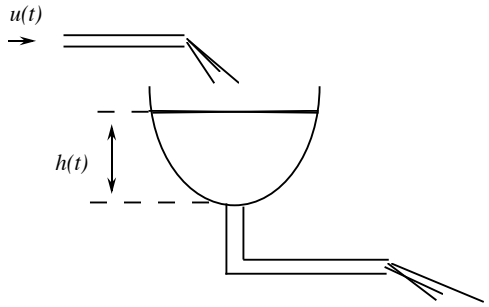


FIGURE 6.44 Surge tank.

The differential equation representing this system is

$$\frac{dh(t)}{dt} = \frac{-c\sqrt{2gh(t)}}{A(h(t))} + \frac{1}{A(h(t))}u(t)$$

where $u(t)$ is the input flow (control input), which can be 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; $g = 9.8\text{m/sec}^2$ is gravity; and $c = 1$ is the known cross-sectional area of the output pipe. Let $r(t)$ be the desired level of the liquid in the tank (the reference input). Assume that $h(0) = 1$. Also assume that a is unknown but that $a \in [a_1, a_2]$, $a_1 \geq 0$, for some fixed real numbers a_1 and a_2, and that $A(h) = ah^2 + b$ where $a \in [a_1, a_2]$, $a_1 \geq 0$, and $b \in [b_1, b_2]$, $b_1 > 0$, where $a_1 = 0.5$, $a_2 = 4$, $b_1 = 1$, $b_2 = 3$ are all fixed.

First, we will choose $a = 1$ and $b = 2$ as the nominal plant parameters. Also, since we will be using a discrete-time identifier, we discretize the plant and use it in all our simulations. In particular, using an Euler approximation

$$h(k+1) = h(k) + T\left[\frac{-\sqrt{19.6h(k)}}{h^2(k)+2} + \frac{1}{h^2(k)+2}u(k)\right]$$

where $T = 0.1$. We have additional restrictions on the plant dynamics. In particular, we assume the plant input saturates at ± 50 so that if the controller generates an input $u'(k)$

$$u(k) = \begin{cases} 50 & \text{if} \quad u'(k) > 50 \\ u'(k) & \text{if} \quad -50 \leq u'(k) \leq 50 \\ -50 & \text{if} \quad u'(k) < -50 \end{cases}$$

Also, to ensure that the liquid level never goes negative (which is physically impos-

sible), we simulate our plant using

$$h(k+1) = \max\left\{0.001, h(k) + T\left[\frac{-\sqrt{19.6h(k)}}{h^2(k)+2} + \frac{1}{h^2(k)+2}u(k)\right]\right\}$$

We can use either gradient or RLS to tune the parameters of the Takagi-Sugeno fuzzy system that we use as our identifier model. Here, we use the RLS method to tune to the consequent parameters and specify a priori the parameters that define the premise certainties. In particular, we use only five rules ($R = 5$), $n = 1$, and $m = 1$, in the last section. Hence, one rule of our identifier model would be

$$\textbf{If } h(k) \text{ is } \tilde{A}_1^j \textbf{ Then } \hat{h}_i(k+1) = \alpha_{i,1}h(k) + \beta_{i,1}u(k)$$

We use Gaussian input membership functions on the $h(k)$ universe of discourse of the form

$$\mu(h(k)) = \exp\left\{-\frac{1}{2}\left(\frac{h(k) - c_i^j}{\sigma}\right)^2\right\}$$

where $c_1^1 = 0$, $c_1^2 = 2.5$, $c_1^3 = 5$, $c_1^4 = 7.5$, $c_1^5 = 10$, and $\sigma = 0.5$. Notice that since there is only one input, $\mu_i(h(k)) = \mu(h(k))$; that is, the membership function certainty is the premise membership function certainty for a rule. Also, if $h(k) \leq c_1^1$, then we let $\mu_1(h(k)) = 1$, and if $h(k) \geq c_1^5$, then we let $\mu_5(h(k)) = 1$. This causes saturation of the outermost input membership functions. With center-average defuzzification our fuzzy system is

$$\hat{h}(k+1) = \theta^\top \xi(h(k))$$

where ξ_i is defined in Equation (6.34) and

$$\xi(h(k)) = \begin{bmatrix} h(k)\xi_1 \\ h(k)\xi_2 \\ \vdots \\ h(k)\xi_R \\ u(k)\xi_1 \\ u(k)\xi_2 \\ \vdots \\ u(k)\xi_R \end{bmatrix}, \quad \theta = \begin{bmatrix} \alpha_{1,1} \\ \alpha_{2,1} \\ \vdots \\ \alpha_{5,1} \\ \beta_{1,1} \\ \beta_{2,1} \\ \vdots \\ \beta_{5,1} \end{bmatrix}$$

We use a nonweighted (i.e., $\lambda = 1$) RLS algorithm with update formulas given by

$$P(k+1) = \frac{1}{\lambda}(I - P(k)\xi(h(k))(\lambda I + (\xi(h(k)))^\top P(k)\xi(h(k)))^{-1}(\xi(h(k)))^\top)P(k)$$

$$\theta(k+1) = \theta(k) + P(k)\xi(h(k))(h(k+1) - (\xi(h(k)))^{\top}\theta(k))$$

Notice that we have adjusted the time indices in these equations so that they solve the identification problem of trying to estimate the output of the identifier model (i.e., $h(k+1)$). We choose $\theta(0) = [0, 2, 4, 6, \ldots, 18]^{\top}$ and $P(0) = P_{\beta}(0) = 2000I$ where I is the identity matrix. The choice of $\theta(0)$ is simply one that is not close to the final tuned values (to see this consider the rules of the Takagi-Sugeno fuzzy system for this case and how, based on our understanding of the dynamics of a tank, how these could not properly represent it).

Our controller that is tuned is given by

$$u(k) = \frac{\sum_{i=1}^{R} u_i(k)\mu_i}{\sum_{i=1}^{R} \mu_i}$$

where we choose

$$u_i(k) = L_i(r(k), h(k)) = k_{i,0}r(k) - k_{i,1}h(k)$$

Using a certainty equivalence approach for the parallel distributed compensator, we view each rule of the controller as if it were controlling only one rule of the plant, and we assume that the identifier is accurate. In particular, we assume that $\hat{h}(k) = h(k)$ and $\hat{h}_i(k) = h_i(k)$, where $h_i(k)$ represents the i^{th} component of the plant model (assuming it can be split this way), so that the identifier is also perfectly estimating the dynamics represented by each rule in the plant. If the plant is operating near its i^{th} rule and there is little or no affect from its other rules, then $h(k) = h_i(k)$ so

$$\hat{h}_i(k+1) = h_i(k+1) = \alpha_{i,1}h_i(k) + \beta_{i,1}\left[k_{i,0}r(k) - k_{i,1}h_i(k)\right] \tag{6.35}$$

We pick $k_{i,0}$ and $k_{i,1}$ for each $i = 1, 2, \ldots, R$ so that the pole of the closed-loop system is at 0.1 and the steady-state error between $h(k)$ and $r(k)$ is zero. In particular, if $H_i(z)$ and $R(z)$ are the z-transforms of $h_i(k)$ and $r(k)$, respectively, then

$$\frac{H_i(z)}{R(z)} = \frac{\beta_{i,1}k_{i,0}}{z + \beta_{i,1}k_{i,1} - \alpha_{i,1}}$$

Choose

$$k_{i,1}\beta_{i,1} - \alpha_{i,1} = -0.1$$

to get the placement of the pole so that our controller designer in our indirect adaptive scheme will pick

$$k_{i,1} = \frac{\alpha_{i,1} - 0.1}{\beta_{i,1}} \tag{6.36}$$

$i = 1, 2, \ldots, R$, at each time step using the estimates of $\alpha_{i,1}$ and $\beta_{i,1}$ from the identifier. Notice that we must ensure that $\beta_{i,1} > 0$ and we can do this by specifying

a priori some $\underline{\beta} > 0$ and adding a rule to the adaptation scheme that says that if at some time the RLS updates any $\beta_{i,1}$ so that it becomes less than $\underline{\beta}$, we let it be equal to $\underline{\beta}$. In this way the lowest value that $\beta_{i,1}$ will take is $\underline{\beta}$. Another way to specify the update method for the $k_{i,1}$ (and $k_{i,0}$ below) would be to use the stability conditions for the parallel distributed compensator from Chapter 4.

Next, we want a zero steady-state error, so we want $h_i(k+1) = h_i(k) = r(k)$ for large k and all $i = 1, 2, \ldots, R$, so from Equation (6.35) we want

$$1 = \alpha_{i,1} + \beta_{i,1} k_{i,0} - \beta_{i,1} k_{i,1}$$

so our controller designer will choose

$$k_{i,0} = \frac{1 - \alpha_{i,1} + \beta_{i,1} k_{i,1}}{\beta_{i,1}} \tag{6.37}$$

$i = 1, 2, \ldots, R$. Equations (6.36) and (6.37) specify the controller designer for the indirect adaptive scheme, and the identifier will provide the values of $\alpha_{i,1}$ and $\beta_{i,1}$ at each time step so that the $k_{i,0}$ and $k_{i,1}$ can be updated at each time step. Notice that the modifications to Equation (6.36) with $\underline{\beta}$ above will also ensure that we will not divide by zero in Equation (6.37).

The results of the RLS-based adaptive parallel distributed compensator are shown in Figure 6.45. Notice that the output of the plant $h(k)$ tracks the reference input $r(k)$ quite well. Next, we show values of $\hat{h}(k)$ and $h(k)$ in Figure 6.46. Notice that with only five rules in the identifier model we get a reasonably good estimate of $h(k)$, and hence we can see why our closed-loop response tracks the reference input so well.

6.7 Summary

In this chapter we have provided an overview of several adaptive fuzzy control methods. First, we introduced the fuzzy model reference learning controller and provided several guidelines for how to design it. We showed three case studies in design and implementation, including the ship steering problem where we compared it against some conventional model reference adaptive control methods. We showed how it could be used for fault-tolerant aircraft control, and provided implementation results for the flexible-link robot.

Next, we showed how "dynamically focused learning" could be used for the FMRLC. We introduced the three DFL strategies (auto-tuning, auto-attentive, and auto-attentive with memory) via a simple academic magnetic levitation control problem. We provided two case studies in DFL design and implementation. In particular, we showed how to design auto-tuning mechanisms for direct fuzzy controllers for the rotational inverted pendulum and the machine scheduling problems studied in Chapter 3.

Finally, we introduced indirect adaptive fuzzy control. There, we discussed indirect adaptive fuzzy control for feedback linearizable systems, introduced the

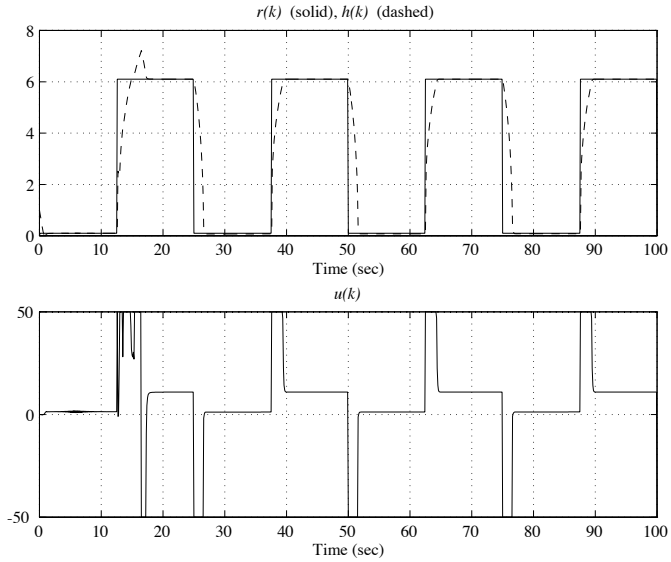


FIGURE 6.45 Response of RLS-based adaptive parallel distributed compensator for the tank (plot produced by Mustafa K. Guven).

adaptive parallel distributed compensator, and studied a tank application.

The overall approach that we take to adaptive control in this chapter is a heuristic one as opposed to a mathematical one. We focused on the use of intuition to motivate why adaptation is needed, and we provided natural extensions to the direct fuzzy control methods described in Chapters 2 and 3.

Upon completing this chapter, the reader should understand the following topics:

- Basic schemes for adaptive control (e.g., model reference adaptive control and direct versus indirect schemes).
- Fuzzy model reference learning control (FMRLC).
- Design methods for the FMRLC.
- Issues in adaptive versus learning control.
- The issues that must be considered in comparing or evaluating conventional versus adaptive fuzzy control.
- How failures can be viewed as a significant plant variation that can be accommodated for via adaptive control.

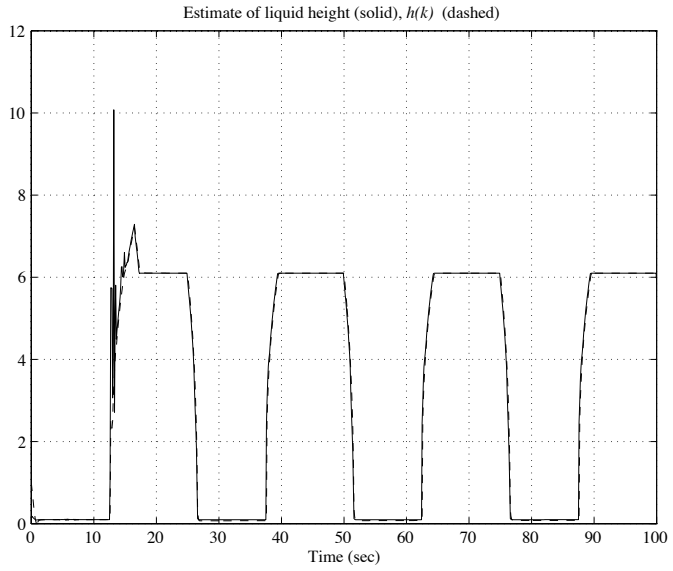


FIGURE 6.46 The values of $\hat{h}(k)$ and $h(k)$ for the RLS-based adaptive parallel distributed compensator for the tank (plot produced by Mustafa K. Guven).

- Dynamically focused learning (DFL) strategies (auto-tuning, auto-attentive, and auto-attentive with memory).
- How to apply DFL strategies to both the FMRLC and a direct fuzzy controller.
- The concept of a certainty equivalence control law.
- How the RLS and gradient methods can be used for on-line identification of a plant model and hence used in indirect adaptive fuzzy control (and in direct adaptive control for identification of a controller; see Design Problem 6.10).
- The feedback linearization and adaptive parallel distributed compensation approaches to indirect adaptive fuzzy control.
- How to construct an indirect adaptive fuzzy controller for a surge tank.

Essentially, this is a checklist of the major topics of this chapter. We encourage the reader to test the adaptive fuzzy control methods by doing some problems at the end of the chapter or by trying them out on your own applications. Additional adaptive methods are introduced in the next chapter.

6.8 For Further Study

The reader wishing to strengthen her or his background in conventional adaptive control should consult [77, 180, 149, 11]. The FMRLC algorithm was first introduced in [111, 112] and grew from research performed on the linguistic self-organizing controller (SOC) presented in [170] (with applications in [181, 214, 78, 40, 39, 38, 239]) and ideas in conventional "model reference adaptive control" (MRAC) [149, 11]. The ship steering application is described in [11, 149] which are the sources for the problem formulation for this chapter. The ship steering section, where the FMRLC is used, is based on [112]. Examples of Lyapunov-based MRAC designs are illustrated in [149, 220]; in the case study in this chapter we use the approach in [149] to design our Lyapunov-based MRAC. The fault-tolerant aircraft control section is based on [104], and the application to the two-link flexible robot is based on [144]. The FMRLC has also been used for a robotics problem and a rocket velocity control problem [113], a cart-pendulum system [111], the control of an experimental induction machine, an automated highway system, in automobile suspension control, for liquid level control in a surge tank [251], for the rotational inverted pendulum of Chapter 3, for a single-link flexible robot, and for a ball on a beam experiment; it has also been used to improve the performance of antiskid brakes on adverse road conditions [114].

The section on dynamically focused fuzzy learning control is based on [103]. For a discussion on how to program rule-base shifts and a detailed analysis of the computational complexity of the DFL strategies relative to conventional MRAC, see [103]. The section on the rotational inverted pendulum is based on [235], and the adaptive machine scheduling work is based on [6]. While in this chapter we have considered only the single machine case (and only for a limited number of types of machines in simulation), in [6] the authors show how to use the the AFS on each machine in a network of machines to improve overall scheduling performance of a flexible manufacturing system (FMS). Basically, this is done by using the fuzzy scheduler, adaptation mechanism, and USSM on each machine in the network. Generally, we can find topologies of FMS for which the AFS can fine-tune themselves so that they optimize certain performance measures (e.g., minimizing the maximum backlog in the FMS). This often involves having the local schedulers on some machines sacrifice their local throughput performance to make it possible to achieve higher performance for the entire FMS. See [6] for more details.

Other alternatives to FMRLC, DFL, and SOC are contained in [47, 87, 26, 229] and in [17, 208, 184]. Other work is also given in [173], where the authors present a knowledge-based fuzzy control system that is constructed off-line. Another example of indirect adaptive fuzzy control presented by Graham and Newell in [62, 61] uses a fuzzy identification algorithm developed by Czogała and Pedrycz [36, 37] to identify a fuzzy process model that is then used to determine the control actions. Batur and Kasparian [19] present a methodology to adapt the initial knowledge-base of a fuzzy controller to changing operating conditions. The output membership functions of their fuzzy controller are adjusted in response to the future or past performance of the overall system, where the prediction is obtained through a linear process model

updated by on-line identification. Other indirect approaches to adaptive fuzzy control are shown in [26, 224]. In addition to all these, there are many other adaptive fuzzy system applications that, to name a few, choose to use neural networks for identification and reinforcement learning [26, 126] or genetic algorithms for natural selection of controller parameters as the learning mechanism (for some references on the genetic algorithm approach to tuning fuzzy systems see the For Further Study section at the end of Chapter 8).

Recent work on adaptive fuzzy systems has focused on merging concepts and techniques from conventional adaptive systems into a fuzzy systems framework and performing stability analysis to guarantee properties of the operation of adaptive fuzzy control systems. For example, the reader could consider the work in [229] and the references contained therein. More recent work on the development of stable direct and indirect adaptive fuzzy controllers is contained in [85, 200, 195, 196, 198, 197, 194, 199, 202, 201, 203] (for a more complete treatment of the literature see the references in [200]). The reader interested in the feedback linearization approach to indirect (and direct) adaptive fuzzy control in Section 6.6.2 starting on page 395 should consult these references.

This is a fairly representative sampling of the literature in adaptive fuzzy control; we emphasize, however, that it is not complete as the amount of literature in this area is quite large, especially considering the many heuristic adaptive approaches that are used for all types of applications.

6.9 Exercises

Exercise 6.1 (FMRLC Programming and Operation): In this problem we will consider an academic control problem for the purpose of focusing on how to program the operation of the FMRLC. In this way we focus only on the operation of the FMRLC and avoid application-specific details. Suppose that you are given a plant

$$G(s) = \frac{k_p}{s^2 + 2\zeta_p \omega_p s + w_p^2}$$

where k_p is the gain in the numerator, ζ_p is the damping ratio, and ω_p is the undamped natural frequency (all unknown but constant). Nominally, we have $k_p = 1$, $\zeta_p = 0.707$, and $\omega_p = 1$. The reference model is a first-order system

$$M(s) = \frac{k_r}{s + a_r}$$

where $k_r = 1$ and $a_r = 1$. Use $r(t) = \sin(0.6t)$ as the reference input.

(a) Design a direct (nonadaptive) fuzzy controller for the plant that as nearly as possible achieves the performance specified in the reference model. Use two inputs to the fuzzy controller, the error $e = r - y$ where r is the reference input and y is the plant output, and the change in this error (i.e., its

derivative approximated by a backward difference).

(b) Design an FMRLC for the plant to make it track the reference model as closely as possible. Use the fuzzy controller designed in (a) to initialize the controller that the FMRLC tunes. Provide the tuned rule-base at the end of the simulation and compare it to the rule-base from (a). Is the FMRLC still tuning the rule-base at the end of the simulation (i.e., are the output centers still moving)?

(c) Repeat (b) except initialize the direct fuzzy controller the same as in (a) except let all the output membership function centers be zero. Provide the tuned rule-base that results at the end of the simulation, and compare it to the ones from (a) and (b).

Exercise 6.2 (Rule-Base Update Procedure for the FMRLC): Consider the rule-base update procedure for the FMRLC given by Equations (6.1) and (6.2). Notice that for our development, when COG is used, this update will guarantee that the previous input would have been $u(kT - T) + p(kT)$. Prove this for both the center-average and COG defuzzification methods.

Exercise 6.3 (Dynamically Focused Learning Methods): In this problem you will explain various characteristics of the DFL methods used in the chapter and propose how to extend these methods.

(a) Explain in words the analogies between the auto-tuning, auto-attentive, and auto-attentive with memory strategies and the way that humans appear to learn.

(b) We have used the auto-tuning method for both tuning a direct fuzzy controller for the rotational inverted pendulum and for adaptive machine scheduling. Suppose that you design an adaptive fuzzy controller by simply using the auto-tuning strategy for a direct fuzzy controller. Suppose that you now want to augment this learning strategy with an ability to remember the control map in regions where it has operated in the past (so you use the auto-tuning method as the basic fuzzy learning controller and add a certain type of fuzzy system experience model). Explain in words how to do this (use rule-base diagrams in your explanation similar to those used in the chapter).

(c) Explain in words how you could add a successive hierarchy of auto-attentive with memory strategies.

6.10 Design Problems

Design Problem 6.1 (FMRLC for Cargo and Tanker Ship Steering): In this problem we will study the use of the FMRLC for steering various ships.

(a) Verify the results in the chapter for the cargo ship by simulating the FMRLC under the same conditions. In particular, generate the plots shown in Figure 6.6.

(b) In this part you will study the steering of a "tanker ship" defined in [11]. The tanker (under "ballast" conditions) has the same differential equations describing its behavior as the cargo ship, but it has $K_0 = 5.88$, $\tau_{10} = -16.91$, $\tau_{20} = 0.45$, and $\tau_{30} = 1.43$, and its length is $l = 350$ (assume that the ship velocity is $u = 5$). Assume that the rudder input to the tanker saturates at $\pm\pi/2$, but also assume that this is the only way that the rudder movement is constrained (i.e., do not assume that there are extra actuator dynamics or a rate limiter on the rudder's movement). Design an FMRLC for the tanker. Initialize it either with your best direct fuzzy controller or in the way we did for the cargo ship in the chapter. It is your responsibility to choose a reasonable reference model, and you must justify your choice. Is the one for the cargo ship appropriate? If you believe so, then justify this choice. Next, design the learning mechanism similar to how we did for the cargo ship but keep in mind that the tanker has somewhat different dynamics so that you may need to tune the fuzzy inverse model. Use the design procedures provided in the chapter to tune the FMRLC.

(c) Evaluate the performance of the tanker closed-loop system using a simulation. Does the system appear to be stable? Is the rise-time, overshoot, and settling time adequate for your system? Use a reference input that commands $+45°$ for 500 sec, then commands $0°$ for 500 sec, then commands $-45°$ for 500 sec, then commands $0°$ for the next 500 sec. Then repeat this 2000-sec sequence at least twice so that the FMRLC has adequate time to adapt.

(d) Study the effects of plant parameter variations and the ability of the tanker control system to adapt to these changes. In particular, for the tanker run a simulation for 2000 sec, and then suddenly at $t = 2000$ sec switch the parameters of the tanker to $K_0 = 0.83$, $\tau_{10} = -2.88$, $\tau_{20} = 0.38$, $\tau_{30} = 1.07$ (this represents the tanker being "full" rather than "ballast," i.e., a weight change on the ship) and study how the FMRLC adapts to such a plant parameter variation.

(e) Study the effects of changing the speed of the ship during the simulations (note that increasing the speed will likely make it possible to improve the performance, while decreasing the speed will make it more difficult to follow the reference trajectory). For example, increasing the speed to $u = 7$ m/sec typically makes the ship a bit easier to control, while changing it to $u = 3$ m/sec makes it much harder to control. If under the speed changes the tanker does not perform too well, tune the controller until acceptable performance is achieved (but make sure it works for the original parameters also—that is, that the values you tune the parameters to do not destroy the performance that you were able to achieve for (c)).

(f) Study the effects of a wind disturbance on the rudder of the tanker as we did in the chapter for the cargo ship.

Design Problem 6.2 (FMRLC for a Surge Tank): Suppose that you are given the surge tank system that is shown in Figure 6.44 on page 399. Recall that the differential equation representing this system is

$$\frac{dh(t)}{dt} = \frac{-c\sqrt{2gh(t)}}{A(h(t))} + \frac{1}{A(h(t))}u(t)$$

where $u(t)$ is the input flow (control input), which can be positive or negative (i.e., it can also pull liquid out of the tank); $h(t)$ is the liquid level (the output $y(t) = h(t)$); $A(h(t))$ is the cross-sectional area of the tank; $g = 9.8\text{m/sec}^2$ is gravity; and $c = 1$ is the known cross-sectional area of the output pipe. Let $h_d(t)$ be the desired level of the liquid in the tank. Assume that $h(0) = 0$. Also assume that a is unknown but that $a \in [a_1, a_2]$, $a_1 \geq 0$, for some fixed real numbers a_1 and a_2, and that $A(h) = ah^2 + b$ where $a \in [a_1, a_2]$, $a_1 \geq 0$, and $b \in [b_1, b_2]$, $b_1 > 0$, where $a_1 = 0.5$, $a_2 = 4$, $b_1 = 1$, $b_2 = 3$ are all fixed. For this problem assume that the input to the plant is saturated at ± 50.

(a) Choose $a = 1$ and $b = 2$ as the nominal plant parameters. Choose a reference model for the system and justify your choice (i.e., explain why it is a reasonable performance request yet one that would represent good performance objectives).

(b) Design an FMRLC for the tank. Use a PD fuzzy controller as the controller that is tuned, and provide the full details on the rationale for your choice of the fuzzy inverse model. Fully explain all of the design choices that you make for the FMRLC (including how you initialize it).

(c) Next, use the following reference input signals to test the performance of the FMRLC: (i) a step input of amplitude 4; (ii) a sinusoid $2 + 2\sin(\frac{\pi}{20}t)$; (iii) a sinusoid $2 + 2\sin(\frac{2\pi}{5}t)$; and (iv) a square wave with a period of $T = 20$ seconds, with a peak-to-peak amplitude of 4, but a constant term of two added to it. Provide simulations that illustrate the performance of your FMRLC for each of these cases.

(d) In this part, you will study how the FMRLC performs when the plant parameters change. In particular, from $t = 30$ seconds to $t = 35$ seconds parameter a decreases from 1 to 0.5, and parameter b decreases linearly from 2 to 1. After that, from $t = 35$ seconds to $t = 45$ seconds 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. Simulate the system using reference inputs (i), (ii), and (iii) from part (c) above to illustrate the performance of the FMRLC. Hint: It is possible to tune the FMRLC to overcome the effects of the plant parameter variations so that performance similar to that obtained in (c) is obtained. It is your objective to tune it so that this is the case.

(e) Let $a = 4$ and $b = 3$ be fixed. Let the reference input be a step of amplitude 7. Simulate the system to illustrate the performance of the FMRLC. Compare to (c) above. Discuss.

Design Problem 6.3 (FMRLC for Rocket Velocity Control): In this problem you will develop an FMRLC that is used to control the velocity of the single-stage rocket that is described in Design Problem 3.8 on page 180.

(a) Design a discrete FMRLC ($T = 100$ milliseconds). Provide all your design variables. Notice that for the design of the inverse model for the rocket process, an increase in the exhaust gas velocity will generally result in an increase in the process output.

(b) Pick an altitude trajectory $y(t)$ and simulate the closed-loop system to verify the performance (i.e., that the output tracks the reference model output).

Design Problem 6.4 (FMRLC for the Base Braking Control Problem): This problem focuses on the development of an FMRLC for the base braking control problem described in Design Problem 3.7 on page 177. You should design your FMRLC so that it is able to track the reference torque trajectory within less than 100 ft-lbs over the entire operation of the brakes with zero steady-state error, even when they heat up and cool down while the FMRLC is operating (use the reference input specified in Chapter 3). Provide a three-dimensional plot of the fuzzy controller surface (the one being tuned) at $t = 0, 8, 16$, and 24 seconds. Explain the connection between the changing shape of the surface and the objectives the FMRLC is trying to achieve.

Design Problem 6.5 (FMRLC and Auto-Tuning for the Rotational Inverted Pendulum): In this problem you will develop an FMRLC and an auto-tuner for the rotational inverted pendulum.

(a) Design an FMRLC to be the balancing controller for the rotational inverted pendulum that is studied in Chapter 3. You should seek to obtain a performance comparable to that achieved in implementation.

(b) Design an auto-tuned direct fuzzy controller for the rotational inverted pendulum that is studied in Chapter 3. Use the same approach as we did in the case study in this chapter (but simply study the control for the nominal case).

Design Problem 6.6 (Adaptive Machine Scheduling via Auto-Tuning): In this problem you will take the same approach as in the chapter to design an adaptive scheduler for a machine.

(a) Simulate the AFS for machine 3 (see Chapter 3) and show how each M_p is adjusted for the first 14 production runs. In particular, show that for later production runs, M_1 remains around 12.338, M_2 around 14.9886, and M_3 around 5.1753. Use the same type of rule-base as the one in the chapter and five membership functions on each input universe of discourse. Let $M_p = 1$ for all p initially.

(b) Show that after 10,000 production runs, $\eta = 1.031$ (which is the same as that produced by CPK after 10,000 production runs).

Design Problem 6.7 (DFL for the Magnetically Suspended Ball): In this problem you will study DFL for the magnetically suspended ball studied in this chapter.

(a) Design two FMRLCs for the magnetically levitated ball problem studied in Section 6.4—one that fails and one that succeeds, just as we did in Section 6.4. Use the exact same FMRLCs as we used in the chapter (i.e., reproduce the results there).

(b) Augment the FMRLC with the auto-tuning mechanism and reproduce the results in the chapter for this case.

(c) Repeat (b) but for the auto-attentive mechanism.

(d) Repeat (b) but for the auto-attentive mechanism with memory.

Design Problem 6.8 (Indirect Adaptive Fuzzy Control for a Tank: RLS Identifier): Repeat the design of the indirect adaptive fuzzy controller (using the adaptive parallel distributed compensation approach) for the tank that was presented in Section 6.6.4 on page 398, simulate the closed-loop system, and provide plots like those shown.

Design Problem 6.9 (Indirect Adaptive Fuzzy Control: Gradient Identifier)*: Design an indirect adaptive fuzzy controller for the tank that was presented in Section 6.6.4 on page 398 using the gradient approach for tuning of the membership functions and the consequent parameters. Use the adaptive parallel distributed compensator approach. Simulate the system to show that you achieve good performance. Fully specify your algorithm and give values for all the design parameters.

Design Problem 6.10 (Alternatives for Direct Adaptive Fuzzy Control)*: Aside from their use in indirect adaptive control, on-line identification techniques can be used for the direct tuning of a controller by using plant data.

(a) Design a direct adaptive control scheme for the surge tank in Design Problem 6.2. To do this, use the gradient approach to identification and simply specify the adaptive control algorithm equations (you do not have to simulate it).

(b) Repeat part (a) except use the RLS-based identifier.

Note that you may want to see the references in the For Further Study section of this chapter to complete this problem.

CHAPTER 7

Fuzzy Supervisory Control

If we could first know where we are,
and whither we are tending,
we could better judge what to do,
and how to do it.
–Abraham Lincoln

7.1 Overview

This chapter focuses on yet another set of adaptive system and control methods; in this sense they can be grouped with those of Chapters 5 and 6 and in some cases can be viewed as extensions or augmentations of those methods. The general approach in this chapter is, however, distinctly different from that of past chapters in that we study methods to *supervise* existing controllers, whether they be fuzzy or conventional ones, adaptive or nonadaptive.

In this chapter we study a multilayer (hierarchical) controller with the supervisor at the highest level, as shown in Figure 7.1. The supervisor can use any available data from the control system to characterize the system's current behavior so that it knows how to change the controller and ultimately achieve the desired specifications. In addition, the supervisor can be used to integrate other information into the control decision-making process. It can incorporate certain user inputs, or inputs from other subsystems. For example, in an automotive cruise control problem, inputs from the driver (user) may indicate that she or he wants the cruise controller to operate either like a sports car or more like a sluggish family car. The other subsystem information that a supervisor could incorporate for supervisory control for an automotive cruise control application could include data from the engine that would help integrate the controls on the vehicle (i.e., engine and cruise control

integration). Given information of this type, the supervisor can seek to tune the controller to achieve higher performance operation or a performance that is more to the liking of the driver.

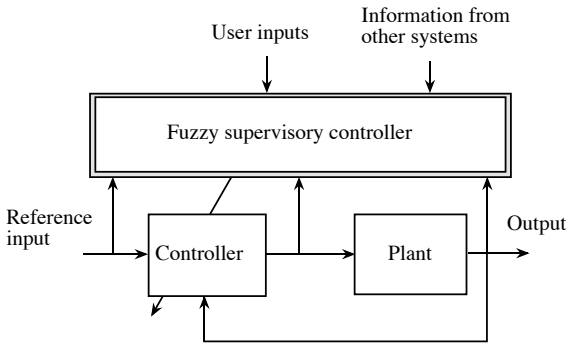


FIGURE 7.1 Supervisory controller.

Conceptually, the design of the supervisory controller can then proceed in the same manner as it did for direct fuzzy controllers: either via the gathering of heuristic control knowledge or via training data that we gather from an experiment. The form of the knowledge or data is, however, somewhat different than in the simple fuzzy control problem. For instance, the type of heuristic knowledge that is used in a supervisor may take one of the following two forms:

1. Information from a human control system operator who observes the behavior of an existing control system (often a conventional control system) and knows how this controller should be tuned under various operating conditions.

2. Information gathered by a control engineer who knows that under different operating conditions controller parameters should be tuned according to certain rules.

Some may view the distinction between a supervisory controller and a conventional direct fuzzy controller as quite subtle, but this is not the case. Basically, the distinction in the past has been simply to call it a "supervisor" if its focus is on the use of inputs that are not traditionally inputs to, for example, a PID controller. Furthermore, the outputs of the supervisor are most often not direct command inputs to the plant. Rather, they dictate changes to a controller that generates these command inputs. Supervisory control is a type of adaptive control since it seeks to observe the current behavior of the control system and modify the controller to improve the performance. In supervisory control, however, we do not constrain ourselves to the conventional structures and methods of adaptive control (e.g., direct or indirect). In fact, in some cases a supervisor is used to tune an adaptive

controller. Supervisory control seeks to integrate a whole variety of information and bring it to bear on a control problem.

This chapter is broken into two main parts: supervision of conventional controllers in Section 7.2, and supervision of fuzzy controllers in Section 7.3. For the supervision of conventional controllers, we begin by discussing the supervision of PID controllers and how the supervisor can act as a gain scheduler. For the supervision of fuzzy controllers, we first study the supervision of direct fuzzy controllers where the supervisor simply chooses between different fixed rule-bases. We also study how to supervise the operation of an adaptive fuzzy controller. Applications to the two-link flexible robot from Chapter 3 (which was also studied in Chapter 6), and the fault-tolerant aircraft control problem from Chapter 6 are used to illustrate these ideas.

The techniques discussed in this chapter are quite pragmatic, often useful in industry, and help to show some more of the fundamental advantages of fuzzy control. This chapter augments the previous adaptive methods with a new set of ideas; hence, to fully understand this chapter, many of the topics in Chapters 5 and 6 must be fully understood. In this sense, for a full understanding of this chapter, all the previous chapters, except Chapter 4, but including the parallel distributed compensator, need to be studied before this one. At the same time, the high-level concepts (e.g., the controller block diagrams) can be understood quite easily once the reader understands the basics of fuzzy control as described in Chapter 2. Hence, we recommend that the reader at least skim this chapter after completing Chapter 2, focusing on the ideas characterized by the controller block diagrams.

7.2 Supervision of Conventional Controllers

Most controllers in operation today have been developed using conventional control methods. There are, however, many situations where these controllers are not properly tuned and there is heuristic knowledge available on how to tune them while they are in operation. There is then the opportunity to utilize fuzzy control methods as the supervisor that tunes or coordinates the application of conventional controllers.

7.2.1 Fuzzy Tuning of PID Controllers

Over 90% of the controllers in operation today are PID controllers. This is because PID controllers are easy to understand, easy to explain to others, and easy to implement. Moreover, they are often available at little extra cost since they are often incorporated into the programmable logic controllers (PLCs) that are used to control many industrial processes. Unfortunately, many of the PID loops that are in operation are in continual need of monitoring and adjustment since they can easily become improperly tuned.

Suppose for the sake of discussion that a PID controller is placed in a unity feedback control system where the input to the plant is u, the plant output is y, the reference input is r, and the error input to the PID controller is $e = r - y$. The

basic form for the PID controller is

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

where K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain.

Because PID controllers are often not properly tuned (e.g., due to plant parameter variations or operating condition changes), there is a significant need to develop methods for the automatic tuning of PID controllers. While there exist many conventional methods for PID auto-tuning, here we will strictly focus on providing the basic ideas on how you would construct a fuzzy PID auto-tuner. The basic approach is summarized in Figure 7.1, where the supervisor is trying to recognize when the controller is not properly tuned and then seeks to adjust the PID gains to obtain improved performance. The other user inputs or subsystem inputs may be particularly useful in practical situations to, for example, provide the supervisor with an indication that there will be different effects on the control system that will cause it to become out of tune.

A more detailed figure showing how a fuzzy PID auto-tuner may work is shown in Figure 7.2. There, the "behavior recognizer" seeks to characterize the current behavior of the plant in a way that will be useful to the PID designer. The whole (upper-level) supervisor may be implemented in a similar fashion to an indirect adaptive controller as described in Chapter 6. Alternatively, simple tuning rules may be used where the premises of the rules form part of the behavior recognizer and the consequents form the PID designer. In this way, a single fuzzy system is used to implement the entire supervisory control level. Some candidate rules for such a fuzzy system may include the following:

- **If** steady-state error is large **Then** increase the proportional gain.
- **If** the response is oscillatory **Then** increase the derivative gain.
- **If** the response is sluggish **Then** increase the proportional gain.
- **If** the steady-state error is too big **Then** adjust the integral gain.
- **If** the overshoot is too big **Then** decrease the proportional gain.

Alternatively, you could use the Zeigler-Nichols PID tuning rules [54] but represent them with a fuzzy system.

We would like to emphasize, however, that while such rules appear to offer a very simple solution to the PID auto-tuning problem, it is not always easy to measure the quantities that are listed in the premises above. For example, when is the system in steady-state operation? How do we precisely quantify "sluggish" behavior? How do we measure a degree of oscillatory behavior? How do we measure the overshoot on-line when we cannot be guaranteed that the system will have a step

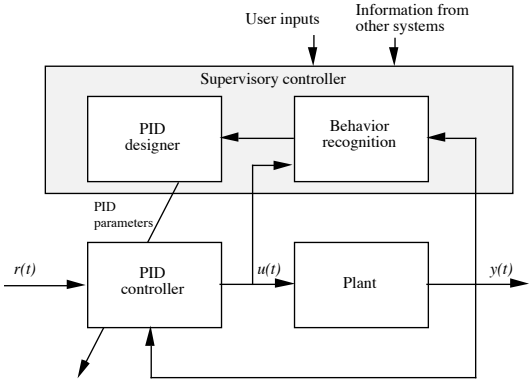


FIGURE 7.2 Fuzzy PID auto-tuner.

for a reference input? The answers to these questions tend to be very application-dependent. Overall, we see that the behavior recognizer will also need to contain some algorithms for preprocessing data from the plant before it can be used in the above rules.

For some applications, it is best to have a tuning phase and an on-line operation phase where tuning does not occur. In the tuning phase, the system takes over and generates a set of reference inputs (e.g., a step) and observes and analyzes the step response to determine the values of the parameters of the premises of the above rules. Then the rule-base is applied to these, the new PID gains are found, and the closed-loop control system is put into operation with these new gains. Clearly, however, it may be unacceptable to expect the system to be taken off-line for such tests and retuning. On the other hand, for some applications (e.g., in process control) it is possible to follow such a retuning scenario.

Overall, we would emphasize that fuzzy PID auto-tuners tend to be very application dependent and it is difficult to present a general approach to on-line fuzzy PID auto-tuning that will work for a wide variety of applications. At this point, however, the reader who understands how a fuzzy system operates and has a specific application in mind, can probably quickly write down a set of rules that quantifies how to keep the PID controller properly tuned. This is often the primary advantage of the fuzzy systems approach to PID auto-tuning over conventional methods.

7.2.2 Fuzzy Gain Scheduling

Conventional gain scheduling involves using extra information from the plant, environment, or users to tune (via "schedules") the gains of a controller. The overall scheme is shown in Figure 7.3. There are many applications where gain scheduling is used. For instance, in aircraft control you often perform linear control designs about a whole range of operating points (e.g., for certain machs and altitudes). These control designs provide a set of gains for the controller at each operating condition over the entire flight envelope. A gain schedule is simply an interpolator

that takes as inputs the operating condition and provides values of the gains as its outputs. One way to construct this interpolator is to view the data associations between operating conditions and controller gains as forming a large table. Then, when we encounter an operating condition that is not in the table, we simply interpolate between the ones in the table to find an interpolated set of gains. In this way, as the aircraft moves about its flight envelope, the gain scheduler will keep updating the gains of the controller with the ones that we would have picked based on a linear design about an operating point. This general gain scheduling approach is widely used in the aircraft industry (e.g., also for engine control).

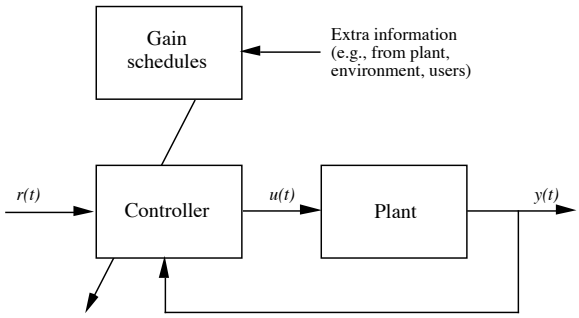


FIGURE 7.3 Conventional gain scheduler.

Notice that Figure 7.1 could be viewed as a gain scheduling approach to control where we come up with a fixed mapping between operating conditions, plant data, and external inputs (e.g., the user inputs) and the controller parameters. The fuzzy system is a static nonlinearity just like many conventional gain schedules, and it simply offers another type of interpolator for gain scheduling. So, what does the fuzzy system methodology offer to conventional gain scheduling approaches? First, as we discuss next, it offers three different methodologies for gain schedule construction.

Heuristic Gain Schedule Construction

It may be possible to use intuition to write down a rule-base that characterizes how the gains should be tuned based on certain operating conditions. In this case the design of a set of linear controllers for various operating conditions may not be necessary as it may be possible to simply specify, in a heuristic fashion, how the gains should change. The use of rules may be particularly useful here. For example, consider the tank level control application described in Section 6.6.4 on page 398. We could use a rule-base to schedule the gain of a linear proportional controller based on our knowledge that for higher liquid levels we will have bigger cross-sectional areas for the tank. In this case the rules may represent that we want to have a lower proportional gain for lower liquid levels since for lower levels the tank will fill faster with smaller amounts of liquid but for high levels the controller will

have to pump more liquid to achieve the same change in liquid level. We see that some practical knowledge of the tank's shape, which is not normally used in the design of a linear controller, can possibly be effectively used to schedule the gain of a linear controller for the tank. It is clear that the knowledge used to construct the gain schedule can be highly application-specific; hence, it is difficult to provide a general methodology for the heuristic construction of fuzzy gain schedulers beyond what has already been presented for techniques to design standard fuzzy controllers.

Identification for Gain Schedule Construction

In the case where we have a set of controller designs (based on linear or nonlinear theory) for each set of operating conditions that we can measure, we can use the methods for function approximation described in Chapter 5 to identify the function that the data inherently represents. This function that interpolates between the data is the gain schedule, and the methods of Chapter 5 can be used for its construction. To use the methods of Chapter 5, consider how we would schedule just one of the (possibly) many gains of the controller. To do this we view the operating condition–gain associations produced by the design of the controllers as data pairs

$$(x^i, y^i) \in G$$

where x^i is the i^{th} operating point and y^i is the gain that the controller design procedure produced at this operating point. The set G is the set of all such data pairs—that is, operating condition–controller associations. Now, the problem is formulated in exactly the same manner as the one for Chapter 5 and hence any of the methods there can apply. For instance, we can use the learning from examples methods, a least squares method, a gradient technique, or a clustering approach.

Parallel Distributed Compensation

Here we consider the case where the plant to be controlled is a nonlinear interpolation of R linear models as discussed in Chapter 2, Section 2.3.7 on page 73, and in Chapter 4, Section 4.3.5 on page 200. Recall that we had rules of a Takagi-Sugeno fuzzy system representing the plant that are of the form

$$\textbf{If } \tilde{x}_1 \text{ is } \tilde{A}_1^j \textbf{ and } \tilde{x}_2 \text{ is } \tilde{A}_2^k \textbf{ and}, \ldots, \textbf{and } \tilde{x}_n \text{ is } \tilde{A}_n^l \textbf{ Then } \dot{x}^i(t) = A_i x(t) + B_i u(t)$$

Similar rules could be created with consequents that use a discrete-time state-space representation or transfer functions, either discrete or continuous. Here we will simply use the continuous-time linear state-space models as consequents for convenience, but the same basic concepts hold for other forms.

No matter what form the rules are in, there are basically two ways that the rules can be constructed to represent the plant. First, for some nonlinear systems the mathematical differential equations describing the plant can be transformed

into

$$\dot{x}(t) = \left(\sum_{i=1}^{R} A_i \xi_i(x(t)) \right) x(t) + \left(\sum_{i=1}^{R} B_i \xi_i(x(t)) \right) u(t) \tag{7.1}$$

which is the model resulting from the above rules. A second approach would be to use system identification to construct the rules (e.g., use the least squares, gradient, or clustering with optimal output predefuzzification methods of Chapter 5). While it is possible to modify the identification methods so that they apply directly to continuous-time systems, it may be easier to simply use a discrete-time representation in the consequents and directly use the methods of Chapter 5.

Recall that in Chapter 4 our controller for the above plant used the rules

$$\textbf{If } \tilde{x}_1 \text{ is } \tilde{A}_1^j \textbf{ and } \tilde{x}_2 \text{ is } \tilde{A}_2^k \textbf{ and}, \ldots, \textbf{and } \tilde{x}_n \text{ is } \tilde{A}_n^l \textbf{ Then } u^i = K_i x(t)$$

where K_i, $i = 1, 2, \ldots, R$, are $1 \times n$ vectors of control gains, and the premises of the rules are identical to the premises of the plant rules. In this case

$$u(t) = \sum_{j=1}^{R} K_j \xi_j(x(t)) x(t) \tag{7.2}$$

Under certain conditions this controller can be used to stabilize the plant (see Section 4.3.5 on page 200).

Many view parallel distributed compensation as a form of gain scheduling since we have shown in Chapter 4 that we can achieve stability by designing the K_j, $j = 1, 2, \ldots, R$ then using Equation (7.2) to interpolate between these gains. Some view this interpolation as a "smooth switching" between the gains so that the controller gains for each rule compensate for their corresponding local linear model for the plant.

The above approach may be particularly appealing since under the conditions outlined in Chapter 4 the gain schedule construction method guarantees global asymptotic stability of the equilibrium point of the closed-loop system. However, it is important to note that the method can sometimes be effectively applied in a more heuristic manner when, for example, the plant cannot be transformed into Equation (7.1) so that we may not be able to guarantee stability. For instance, you may use the above approach to interpolate between different nonlinear controllers when a plant can be viewed as an interpolation of R nonlinear systems. Such an approach can be very valuable in practical applications.

Gain Schedule Implementation

Regardless of which of the above methodologies is used to construct the gain schedule, these approaches also automatically provide for the fuzzy system implementation of the gain scheduler (interpolator), which can at times be an advantage over conventional methods. That is, the fuzzy system approach provides methods to

construct gain schedules, and then as a by-product of the construction procedure, one obtains a method for implementing the fuzzy gain scheduler.

It is important to note that computational complexity is of concern in the implementation of a gain scheduler. Sometimes it is not computation time that is the main problem (since the interpolator is usually relatively simple), but memory storage since the tables of gains can be quite large for practical applications. Some may think that the fuzzy system methodologies could offer an approach to reduce the memory storage requirements since the rules will tend to provide an interpolation that may make it possible to omit some data. However, we have found that for practical applications this is sometimes simply wishful thinking. The conventional methods used for gain scheduling use interpolation techniques that are basically quite similar (sometimes identical) to what is implemented by a fuzzy system. If the original designers found that the extra data were needed to make the mapping that the gain scheduler implements more accurate, then these extra data are probably needed for the fuzzy system also.

7.2.3 Fuzzy Supervision of Conventional Controllers

In the two previous subsections we have studied how to use a fuzzy system (supervisor) to tune the gains of a PID controller and schedule controller gains. Each of these subsections offers a method for fuzzy supervision of conventional controllers. In general, however, we view the supervisory control approach as offering more general functionalities than the simple tuning of gains: They also provide the capability to completely switch which controllers are operating at the lower level. That is, they can switch between linear and nonlinear controllers, controllers of different order or structure, or between controllers that regulate and actuate different signals. Next, we briefly consider an application where we have found it necessary to both tune the gains of the lower-level controllers and switch between different controllers.

In particular, it has been shown [57] that for a two-link flexible robot (the one studied in Chapters 3 and 6 and later in this chapter) we can design very good conventional controllers for certain operating conditions and slews of the robot. For instance, after years of experience with this robot, we know that for small slews we need a certain type of controller, whereas for large slews and counterrelative slews we need a different kind of controller. Past work showed, however, that none of the controllers developed for one set of slew conditions would work perfectly for other sets of conditions. However, in [57] it was shown that since we know what slew is commanded by the user, this information can be incorporated into a supervisor that simply tunes the lower-level controllers or selects the proper controller for the slew that is commanded. This supervisory control approach proved to be successful. However, it could not accommodate for *unknown* effects such as adding a payload to the endpoint of the second link as we did in Chapter 6. The reason that this sort of condition could not be accommodated for in this supervisory method is that it cannot be directly measured and used to tune the controllers. We emphasize, however, that if you were to add some preprocessing (e.g., on-line frequency domain analysis), it is possible to guess what payload has been added by correlating the

movement of the modes of vibration with the known effects of mass loading. We did not, however, pursue this as the adaptive methods in Chapter 6 and the rule-base supervision approach presented in the next section proved to be quite effective.

7.3 Supervision of Fuzzy Controllers

The basic architecture for the operation of a supervisor for fuzzy controllers is the same as the one shown in Figure 7.1 with the controller at the lower level being a fuzzy controller or adaptive fuzzy controller that is tuned by the supervisor. The supervisor could be in the same form as a basic fuzzy controller or it could be a general rule-based expert system. We have not yet found, however, a clear need for the general knowledge-base and inference strategies found in conventional expert systems. It is for this reason that in this section we will consider the use of fuzzy systems or simple sets of rules for the tuning of fuzzy controllers (note that simple sets of crisp rules can actually be represented by fuzzy systems with an appropriate choice of membership functions and inference strategy; in this sense we really only consider fuzzy supervision of fuzzy controllers).

This section is broken into two parts, each of which has an associated design case study. First, we discuss the supervision of a direct fuzzy controller by supervising its rule-base. Next, we discuss the supervision of an adaptive fuzzy controller by supervision of its adaptation mechanism.

7.3.1 Rule-Base Supervision

When you consider the possibility of tuning direct fuzzy controllers, you may be struck by the possibility of automatically adjusting each and every component of the fuzzy controller. This is, however, not usually a good idea. Since there are very few heuristic ideas on why it is best to choose a certain fuzzification or defuzzification method, it is very difficult to know when or how to tune these on-line. Moreover, it is difficult to gather on-line information that suggests how these components of a fuzzy system should be tuned. It is possible to develop tuning methods for the parameters of the inference mechanism, but some parameterization of the inference strategy is needed and a rationale for tuning must be specified.

It is for these reasons that most often the focus is on tuning the rule-base (both membership functions and rules), which we often have good heuristic ideas on how to tune, and on-line signals that tell us what tuning needs to be done. For instance, when we design a direct fuzzy controller we may initially have had a certain set of operating conditions in mind. We may also know that if different operating conditions are encountered, the controller's performance will degrade and a different controller should be used. The higher-level knowledge of when different controllers are needed for different operating conditions can be loaded into the rule-based supervisor.

The supervisory tuning of rule-bases is, however, very application-specific; this is not surprising as the very design of the rule-base for the direct fuzzy controller that we seek to tune is application-specific. It is for this reason that to further

explain how rule-bases can be supervised, we provide a design and implementation case study for a rule-base supervisor.

7.3.2 Case Study: Vibration Damping for a Flexible Robot

The experiments using direct and adaptive fuzzy control for the two-link flexible robot described in Chapters 3 and 6 show a considerable improvement over the no-control case shown in Figure 3.2 on page 127, but are not quite the best possible. The uncoupled direct fuzzy controller described in Chapter 3 has a fast rise-time but has the drawback of having a large overshoot and oscillations near its set-point. Coupling the two controllers via the endpoint acceleration signal reduces the overshoot and oscillation problems considerably but makes the overall response a bit slower due to the reduction of speed of the elbow link while the shoulder link is moving fast. This reduction of speed of the elbow link was necessary to prevent the oscillations of the elbow link endpoint near the set-point, caused by the inertia of the links.

We can overcome this problem if we are able to make a smooth transition in the speed of the shoulder link. This can be achieved by using a higher level of control for monitoring and adjusting the direct fuzzy controller. Here, we will use a simple "expert controller," shown in Figure 7.4, which monitors the position error input $e_1(t)$ to the shoulder motor controller and changes the fuzzy sets and the rule-base of the shoulder motor controller. As we saw in the response for the coupled direct fuzzy controller (see Figure 3.9 on page 140), the main cause for the hump appearing at about 0.9 seconds in the plot is the sudden change in speed of the shoulder link. The elbow link is coupled to the shoulder link using the endpoint acceleration from the shoulder link, and its speed is varied depending on the oscillation of the shoulder link. To eliminate the hump, we use the expert controller to vary the speed of the shoulder link gradually so as to avoid exciting oscillatory modes, which result in excessive endpoint vibrations.

The rule-base for the expert controller consists of two single-input multiple-output rules:

1. **If** $|e_1(t)| \geq 20^\circ$ **Then** use Rule-Base 1 (Table 7.1) **and** use expanded universes of discourse.

2. **If** $|e_1(t)| < 20^\circ$ **Then** use Rule-Base 2 (Table 7.2) **and** use compressed universes of discourse.

The expert controller expands or compresses the universes of discourse by simply changing the scaling gains (explained below). When the universe of discourse is expanded, a "coarse control" is achieved, and when it is compressed, a "fine control" is achieved. You may think of this as being similar to the dynamically focused learning that we studied in Chapter 6; it is a type of auto-tuning strategy. The form of the premises of the rules of the supervisor guarantees that one (and only one) of the rules will be enabled and used at each time step. Since the control objectives can be achieved using only these two rules and since only one rule will be enabled

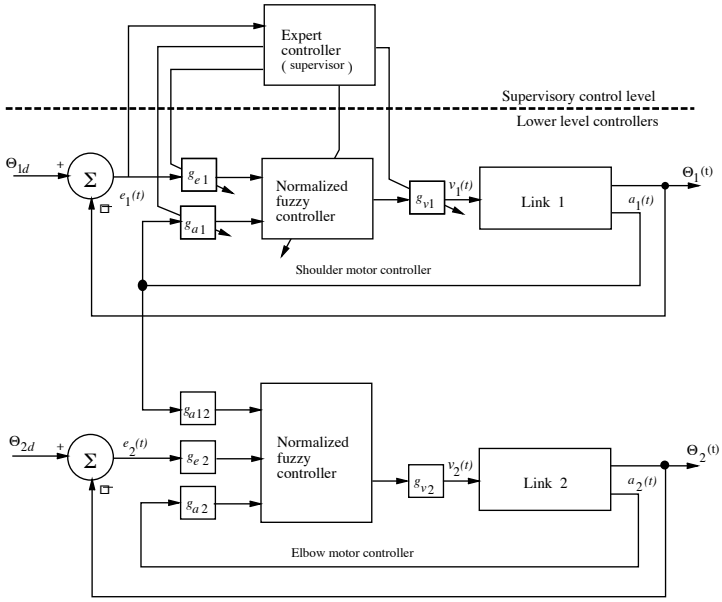


FIGURE 7.4 Fuzzy control system with a supervisory level (figure taken from [145], © IEEE).

at each time step, there is no need for the use of complex inference strategies in the expert controller.

Clearly, it would be possible to view the premises of the above rules as linguistic statements then define membership functions to quantify their meaning. We could also use membership functions to quantify the meaning of the consequents (how?). Then, our supervisor would be a fuzzy system that would gradually rather than abruptly switch between the two conditions. Here, we did not pursue this approach. Instead, we made sure that the rule-bases were designed so that a smooth transition would occur as the supervisor switched between the rules.

Rule-Base Construction

The membership functions and the rule-base for the elbow link controller were kept the same as in Figure 3.8 on page 136. In addition to the rule-base in Table 7.1, another rule-base was added for the shoulder motor controller. The expert controller therefore switches between these two rule-bases for the shoulder link, for "coarse control" and "fine control." The membership functions for the coarse controller are similar to those used in the coupled direct fuzzy controller case (see Figure 3.4 on page 131) where the universe of discourse is $[-250, +250]$ degrees for the position error, $[-2, +2]$ g for the endpoint acceleration, and $[-1.5, +1.5]$ volts for the output voltage. The fine controller, with rule-base shown in Table 7.2, uses the same shape for the membership functions as shown in Figure 3.4 except the universes of dis-

course for the inputs and the outputs are compressed. The universe of discourse for the position error is $[-25, +25]$ degrees, and the universe of discourse for the endpoint acceleration is $[-1, +1]$ g. The output universe of discourse is $[-0.15, +0.15]$ volts.

TABLE 7.1 Rule-Base for Coarse Control

V_1^m		A_1^k										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
E_1^j	-5	-5	-5	-5	-4	-4	-3	-3	-2	-2	-1	0
	-4	-5	-5	-4	-4	-3	-3	-2	-2	-1	0	1
	-3	-5	-4	-4	-3	-3	-2	-2	-1	0	1	2
	-2	-4	-4	-3	-3	-2	-2	-1	0	1	2	2
	-1	-4	-3	-3	-2	-2	-1	0	1	2	2	3
	0	-3	-3	-2	-2	-1	0	1	2	2	3	3
	1	-3	-2	-2	-1	0	1	2	2	3	3	4
	2	-2	-2	-1	0	1	2	2	3	3	4	4
	3	-2	-1	0	1	2	2	3	3	4	4	5
	4	-1	0	1	2	2	3	3	4	4	5	5
	5	0	1	2	2	3	3	4	4	5	5	5

Notice that while going from the coarse to the fine control, the widths of the universes of discourse for the position error and the output of the shoulder link controller have been reduced by a factor of ten, while the width of the universe of discourse for the endpoint acceleration is reduced by a factor of two. This choice was made after several experiments where it was found that when the width of the universe of discourse for the acceleration was reduced by a large factor, the controller became too sensitive near the set-point.

Tables 7.1 and 7.2 show the rule-bases used for the coarse and fine control, respectively. Notice that in row $j = 0$ for the rule-base for fine control there are extra zeros as before to reduce the sensitivity of the controller to a noisy acceleration signal. The rule-base for coarse control does not have these zeros as the offset voltage from the accelerometers is of no consequence as long as the controller is operating in this region. Also notice that while the patterns in the bodies of the tables shown in Tables 7.1 and 7.2 are similar, there are differences included to reflect the best way to control the robot. Notice that the center values in the fine control rule-base change more rapidly as we move away from the center of the rule-base as compared to the coarse control rule-base. This causes a bigger change in the output of the controller for smaller changes in the input, resulting in better control over the shaft speed of the motor, preventing it from overshooting its set-point and at the same time causing gradual braking of the motor speed. Finally, we note that the fine control rule-base was selected so that the output is not changed too much when the rule-bases are switched, promoting a smooth transition between the rule-bases.

The number of rules used by the supervisory control algorithm is 121 for the coarse controller, plus 121 for the fine controller, plus 343 for the elbow controller,

TABLE 7.2 Rule-Base for Fine Control

V_1^m		A_1^k										
		−5	−4	−3	−2	−1	0	1	2	3	4	5
E_1^j	−5	−5	−5	−5	−5	−4	−4	−3	−3	−2	−1	0
	−4	−5	−5	−5	−4	−4	−3	−3	−2	−1	0	1
	−3	−5	−5	−4	−4	−3	−3	−2	−1	0	1	2
	−2	−5	−4	−4	−3	−3	−2	−1	0	1	2	3
	−1	−4	−4	−3	−3	−2	−1	0	1	2	3	3
	0	−4	−3	−2	−1	0	0	0	1	2	3	4
	1	−3	−3	−2	−1	0	1	2	3	3	4	4
	2	−3	−2	−1	0	1	2	3	3	4	4	4
	3	−2	−1	0	1	2	3	3	4	4	5	5
	4	−1	0	1	2	3	3	4	4	5	5	5
	5	0	1	2	3	3	4	4	5	5	5	5

plus 2 for the expert controller, resulting in a total of 587 rules. As either the coarse controller or the fine controller is active at any time, effectively the number of rules used is $587 - 121 = 466$ rules (which is similar to what was used for the coupled direct fuzzy controller in Chapter 3).

Experimental Results

Experimental results obtained using this supervisory scheme are shown in Figure 7.5. The requested slew is 90° for both links as shown in the inset. The response is relatively fast with very little overshoot. Comparing this response to the response obtained for the coupled direct fuzzy controller (see Figure 3.9 on page 140), we can see that the response from the supervisory controller has a much smaller settling time and that the hump in the initial portion of the graph is almost eliminated. Note that this is similar to what we obtained for the FMRLC (see Figure 6.17 on page 362).

The response of the robot to a counterrelative slew is better than the response when using the direct fuzzy controllers (coupled or uncoupled). The responses for small-angle slews are similar to those obtained for the direct fuzzy controller and the FMRLC. Figure 7.6 shows the response of the robot with a 30-gram payload on the endpoint. The commanded slew is 90° for both the links, as shown in the inset. The response is significantly improved as compared to the response from the direct fuzzy control schemes (see Figures 3.6 and 3.10 on pages 134 and 141) and slightly better than that of the FMRLC (see Figure 6.18 on page 363). The oscillations in the endpoint due to the added inertia, visible in the direct fuzzy control case (see Figures 3.6 and 3.10), are eliminated here.

From the results obtained for the direct fuzzy control techniques (see Chapter 3), the FMRLC (see Chapter 6), and the supervisory control technique (here), we see that the results from the latter are superior in all the cases tested. The supervisor gave better results in the case of large, counterrelative, and loaded-tip slews, and was comparable to the results obtained from the direct fuzzy controller

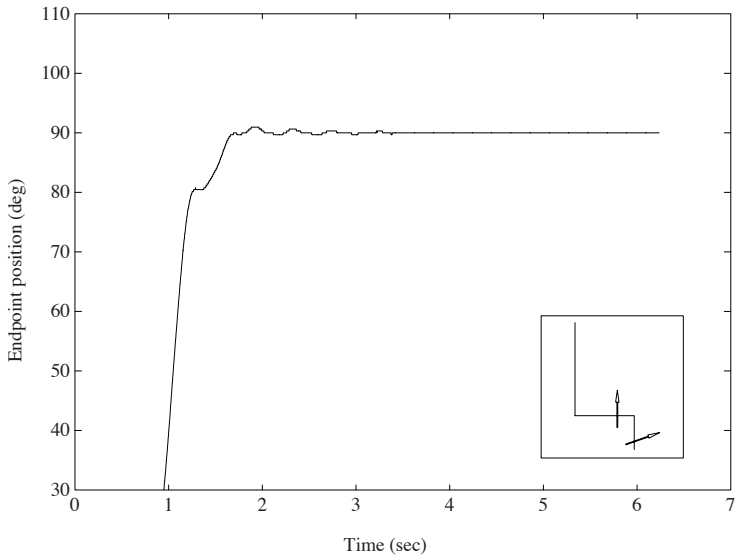


FIGURE 7.5 Endpoint position for supervisory control (figures taken from [145], © IEEE).

in the case of small slews. Compared to the FMRLC, the supervisor did slightly better on the counterrelative slew and the case where there is a payload; not only that but the FMRLC needed 1150 rules compared to the 466 needed for the supervisory approach. The major difference in the implementation of the supervisory controller and the direct fuzzy controllers is the addition of the expert controller, and the additional rule-base for the shoulder link controller. This addition does increase the complexity of the control algorithm but not to a large extent (recall that we used 464 rules for the coupled controller and only 466 for the supervisory controller with an extra 121 for the second shoulder controller). The loop execution time increased very little, and the same sampling time (15 milliseconds) as in the direct fuzzy control case was used. The supervisory control does use extra memory as compared to the direct fuzzy algorithms since the second rule-base (with 121 rules) for the shoulder controller had to be stored; however, in implementation, the supervisor simply has to select one rule-base or the other.

Overall, we see that rule-base supervision can be used to significant benefit in a practical application. Next, we study how to supervise an adaptive fuzzy controller.

7.3.3 Supervised Fuzzy Learning Control

A supervisor for an adaptive fuzzy controller could serve to supervise the adaptation mechanism and fuzzy controller in the case of direct adaptive fuzzy control, or the

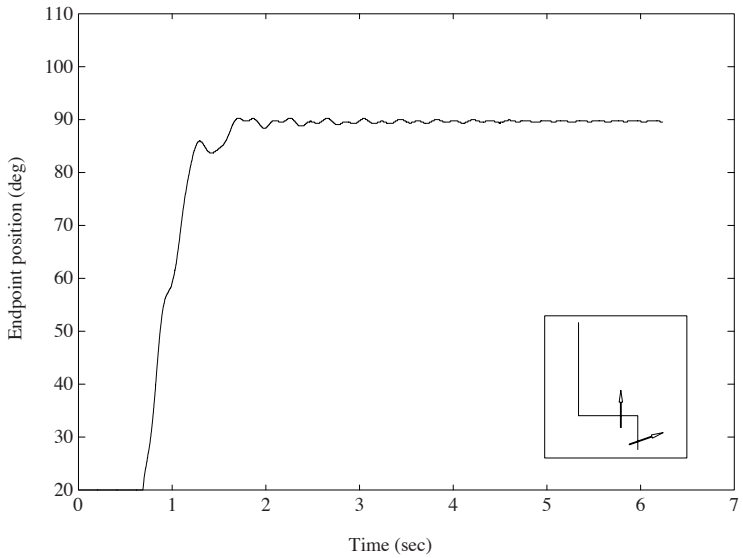


FIGURE 7.6 Endpoint position for supervisory controller design with payload (figure taken from [145], © IEEE).

identifier and controller designer in the case of indirect adaptive control. If a model reference approach is used then there may be a need to supervise the reference model. Basically, supervision allows for the incorporation of additional information on how the adaptation process should proceed. In the indirect case, certain extra information may be useful in helping tune the model of the plant. Or it may be the case that certain information can be used by the controller designer. The supervision of the adaptation mechanism is useful to speed or slow the adaptation rate or to try to maintain stable operation. A supervisor for a reference model will update it so that it continually represents the desired specifications.

The number of possibilities for supervising adaptive fuzzy controllers is quite extensive. At present there are no general guidelines on which approach to choose, so the designer is often left to use intuition to make a best guess at an approach to take (especially for complex applications). As with the previous supervisory control approaches, the supervision method for adaptive fuzzy controllers is highly dependent on the application at hand.

To illustrate the ideas, we will provide a case study where we supervise the learning mechanism and reference model of the FMRLC for a fault-tolerant control application.

7.3.4 Case Study: Fault-Tolerant Aircraft Control

The FMRLC designed for the fault-tolerant aircraft control case study in Chapter 6 gives promising results; however, the learning process takes considerable time to completely reconfigure the control laws due to the actuator failure (approximately 9 seconds, as shown in Figure 6.15 on page 357). This time requirement arises from the fact that we are forced to use a "slower" learning mechanism to ensure stable operation for a wide variety of failures, and the FMRLC does not use special information about the actuator failure (e.g., information about when the failure occurs and where the actuator is stuck). Basically, no matter what kind of failure occurs, the FMRLC of Chapter 6 will try to redistribute the remaining control authority to the healthy channels so that the aircraft will behave, if not exactly the same, as close as possible to its unimpaired condition. It is, however, somewhat unreasonable to expect the FMRLC to achieve performance levels comparable to the unimpaired case if, for example, the aileron becomes stuck at near full deflection. For such failures, we would find it acceptable to achieve somewhat lower performance levels.

In this section, we will show that if failure detection and identification (FDI) information is exploited, a supervisory level can be added to the FMRLC to tune the reference model characterization of the desired performance according to the type of failure that occurred. We show that if the supervisory level uses FDI information to tune the reference model, and if it appropriately adjusts the learning mechanism for different failures, then adequate performance levels can be achieved, even for more drastic failures. We begin by investigating two such supervised FMRLC techniques, one that uses limited FDI information, and the other that uses perfect FDI knowledge. Moreover, utilizing the approach and results in [108], we explain how a fuzzy system can be used for failure estimation and can be coupled with the fuzzy supervisor so that we do not need to assume that we have perfect knowledge about failures.

FMRLC Supervision with Limited FDI Information

We will first consider the case where only limited failure information is available. In particular, we assume that we have an FDI system that can only indicate whether or not an actuator has failed; hence, while it indicates when an actuator failure occurs, it does not indicate the severity of the failure (we also assume it provides no false alarms). A supervised FMRLC that uses such failure information is shown in Figure 7.7. The FMRLC supervisor consists of an FDI system, which gives binary information about an actuator failure, and a control supervisor, which will tune the reference model and learning mechanism when an actuator failure is detected.

Because only limited knowledge about the nature of the actuator failure is provided by our FDI system, it is not possible to design very complex supervision strategies. In general, in this case there are two approaches to supervising the FMRLC. One possible supervision strategy is to increase the learning capabilities of the FMRLC (i.e., increase the output gain m_0 of the fuzzy inverse model) so that the failure recovery can be accelerated. However, in order to obtain an adequate

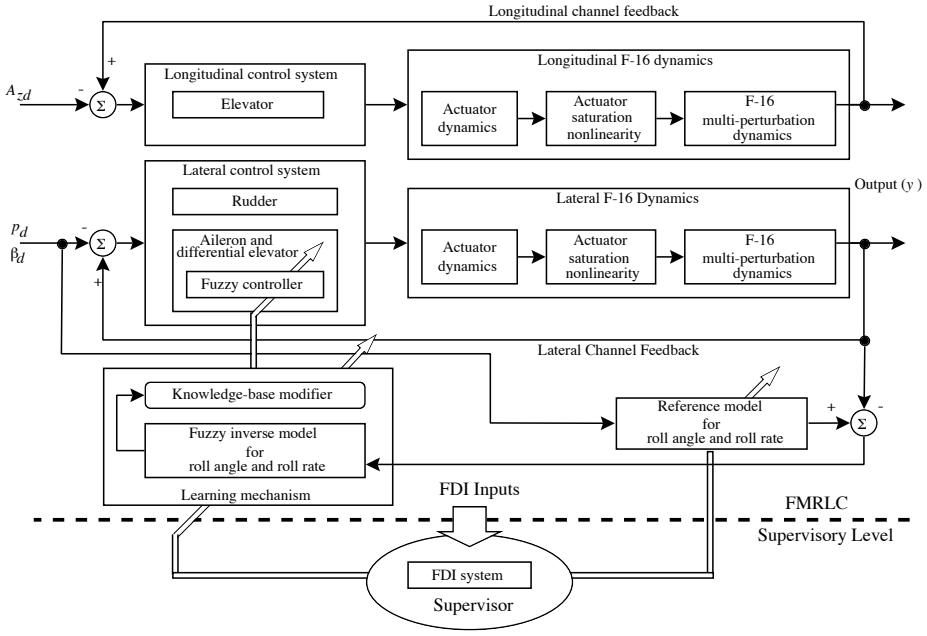


FIGURE 7.7 Supervised FMRLC using FDI information (figure taken from [104], © IEEE).

increased output gain for the fuzzy inverse model, the reference model needs to be "slowed down" to help ensure stability. Therefore, the reference models must be designed much more carefully to represent achievable performance levels of the impaired aircraft. Intuitively, the pilot will probably not push the performance of an impaired aircraft to the same level as its unimpaired condition. In the case of an aileron actuator malfunction, the pilot loses the primary roll control effector. Even though the differential elevator can be used to reconfigure for this failure, the pilot will never expect a full recovery of the original performance, realizing that the differential elevators alone do not have enough control authority to overcome the failure and replace all the functions originally designed for the ailerons. In all cases, when an aircraft loses a control surface, it loses some performance. Hence, there is a general decrease in performance expectations from the pilot, which can be mimicked by the change in the functionality provided by the reference model in the FMRLC design.

Hence, as soon as a failure is detected, a different set of reference models should be used. The second- and third-order reference models chosen for the FMRLC have two parameters available for modification; they are ζ and ω_n (which are 0.85 and 14.14, respectively, in Chapter 6, Section 6.3.2). In the case of actuator failures, a pilot would expect the aircraft to react in a much slower fashion with overdamped responses. Hence, this expectation is translated to a larger ζ (i.e., $\zeta > 0.85$) and

a smaller ω_n (i.e., $\omega_n < 14.4$). We find that the values $\zeta = 1.0$ and $\omega_n = 10.0$ are reasonable choices—that is, they do not jeopardize performance too much.

With this set of slower reference models, the fuzzy inverse model is re-tuned (using the approach introduced in the Chapter 6 case study), and it is found that the output gain m_0 is 0.6. Here, by reducing the performance requested by the reference model, the learning capabilities (i.e., its adaptation rate) are increased by a factor of six. This greater learning capacity will then speed up the control reconfiguration process.

As an alternative approach, the fuzzy controller can be modified directly in the direction of how the learning mechanism will do the control reconfiguration. Referring to the F-16 nominal controller of the lateral channel as shown in Figure 6.11 on page 349, the effectiveness of the differential elevators as the roll effector is actually depressed by a factor of four compared to the ailerons (see the 0.25 factor between the δ_a and δ_{de} controller outputs). Thus, in the control reconfiguration process, the learning mechanism of the standard FMRLC needs to bring the effectiveness of the differential elevators up to the level of the ailerons, and then further fine-tune the control laws (i.e., the rule-base of the fuzzy controller) to compensate for the actuator failure. With this process in mind, another approach to accelerate the learning process is to assist the learning mechanism in increasing the effectiveness of the differential elevators. This approach can be achieved simply by increasing the output gain of the fuzzy controller for the differential elevators by a factor of, for example, four as soon as the stuck aileron is detected. Using this direct controller modification, instead of using a slower learning mechanism to achieve the same goal, the control reconfiguration process will definitely be accelerated.

Notice that in the first approach, the learning capabilities are increased by reducing the requirements posed by the reference model, whereas the second method allows a direct change in the configuration of the controller itself. The results of applying these two supervision approaches are actually similar, and in both cases it is necessary to incorporate them in a fashion that helps ensure stability. After some simulation studies, it was found that it is best to use a combination of the two methods. The output gain $m_0 = 0.3$ is chosen with the slower reference model given above in the first approach, and the controller output gain of the differential elevators is increased by a factor of two to 0.5. This choice reflects a moderate contribution from each approach, but detailed simulation studies show that virtually any combination of the two approaches will also work with relatively minor differences. Hence, as soon as the FDI system detects the aileron failure, the supervisor will switch the reference models, increase the output gain of the fuzzy inverse model, and alter the fuzzy controller as described above.

FMRLC Supervision Results

Figure 7.8 shows the responses of the F-16 using the supervised FMRLC. By comparing these to the responses in which no FDI information is used (see the dotted line in Figure 7.8), the results show improvements in that there are fewer oscillations in the first 9 seconds (see, e.g., the arrows in the roll rate and the roll angle

plots) compared with the unsupervised case. The supervised FMRLC ensures that the system follows the "slower" reference models in case of failure, which prevents the controller from pushing the aircraft beyond its capabilities. By allowing the FMRLC to learn the failure situation more rapidly, the actuator response of the differential elevators is more active than that of the response in the FMRLC without the supervisor (see the arrows in the differential elevator plots in Figure 7.8). However, the choice of a set of slower reference models results in a larger steady state error in the roll angle after and during the maneuver (see the arrows in the roll angle plot in Figure 7.8). This is due to the fact that the slower roll rate model causes the final value of the roll angle to shift. This phenomenon is a characteristic of the new set of reference models. From the simulation results, this study shows that a "slower" reference model for FMRLC will often give a less oscillatory overall response; clearly, there exists a trade-off between performance and stability for an impaired aircraft.

We have also shown that if there is a delay in the FDI system providing its failure information, there is little degradation in performance. Actually, a 0.5-second delay in obtaining FDI information does not further degrade the response of the aircraft; rather, the time delay only briefly suspends the enhancement made by the supervised FMRLC.

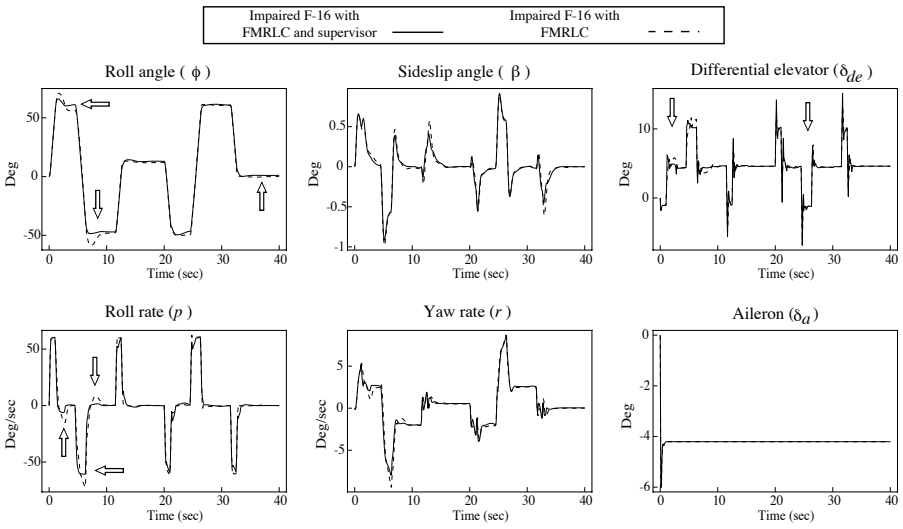


FIGURE 7.8 Impaired F-16 with supervised FMRLC (aileron stuck at 1 second) using limited FDI information (figure taken from [104], © IEEE).

FMRLC Supervision: Failure Estimation and Other Scenarios

In more advanced FDI systems, estimation of the actual failure condition is possible. For instance, the actuator position can be estimated from the input-output measurements of the aircraft, which allows the supervisor to accelerate the control reconfiguration. Actually, if either exact FDI information or estimated failure information is provided by a fuzzy estimator, then we can further improve the performance of the fault-tolerant control strategy (even if there is some delay in obtaining the FDI information). Basically, the supervised FMRLC with perfect FDI information will accelerate the control reconfiguration so that there is no noticeable oscillation in the learning process and will outperform the supervised FMRLC that uses only limited FDI information.

The supervised FMRLC with perfect information gives excellent results compared to the standard FMRLC of Chapter 6, Section 6.3.2, and when fuzzy failure estimation is used, we get essentially the same results. However, there is at least one more problem that must be addressed. In the case where the failed aileron sticks at near maximum deflection, the differential elevator is saturated by the control reconfiguration process since its roll capabilities are less than that of the ailerons. This causes an instability due to the fact that when the healthy channels are saturated under the standard FMRLC, the learning mechanism modifies the control laws in the wrong direction with even more severe saturation. As an illustration, assume that the fuzzy controller gives a control output that saturates the actuator. If the plant responses are significantly slower than that of the reference models, this results in larger inputs to the fuzzy inverse model (i.e., the larger errors between the plant and the reference models). Hence, the monotonic relationship given by the fuzzy inverse model will try to update the fuzzy controller to increase its output in the direction of further saturating the actuator. Finally, the system will go unstable because the learning mechanism is not working in its designed fashion to "memorize" the past successful actions. Note that if there are extra redundant control surfaces for control reconfiguration, this situation may not occur.

In order to accommodate for the actuator failures at high deflection angle, which will cause the healthy channel to saturate, an extra (crisp) rule is used in the supervisor to correct the deficiency in the standard FMRLC:

If any actuator is saturated **Then** do not update the corresponding fuzzy controller.

With this rule, the controller update is ceased by the saturation of any actuator in the healthy channels. This action assumes that the controller output at which the saturation occurred is the best control signal the plant can handle, and further modification should not be made.

Since we have studied only aileron failures to this point, we now investigate a bit more severe failure in the differential elevator. To simulate the failure in the differential elevator, at the time of failure instant $t = 1$ second, we will force the differential elevator to move at its highest rate until 85% of maximum deflection is reached (for the F-16 simulation, a failure at 100% maximum deflection in either the

aileron or the differential elevator results in severe performance degradation, to the extent that the loaded roll sequence cannot be followed in the manner specified by the reference model, at least using the present control methods). The rule described above is used in the supervisor for the standard FMRLC (i.e., no other previous supervision strategies are used), and the results are compared with the impaired and unimpaired F-16 with the nominal controller, as shown in Figure 7.9. The rule can also be applied to the FMRLC with the FDI-based supervisors, but the results are similar to those in Figure 7.9 because any further improvements in performance are hindered by actuator saturation. We note that the responses with the standard FMRLC are extremely oscillatory (as it cannot recover the performance requested by the reference model), and are therefore not shown in Figure 7.9.

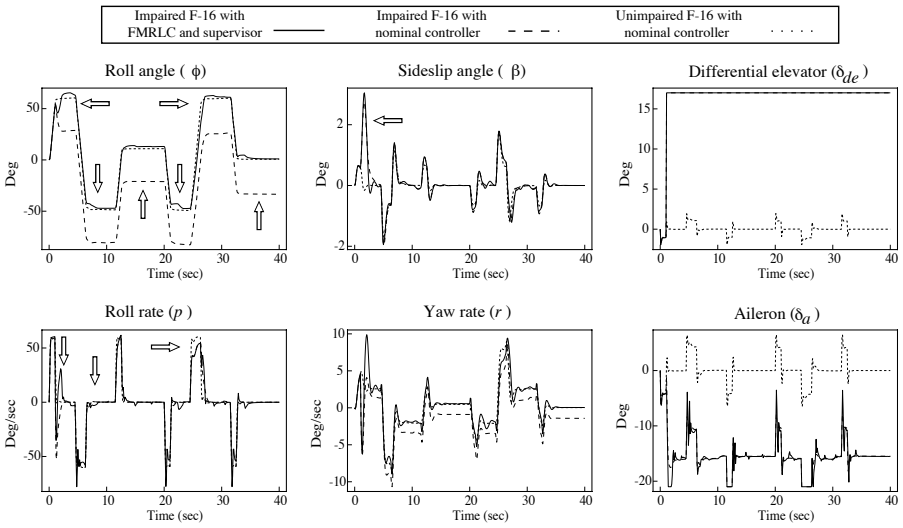


FIGURE 7.9 Impaired F-16 with supervised FMRLC (differential elevator goes to 85% maximum deflection at 1 second) with no FDI information (figure taken from [104], © IEEE).

As shown in the aileron actuator plot in Figure 7.9, the actuator output is saturated during certain parts of the loaded roll sequence. The resulting roll rate plot shows that the plant output is degraded in the positive side of the motion (as indicated by the arrows where the output is slower than the reference responses). Moreover, the overall roll rate responses are oscillatory, and the sideslip peaked at 2.6° at 1.8 seconds. The performance degradation is due to the momentary suspension of learning by the supervisor rule given above, which hinders the normal reconfiguration process. However, the roll angle response is reasonably well behaved.

Discussion: Significant Further Investigations Needed

While in some respects the supervised FMRLC approach appears promising, there is a significant need to investigate the following:

- The performance of the supervised FMRLC for a more complete nonlinear simulation model, and for a wider range of failures over more aircraft operating conditions.
- Stability and convergence issues.
- Robustness issues for the developed techniques, including determining if acceptable plant parameter variations can cause the failure estimation techniques to produce false alarms, and determining if the reconfiguration strategies can accommodate for other types of failures (including simultaneous and intermittent failures).
- Comparative analyses between some of the conventional approaches to reconfigurable control and FDI, and the fuzzy reconfigurable control and fuzzy FDI approaches investigated in this case study, in order to determine the achievable performance levels and computational complexity.

Moreover, issues related to proper interface between the pilot and the reconfiguration strategy are critical to the development of a reconfigurable control system.

The reader is cautioned to view the above application as being illustrative of an approach to supervising an adaptive fuzzy controller. It is not to be interpreted as a solution to the general problem of fault-tolerant aircraft control. We must emphasize that the solution studied in this case study falls significantly short of providing a practical solution to this very important problem (but many other "solutions" have the same basic problems). Significant research needs to be done, including flight tests, before you could come to trust a control reconfiguration strategy.

7.4 Summary

In this chapter we have provided an overview of the variety of methods that are available for fuzzy supervisory control. Basically, the techniques should be viewed as providing design methodologies for hierarchical controllers for nonlinear systems. We have found the methods to be very practical and useful for industrial applications.

The chapter was basically divided into two parts. In the first part, we showed how to supervise conventional controllers, including a PID controller, and explained how fuzzy systems are useful for gain scheduling. In the second part, we showed how to supervise both direct fuzzy controllers (i.e., rule-base supervision) and an adaptive fuzzy controller. We used the two-link flexible robot implementation case study to illustrate one way to perform rule-base supervision, and we used the fault-tolerant aircraft control problem to illustrate how to supervise an adaptive fuzzy controller.

Upon completing this chapter, the reader should understand the following:

- The basic approach to using fuzzy systems to tune PID controllers.
- How to use heuristic information to construct fuzzy gain schedulers.
- How to use the identification methods of Chapter 5 to construct fuzzy gain schedules when you are given as set of controllers designed at a set of measurable operating conditions.
- How the "parallel distributed compensator" can be viewed as a gain scheduling methodology.
- Basic ideas in the tuning and supervision of conventional controllers.
- Rule-base supervision via tuning of membership functions and switching between rule-bases.
- How rule-base supervision can be used for control of a two-link flexible robot.
- How both direct and indirect adaptive fuzzy controllers can be supervised (e.g., via supervision of the reference model or learning mechanism).
- How supervised adaptive fuzzy control provides an approach to "performance adaptive" control (in the sense that it seeks an appropriate performance level based on the failure conditions) and fault-tolerant control for an aircraft.
- The limitations of the methods for fault-tolerant aircraft control and the need for future research.

Essentially, this is a checklist of the major topics of this chapter. We must emphasize that this chapter is different from the past ones in that realistic applications of these approaches are difficult to define in a short amount of space. Hence, while it is easy to explain the ideas at a conceptual level, it is difficult to provide general design guidelines for supervisory control approaches. It is our hope, however, that with the foundation built by the first six chapters of this book, the concepts discussed here can be readily used for your own applications.

7.5 For Further Study

A variety of conventional PID auto-tuning methods are presented in [9] (a summary of several of the ideas in this book is contained in [121]). In addition to [57], the authors in [221, 152, 42, 216, 147] study fuzzy supervisory controllers that tune conventional controllers, especially ones that tune PID controllers. Conventional gain scheduling has been studied extensively in the literature, especially for a wide range of practical applications. See [182, 176, 183] for some more theoretical studies of gain scheduling. The connections between fuzzy supervision and gain scheduling have been highlighted by several researchers. A more detailed mathematical study of

the connections is provided in [154]. The idea of using a supervisor for conventional adaptive controllers was studied earlier in [8, 6]. The case study for supervisory control of the two-link flexible robot was first presented in [145]. The approach to supervision there bears some similarity to the one in [125]. The case study for the fault-tolerant aircraft control problem was taken from [104]. General issues in hierarchical fuzzy control are discussed in [43].

7.6 Design Problems

Design Problem 7.1 (Fuzzy Gain Scheduling for Liquid Level Control)*: In this problem you will study how to heuristically construct a fuzzy gain scheduler for the tank problem described in in Section 6.6.4. This problem is only partly specified. It is up to you to fully define the assumptions and approach.

(a) Design a conventional controller for the tank and use simulations to show how it performs in regulating the liquid level.

(b) Design a fuzzy scheduler for the controller you designed in (a) based on your knowledge that the tank has a larger cross-section at the top of the tank (but do not assume that you know $A(h)$ perfectly). Use a simulation to show how the gain scheduler performs. Compare it to the results in (a). If you design the problem correctly, you should be able to show that by using the additional information on the cross-sectional area of the tank you can achieve better performance than in the case where this information is not used in (a).

Design Problem 7.2 (Gain Scheduling and PID Tuning for the Cargo Ship)*: Suppose that you consider the speed u of the cargo ship described in Chapter 6 to dictate the ship's operating conditions and allow it to be an input to a gain scheduler. Note that if the ship is traveling very fast, a small change in the rudder angle will result in a large change in the heading. On the other hand, if the ship is traveling very slowly, then the rudder loses its control authority and the ship is harder to control. In this problem you will study the use of a gain scheduler for the control of the cargo ship. This problem is only partly specified. It is up to you to fully define the assumptions and approach. Overall, however, you should recognize that if a gain scheduler can properly tune the parameters of a controller, it should be possible to make the performance of the ship (in terms of heading response) less sensitive to changes in the ship's speed.

(a) Develop a PD controller for the cargo ship. Illustrate its performance in simulation. For a guideline for the type of performance that you should be able to achieve, use the plots shown in the cargo ship case study in Chapter 6 and the reference models provided there.

(b) Design a fuzzy gain scheduler that tunes the PD gains based on the speed as which the ship operates. Use a heuristic approach to specify the rules for a fuzzy gain scheduler that uses the above knowledge.

(c) Develop several PD controllers for a variety of speeds, and then use the methods of Chapter 5 to construct a gain schedule from the operating point–controller design data pairs. Illustrate how this and the methods in (b) can be used to improve performance over the case where no speed input is used (i.e., compare to (a)).

(d) Use the parallel distributed compensator approach to construct a gain scheduled control law for the plant. Illustrate its performance in simulation.

Design Problem 7.3 (Gain Scheduling and PID Tuning for the Tanker Ship)*: This problem focuses on the construction of gain schedulers for the tanker ship.

(a) Repeat Design Problem 7.2 for the tanker ship defined in Design Problem 6.1 on page 407 for the case where there are ballast conditions.

(b) Suppose that you can also obtain from the ship's captain an indication of the weight of the ship (e.g., when it is "full"). Explain how to design a gain scheduler using this information. Do not fully design the gain scheduler; simply provide a block diagram explaining how it would operate.

CHAPTER 8

Perspectives on Fuzzy Control

He who knows only his own side of the case,
knows little of that.
–John Stuart Mill

8.1 Overview

Fuzzy control does not exist as an isolated topic devoid of relationships to other fields, and it is important to understand how it relates to these other fields in order to strengthen your understanding of it. We have emphasized that fuzzy control has its foundations in conventional control and that there are many relationships to techniques, ideas, and methodologies there. Fuzzy control is also an "intelligent control" technique, and hence there are certain relationships between it and other intelligent control methods. In this chapter we will provide a brief overview of some of the basic relationships between fuzzy control and other control methods. This will give the reader who has a good understanding of fuzzy control a glimpse of related topics in other areas. Moreover, it will give the reader who has a good understanding of other areas of control an idea of what the field of fuzzy control is concerned with.

We begin the chapter in Section 8.2 by providing a conventional control engineering perspective on fuzzy control. This is essentially a summary of many of the points that we have made throughout the text, but here we bring them all together. Following this, in Section 8.3 we introduce two popular areas in neural networks, the multilayer perceptron and the radial basis function neural network. We explain that a class of radial basis function neural networks is identical to a class of fuzzy systems. Moreover, we explain how techniques covered in this book (e.g., gradient training and adaptive control) can be used for neural networks. In Section 8.4 we

explain genetic algorithms, their relationship to the field of control, and particularly their use with fuzzy systems. Next, in Section 8.5 we provide an overview of some of the relationships to knowledge-based systems, particularly expert systems (and hence expert control) and planning systems. Finally, in Section 8.6 we provide an overview of the general area of (hierarchical) intelligent and autonomous control where we offer some ideas on how to define the field of intelligent control and how some of the most general intelligent controllers operate. We use an "intelligent vehicle highway system" problem to illustrate the use of the intelligent autonomous controller functional architecture.

This chapter is meant to provide a view of, and motivation for, the main areas in the field of intelligent control. The reader interested only in fuzzy control can certainly ignore this chapter; we do not, however, advise this as the relationships to other fields often suggest ideas on how to expand the basic fuzzy control methods and may provide key ideas on how to solve a control problem for a particular application.

8.2 Fuzzy Versus Conventional Control

What are the advantages and disadvantages of fuzzy control as compared to conventional control? What are the perspectives of conventional control engineers on fuzzy control? In this section we will attempt to give answers to these questions by asking, and at least partially answering, a series of questions that we have accumulated over the years from a variety of engineers in industry and universities concerned about whether to use fuzzy or conventional control. We break the questions into three categories and use the questions to summarize several points made in earlier chapters.

8.2.1 Modeling Issues and Design Methodology

First, we will discuss several issues related to modeling and the overall fuzzy controller design methodology.

1. Is the fuzzy controller design methodology viable? Success in a variety of applications (e.g., the flexible-link robot application studied in this book) has proven fuzzy control to be a viable methodology and therefore worthy of consideration.

2. Do engineers like the methodology? Some do, and some do not. Engineers who have found success with it tend to like it. Often, we find that if engineers invest the time into learning it, they find it to be a tool with which they are comfortable working (they feel like it is "one more tool in their toolbox"). This may be because fuzzy systems are interpolators and engineers are used to thinking about using interpolation as a solution to a wide variety of problems.

3. Will the methodology always work? No. The reason we can be so definite in this answer is that it is not the methodology that ultimately leads to success; it is the clever ideas that the control engineer uses to achieve high-performance

control. Fuzzy control is a vehicle, and the engineer is the driver. Some find that the vehicle is comfortable and that they can coax it into performing all kinds of functions for them. Others are not so comfortable with it.

4. Does the design methodology always shorten the "lead time" to design and implementation? In talking with many people in industry, we have found that most often it does (and this is very important, especially in today's competitive climate), but we have also heard of instances where people factor in the cost of having their engineers learn the method and then found the membership functions very hard to tune. In these cases the clear answer from the engineers was that it did not make things easier. We have heard from some that fuzzy logic implements, in a similar way, the standard logic and interpolation methods they already use. Sometimes such engineers find that the fuzzy control jargon clouds the issues that are central to the control problem. Others like that it helps to formalize what they have been doing and helps to suggest ideas for other approaches.

5. Is a model used in the fuzzy control design methodology? It is possible that a mathematical model is not used. However, often it is used in simulation to redesign a fuzzy controller. Others argue that a model is always used: even if it is not written down, some type of model is used "in your head."

6. Since most people claim that no formal model is used in the fuzzy control design methodology, the following questions arise:

   (a) Is it not true that there are few, if any, assumptions to be violated by fuzzy control and that the technique can be indiscriminately applied? Yes, and sometimes it is applied to systems where it is clear that a PID controller or look-up table would be just as effective. So, if this is the case, then why not use fuzzy control? Because it is more computationally complex than a PID controller and the PID controller is much more widely understood.

   (b) Are heuristics all that are available to perform fuzzy controller design? No. Any good models that can be used, probably should be.

   (c) By ignoring a formal model, if it is available, is it not the case that a significant amount of information about how to control the plant is ignored? Yes. If, for example, you have a model of a complex process, we often use simulations to gain an understanding of how best to control the plant—and this knowledge can be used to design a fuzzy controller.

   (d) Can standard control theoretic analysis be used to verify the operation of the resulting control system? Sometimes, if the fuzzy control system satisfies the assumptions needed for the mathematical analysis. This will be discussed in more detail in the next section.

   (e) Will it be difficult to clearly characterize the limitations of various fuzzy control techniques (i.e., to classify which plants can be controlled best with different fuzzy or conventional controllers)? Yes.

(f) Will it be difficult to clearly relate the results of using the fuzzy controller to previous work in conventional control to definitively show that contributions are being made to the field of control? Yes.

7. Is there always a formal model available for control design? No, but for most systems there is at least an approximate model available. This information is often valuable and should not be ignored.

8. Does the use of fuzzy controllers limit the design methodology as compared to the use of more general expert controllers? Expert controllers use more general knowledge-representation schemes and inference strategies (see more details in Section 8.5.1), so for some plants it may be advantageous to use the expert controller. It is, however, not clear at this point what class of plants call for the use of expert control.

8.2.2 Stability and Performance Analysis

Next, we will discuss several issues related to the performance analysis of fuzzy control systems.

1. Is verification and certification of fuzzy control systems important? Yes, especially for safety-critical systems (e.g., an aircraft). It may not be as important for certain applications (e.g., a washing machine with a fuzzy control system).

2. What are the roles of simulation and implementation in evaluating the performance of fuzzy control systems? They play exactly the same role as for conventional control systems.

3. What are the roles of the following nonlinear analysis approaches in fuzzy control system design?

   (a) Phase plane analysis (see the references at the end of Chapter 4, in Section 4.9 on page 223).

   (b) Describing function analysis (see Chapter 4, Section 4.6 on page 214).

   (c) Stability analysis: Lyapunov's first and second methods (see Chapter 4, Section 4.3 on page 193); absolute stability (see Section 4.4 on page 204); and the small gain theorem (see Section 4.9 on page 223).

   (d) Analysis of steady-state errors (see Chapter 4, Section 4.5 on page 210) .

   (e) Method of equivalent gains (see Section 4.9 on page 223).

   (f) Cell-to-cell mapping approaches (see the references at the end of Chapter 4 in Section 4.9 on page 223).

   Several of these approaches may apply to the analysis of the behavior of the fuzzy control system you design. As you can see, in Chapter 4 we discussed many of these topics.

4. What are the problems with utilizing mathematical analysis for fuzzy control system verification? The techniques take time to learn. The problems for which fuzzy control are particularly well suited, and where there is often very good motivation to use fuzzy rather than conventional control, are the control problems where the plant has complex nonlinear behavior, and where a model is hard to derive due to inherent uncertainties. Each of these characteristics often makes the assumptions that are needed for the nonlinear analysis techniques invalid, so the theory often does not end up offering much when it is really needed.

5. Does fuzzy control provide "robust control"? If so, can this be demonstrated mathematically or experimentally? There has been a recent focus in research on stability analysis to show that fuzzy control does provide robust control. It is very difficult, of course, to show robustness via experimentation since by its very definition robustness verification requires extensive experimentation (e.g., you could not call the fuzzy controller for the rotational inverted pendulum case study in Chapter 3 or Chapter 6 "robust" when it was only shown to be successful for one disturbance condition).

8.2.3 Implementation and General Issues

Finally, we will discuss several issues related to implementation and the overall fuzzy controller design methodology.

1. Are there computational advantages in using fuzzy control as compared to conventional control? Not always. PID control is simpler than fuzzy control; however, there are some types of conventional control that are very difficult to implement where a fuzzy controller can be simpler. It depends on the application and the methods you choose.

2. Should I use a conventional or "fuzzy processor" for implementation? We have typically found that our needs can be met if we use a conventional processor that has a better track record with reliability; however, there may be some advantages to fuzzy processors when large rule-bases are used and fast sampling times are needed.

3. Are there special "tricks of the trade" in the implementation of fuzzy controllers that have many rules? Yes. Several of these are listed in Chapter 2, in Section 2.6 beginning on page 97.

4. Does fuzzy control provide for a user-friendly way to tune the controller during implementation studies? Often it does. We have found in field studies that when you know generally what to do to get a controller to work, it is sometimes hard to get this information into the gains of a conventional controller and easier to express it in rules and load them into a fuzzy controller or fuzzy supervisor.

Overall, in comparing fuzzy to conventional control, it is interesting to note that there are conventional control schemes that are analogous to fuzzy ones: (1) direct fuzzy control is analogous to direct nonlinear control, (2) fuzzy adaptive control is analogous to conventional adaptive control (e.g., model reference adaptive control), and (3) fuzzy supervisory control is analogous to hierarchical control. Does there exist an analogous conventional approach to every fuzzy control scheme? If so, then in doing fuzzy control research it seems to be very important to compare and contrast the performance of the fuzzy versus the conventional approaches.

8.3 Neural Networks

Artificial neural networks are circuits, computer algorithms, or mathematical representations of the massively connected set of neurons that form biological neural networks. They have been shown to be useful as an alternative computing technology and have proven useful in a variety of pattern recognition, signal processing, estimation, and control problems. Their capabilities to learn from examples have been particularly useful.

In this section we will introduce two of the more popular neural networks and discuss how they relate to the areas of fuzzy systems and control. We must emphasize that there are many topics in the area of neural networks that are not covered here. For instance, we do not discuss associative memories and Hopfield neural networks, recurrent networks, Boltzmann machines, or Hebbian or competitive learning. We refer the reader to Section 8.8, For Further Study, for references that cover these topics in detail.

8.3.1 Multilayer Perceptrons

The multilayer perceptron is a feed-forward neural network (i.e., it does not use past values of its outputs or other internal variables to compute its current output). It is composed of an interconnection of basic neuron processing units.

The Neuron

For a single neuron, suppose that we use x_i, $i = 1, 2, \ldots, n$, to denote its inputs and suppose that it has a single output y. Figure 8.1 shows the neuron. Such a neuron first forms a weighted sum of the inputs

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) - \theta$$

where w_i are the interconnection "weights" and θ is the "bias" for the neuron (these parameters model the interconnections between the cell bodies in the neurons of a biological neural network). The signal z represents a signal in the biological neuron, and the processing that the neuron performs on this signal is represented with an "activation function." This activation function is represented with a function f,

and the output that it computes is

$$y = f(z) = f\left(\left(\sum_{i=1}^{n} w_i x_i\right) - \theta\right) \tag{8.1}$$

Basically, the neuron model represents the biological neuron that "fires" (turns on) when its inputs are significantly excited (i.e., z is big enough). The manner in which the neuron fires is defined by the activation function f. There are many ways to define the activation function:

- *Threshold function:* For this type of activation function we have

$$f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

  so that once the input signal z is above zero the neuron turns on.

- *Sigmoid function:* For this type of activation function we have

$$f(z) = \frac{1}{1 + \exp(-bz)} \tag{8.2}$$

  so that the input signal z continuously turns on the neuron an increasing amount as it increases (plot the function values against z to convince yourself of this). The parameter b affects the slope of the sigmoid function. There are many functions that take on a shape that is sigmoidal. For instance, one that is often used in neural networks is the hyperbolic tangent function

$$f(z) = \tanh\left(\frac{z}{2}\right) = \frac{1 - \exp(z)}{1 + \exp(z)}$$

Equation (8.1), with one of the above activation functions, represents the computations made by one neuron in the neural network. Next, we define how we interconnect these neurons to form a neural network—in particular, the multilayer perceptron.

Network of Neurons

The basic structure for the multilayer perceptron is shown in Figure 8.2. There, the circles represent the neurons (weights, bias, and activation function) and the lines represent the connections between the inputs and neurons, and between the neurons in one layer and those in the next layer. This is a three-layer perceptron since there are three stages of neural processing between the inputs and outputs. More layers can be added by concatenating additional "hidden" layers of neurons.

The multilayer perceptron has inputs x_i, $i = 1, 2, \ldots, n$, and outputs y_j, $j = 1, 2, \ldots, m$. The number of neurons in the first hidden layer (see Figure 8.2) is n_1. In the second hidden layer there are n_2 neurons, and in the output layer there are

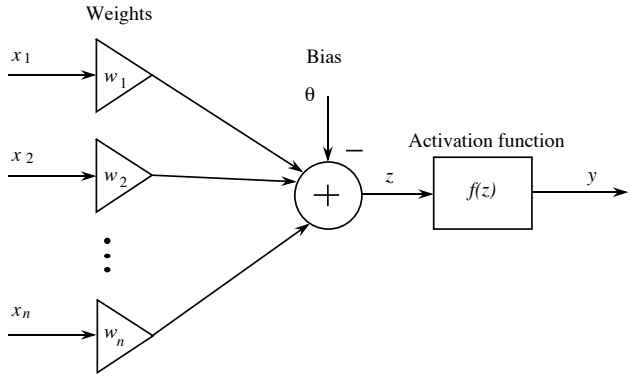


FIGURE 8.1 Single neuron model.

m neurons. Hence, in an N layer perceptron there are n_i neurons in the i^{th} hidden layer, $i = 1, 2, \ldots, N-1$.

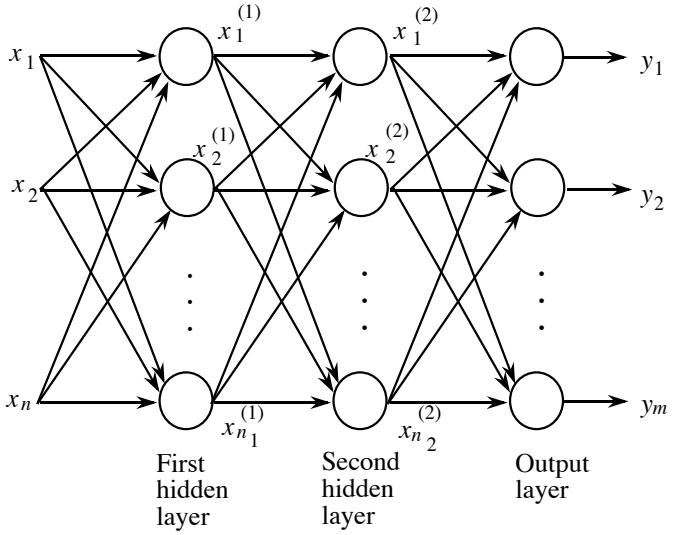


FIGURE 8.2 Multilayer perceptron model.

The neurons in the first layer of the multilayer perceptron perform computations, and the outputs of these neurons are given by

$$x_j^{(1)} = f_j^{(1)} \left(\left(\sum_{i=1}^{n} w_{ij}^{(1)} x_i \right) - \theta_j^{(1)} \right)$$

with $j = 1, 2, \ldots, n_1$. The neurons in the second layer of the multilayer perceptron perform computations, and the outputs of these neurons are given by

$$x_j^{(2)} = f_j^{(2)} \left(\left(\sum_{i=1}^{n_1} w_{ij}^{(2)} x_i^{(1)} \right) - \theta_j^{(2)} \right)$$

with $j = 1, 2, \ldots, n_2$. The neurons in the third layer of the multilayer perceptron perform computations, and the outputs of these neurons are given by

$$y_j = f_j \left(\left(\sum_{i=1}^{n_2} w_{ij} x_i^{(2)} \right) - \theta_j \right)$$

with $j = 1, 2, \ldots, m$.

The parameters (scalar real numbers) $w_{ij}^{(1)}$ are called the weights of the first hidden layer. The $w_{ij}^{(2)}$ are called the weights of the second hidden layer. The w_{ij} are called the weights of the output layer. The parameters $\theta_j^{(1)}$ are called the biases of the first hidden layer. The parameters $\theta_j^{(2)}$ are called the biases of the second hidden layer, and the θ_j are the biases of the output layer. The functions f_j (for the output layer), $f_j^{(2)}$ (for the second hidden layer), and $f_j^{(1)}$ (for the first hidden layer) represent the activation functions. The activation functions can be different for each neuron in the multilayer perceptron (e.g., the first layer could have one type of sigmoid, while the next two layers could have different sigmoid functions or threshold functions).

This completes the definition of the multilayer perceptron. Next, we will introduce the radial basis function neural network. After that we explain how both of these neural networks relate to the other topics covered in this book.

8.3.2 Radial Basis Function Neural Networks

A locally tuned, overlapping receptive field is found in parts of the cerebral cortex, in the visual cortex, and in other parts of the brain. The radial basis function neural network model is based on these biological systems.

A radial basis function neural network is shown in Figure 8.3. There, the inputs are x_i, $i = 1, 2, \ldots, n$, and the output is $y = f(x)$ where f represents the processing by the entire radial basis function neural network. Let $x = [x_1, x_2, \ldots, x_n]^\top$. The input to the i^{th} receptive field unit is x, and its output is denoted with $R_i(x)$. It has what is called a "strength" which we denote by $\bar{y}_i$. Assume that there are M receptive field units. Hence, from Figure 8.3,

$$y = f(x) = \sum_{i=1}^{M} \bar{y}_i R_i(x) \tag{8.3}$$

is the output of the radial basis function neural network.

There are several possible choices for the "receptive field units" $R_i(x)$:

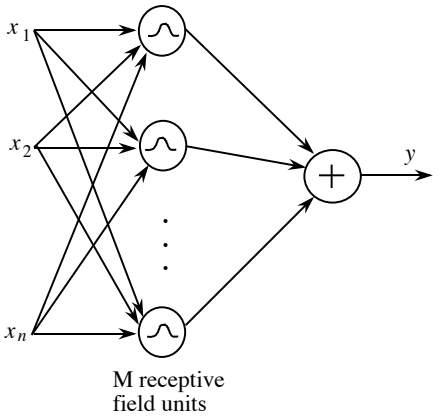


FIGURE 8.3 Radial basis function neural network model.

1. We could choose

$$R_i(x) = \exp\left(-\frac{|x - \underline{c}_i|^2}{\sigma_i^2}\right)$$

where $\underline{c}_i = [c_1^i, c_2^i, \ldots, c_n^i]^\top$, σ_i is a scalar, and if z is a vector then $|z| = \sqrt{z^\top z}$.

2. We could choose

$$R_i(x) = \frac{1}{1 + \exp\left(-\frac{|x-\underline{c}_i|^2}{\sigma_i^2}\right)}$$

where $\underline{c}_i$ and σ_i are defined in choice 1.

There are also alternatives to how to compute the output of the radial basis function neural network. For instance, rather than computing the simple sum as in Equation (8.3), you could compute a weighted average

$$y = f(x) = \frac{\sum_{i=1}^M \bar{y}_i R_i(x)}{\sum_{i=1}^M R_i(x)} \tag{8.4}$$

It is also possible to define multilayer radial basis function neural networks.

This completes the definition of the radial basis function neural network. Next, we explain the relationships between multilayer perceptrons and radial basis function neural networks and fuzzy systems.

8.3.3 Relationships Between Fuzzy Systems and Neural Networks

There are two ways in which there are relationships between fuzzy systems and neural networks. First, techniques from one area can be used in the other. Second, in some cases the functionality (i.e., the nonlinear function that they implement) is identical. Some label the intersection between fuzzy systems and neural networks with the term "fuzzy-neural" or "neuro-fuzzy" to highlight that techniques from both fields are being used. Here, we avoid this terminology and simply highlight the basic relationships between the two fields.

Multilayer Perceptrons

The multilayer perceptron should be viewed as a nonlinear network whose nonlinearity can be tuned by changing the weights, biases, and parameters of the activation functions. The fuzzy system is also a tunable nonlinearity whose shape can be changed by tuning, for example, the membership functions. Since both are tunable nonlinearities, the following approaches are possible:

- Gradient methods can be used for training neural networks to perform system identification or to act as estimators or predictors in the same way as fuzzy systems were trained in Chapter 5, Section 5.4, beginning on page 260. Indeed, the gradient training of neural networks, called "back-propagation training," was introduced well before the gradient training of fuzzy systems, and the idea for training fuzzy systems this way came from the field of neural networks.
- Hybrid methods for training, which were covered in Chapter 5, Section 5.7, on page 291, can also be used for neural networks. For instance, gradient methods may be used in conjunction with clustering methods applied to neural networks.
- Indirect adaptive control, which was covered in Chapter 6, Section 6.6 on page 394, can also be achieved with a multilayer perceptron. To do this we use two multilayer perceptrons as the tunable nonlinearities in the certainty equivalence control law and the gradient method for tuning.
- Gain scheduled control, covered in Chapter 7, Section 7.2.2, on page 417 may be achieved by training a multilayer perceptron to map the associations between operating conditions and controller parameters.

This list is by no means exhaustive. It simply shows that multilayer perceptron networks can take on a similar role to that of a fuzzy system in performing the function of being a tunable nonlinearity. An advantage that the fuzzy system may have, however, is that it often facilitates the incorporation of heuristic knowledge into the solution to the problem, which can, at times, have a significant impact on the quality of the solution (see Section 5.2.4 on page 241).

Radial Basis Function Neural Networks

Some radial basis function neural networks are *equivalent* to some standard fuzzy systems in the sense that they are functionally equivalent (i.e., given the same inputs, they will produce the same outputs). To see this, suppose that in Equation (8.4) we let $M = R$ (i.e., the number of receptive field units equal to the number of rules), $\bar{y}_i = b_i$ (i.e., the receptive field unit strengths equal to the output membership function centers), and choose the receptive field units as

$$R_i(x) = \mu_i(x)$$

(i.e., choose the receptive field units to be the same as the premise membership functions). In this case we see that the radial basis function neural network is *identical* to a certain fuzzy system that uses center-average defuzzification. This fuzzy system is then given by

$$y = f(x) = \frac{\sum_{i=1}^{R} b_i \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

It is also interesting to note that the functional fuzzy system (the more general version of the Takagi-Sugeno fuzzy system), which was covered in Section 2.3.7 on page 73, is equivalent to a class of two-layer neural networks [200].

The equivalence between this type of fuzzy system and a radial basis function neural network shows that *all the techniques in this book for the above type of fuzzy system work in the same way for the above type of radial basis function neural network (or, using [200], the techniques for the Takagi-Sugeno fuzzy system can be used for a type of multilayer radial basis function neural network).*

Due to the above relationships between fuzzy systems and neural networks, some would like to view fuzzy systems and neural networks as identical areas. This is, however, not the case for the following reasons:

- There are classes of neural networks (e.g., dynamic neural networks) that may have a fuzzy system analog, but if so it would have to include not only standard fuzzy components but some form of a differential equation component.
- There are certain fuzzy systems that have no clear neural analog. Consider, for example, certain "fuzzy dynamic systems" [48, 167]. We can, however, envision how you could go about designing a neural analog to such fuzzy systems.
- The neural network has traditionally been a "black box" approach where the weights and biases are trained (e.g., using gradient methods like back-propagation) using data, often without using extra heuristic knowledge we often have. In fuzzy systems you can incorporate heuristic information and use data to train them (see Chapter 5, Section 5.2.4 on page 241). This last difference is often quoted as being one of the advantages of fuzzy systems over neural networks, at least for some applications.

Regardless of the differences, it is important to note that many methods in neural control (i.e., when we use a neural network for the control of a system) are quite similar to those in adaptive fuzzy control. For instance, since the fuzzy system and radial basis function neural network can be linearly parameterized (see [200]), we can use them as the identifier structures in direct or indirect adaptive control schemes and use gradient or least squares methods to update the parameters (see Chapters 5 and 6 and [200, 229, 50]). Indeed, we could have used neural networks as the structure that we trained for all of the identification methods in Chapter 5. In this sense we can use neural networks in system identification, estimation, and prediction, and as a direct (fixed) controller that is trained with input-output data. Basically, to be fluent with the methods of adaptive fuzzy systems and control, you must know the methods of neural control—and vice versa.

8.4 Genetic Algorithms

A genetic algorithm (GA) uses the principles of evolution, natural selection, and genetics from natural biological systems in a computer algorithm to simulate evolution. Essentially, the genetic algorithm is an optimization technique that performs a parallel, stochastic, but directed search to evolve the most fit population. In this section we will introduce the genetic algorithm and explain how it can be used for design and tuning of fuzzy systems.

8.4.1 Genetic Algorithms: A Tutorial

The genetic algorithm borrows ideas from and attempts to simulate Darwin's theory on natural selection and Mendel's work in genetics on inheritance. The genetic algorithm is an optimization technique that evaluates more than one area of the search space and can discover more than one solution to a problem. In particular, it provides a stochastic optimization method where if it "gets stuck" at a local optimum, it tries to simultaneously find other parts of the search space and "jump out" of the local optimum to a global one.

Representation and the Population of Individuals

The "fitness function" measures the fitness of an individual to survive in a population of individuals. The genetic algorithm will seek to maximize the fitness function $J(\theta)$ by selecting the individuals that we represent with θ. To represent the genetic algorithm in a computer, we make θ a string. In particular, we show such a string in Figure 8.4. A string is a chromosome in a biological system. It is a string of "genes" that can can take on different "alleles." In a computer we use number systems to encode alleles. Hence, a gene is a "digit location" that can take on different values from a number system (i.e., different types of alleles).

For instance, in a base-2 number system, alleles come from the set $\{0, 1\}$, while in a base-10 number system, alleles come from the set $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Hence, a binary chromosome has zeros or ones in its gene locations. As an example,

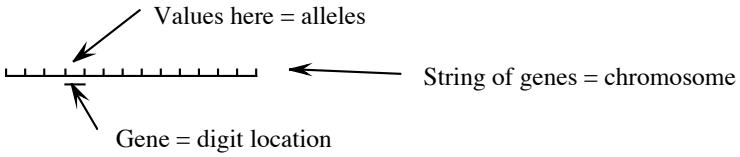


FIGURE 8.4 String for representing an individual.

consider the binary chromosome

$$1011110001010$$

which is a binary string of length 13. If we are seeking to optimize parameters of a system that come in a base-10 number system then we will need to encode the numbers into the binary number system (using the standard method for the conversion of base-10 numbers to base-2 numbers). We will also need to decode the binary strings into base-10 numbers to use them. Here, we will develop the genetic algorithm for base-2 or base-10 number systems but we will favor the use of the base-10 representation since then there is no need for encoding or decoding (which can be computationally expensive for on-line, real-time applications).

As an example of a base-10 chromosome, consider

$$8219345127066$$

that has 13 gene positions. For such chromosomes we add a gene for the sign of the number (either "+" or "−") and fix a position for the decimal point. For instance, for the above chromosome we could have

$$+821934.5127066$$

where there is no need to carry along the decimal point; the computer will just have to remember its position. Note that you could also use a floating point representation where we could code numbers in a fixed-length string plus the number in the exponent (e.g., as $XXX \times 10^{YY}$). The ideas developed here work just as readily for this number representation system as for standard base-2 or base-10.

Since we are interested in applying the genetic algorithm to controller or estimator design and tuning, we will have as individuals parameters that represent, for instance, a conventional or fuzzy controller (i.e., a vector of parameters). The vector of parameters that encodes a fuzzy or conventional controller can be loaded into a single chromosome. For example, suppose that you have a PD controller with

$$K_p = +5.12,\ K_d = -2.137$$

then we would represent this in a chromosome as

$$+051200 - 021370$$

which is a concatenation of the digits, where we assume that there are six digits for the representation of each parameter plus the sign digit (this is why you see the extra padding of zeros). The computer will have to keep track of where the decimal point is. We see that each chromosome will have a certain structure (its "genotype" in biological terms), but here rather than a set of chromosomes for the structure we just concatenate the parameters and use one one chromosome for convenience. Each chromosome represents a point in the search space of the genetic algorithm (i.e., a "phenotype" in biological terms).

Next, we develop a notation for representing a whole set of individuals (i.e., a population). Let $\theta_i^j(k)$ be a single parameter at time k (a fixed-length string with sign digit), and suppose that chromosome j is composed of N of these parameters, which are sometimes called "traits." Let

$$\theta^j(k) = \left[\theta_1^j(k), \theta_2^j(k), \ldots, \theta_N^j(k)\right]^\top$$

be the j^{th} chromosome. Note that earlier we had concatenated elements in a string while here we simply take the concatenated elements and form a vector from them. We do this simply because this is probably the way that you will want to code the algorithm in the computer. We will at times, however, still let θ^j be a concatenated string when it is convenient to do so.

The population of individuals at time k is given by

$$P(k) = \left\{\theta^j(k) | j = 1, 2, \ldots, S\right\} \tag{8.5}$$

and the number of individuals in the population is given by S. We want to pick S to be big enough so that the population elements can cover the search space. However, we do not want S to be too big since this increases the number of computations we have to perform.

Genetic Operations

The population $P(k)$ at time k is often referred to as the "generation" of individuals at time k. Evolution occurs as we go from a generation at time k to the next generation at time $k+1$. Genetic operations of selection, crossover, and mutation are used to produce one generation from the next.

Selection: Basically, according to Darwin the most qualified individuals survive to mate. We quantify "most qualified" via an individual's fitness $J(\theta^j(k))$ at time k. For selection we create a "mating pool" at time k, which we denote by

$$M(k) = \left\{m^j(k) | j = 1, 2, \ldots, S\right\} \tag{8.6}$$

The mating pool is the set of chromosomes that are selected for mating. We select an individual for mating by letting each $m^j(k)$ be equal to $\theta^i(k) \in P(k)$ with

probability

$$p_i = \frac{J(\theta^i(k))}{\sum_{j=1}^{S} J(\theta^j(k))} \tag{8.7}$$

To clarify the meaning of this formula and hence the selection strategy, Goldberg [58] uses the analogy of spinning a unit circumference roulette wheel where the wheel is cut like a pie into S regions where the i^{th} region is associated with the i^{th} element of $P(k)$. Each pie-shaped region has a portion of the circumference that is given by p_i in Equation (8.7). You spin the wheel, and if the pointer points at region i when the wheel stops, then you place θ^i into the mating pool $M(k)$. You spin the wheel S times so that S elements end up in the mating pool. Clearly, individuals who are more fit will end up with more copies in the mating pool; hence, chromosomes with larger-than-average fitness will embody a greater portion of the next generation. At the same time, due to the probabilistic nature of the selection process, it is possible that some relatively unfit individuals may end up in the mating pool.

Reproduction Phase, Crossover: We think of crossover as mating in biological terms, which at a fundamental biological level involves the process of combining chromosomes. The crossover operation operates on the mating pool $M(k)$. First, you specify the "crossover probability" p_c (usually chosen to be near one since when mating occurs in biological systems, genetic material is swapped between the parents). The procedure for crossover consists of the following steps:

1. Randomly pair off the individuals in the mating pool $M(k)$ (i.e., form pairs to mate by the flip of a coin). If there are an odd number of individuals in $M(k)$, then, for instance, simply take the last individual and pair it off with another individual who has already been paired off.

2. Consider chromosome pair θ^j, θ^i that was formed in step 1. Generate a random number $r \in [0, 1]$.

   (a) If $r < p_c$ then cross over θ^j and θ^i. To cross over these chromosomes select at random a "cross site" and exchange all the digits to the right of the cross site of one string with those of the other. This process is pictured in Figure 8.5. In this example the cross site is position five on the string, and hence we swap the last eight digits between the two strings. Clearly, the cross site is a random number between one and the number of digits in the string minus one.

   (b) If $r > p_c$ then we will not cross over; hence, we do not modify the strings, and we go to the mutation operation below.

3. Repeat step 2 for each pair of strings that is in $M(k)$.

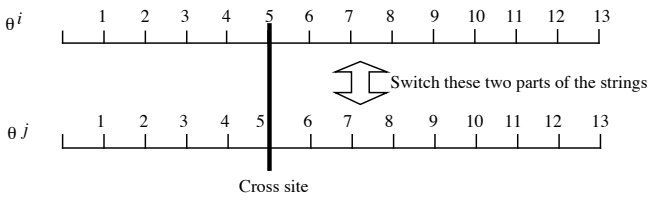


FIGURE 8.5 Crossover operation example.

As an example, suppose that $S = 10$ and that in step 1 above we randomly pair off the chromosomes. Suppose that θ^5 and θ^9 $(j = 5, i = 9)$ are paired off where

$$\theta^5 = +2.9845$$

and

$$\theta^9 = +1.9322$$

Suppose that $p_c = 0.9$ and that when we randomly generate r we get $r = 0.34$. Hence, by step 2 we will cross over θ^5 and θ^9. According to step 2 we randomly pick the cross site. Suppose that it is chosen to be position three on the string. In this case the strings that are produced by crossover are

$$\theta^5 = +2.9322$$

and

$$\theta^9 = +1.9845$$

Besides the fact that crossover helps to model the mating part of the evolution process, why should the genetic algorithm perform crossover? Basically, the crossover operation perturbs the parameters near good positions to try to find better solutions to the optimization problem. It tends to help perform a localized search around the more fit individuals (since on average the individuals in the mating pool at time k are more fit than the ones in the population at time k).

Reproduction Phase, Mutation: Like crossover, mutation modifies the mating pool (i.e., after selection has taken place). The operation of mutation is normally performed on the elements in the mating pool after crossover has been performed. The biological analog of our mutation operation is the random mutation of genetic material. To perform mutation in the computer, first choose a mutation probability p_m. With probability p_m, change (mutate) each gene location on each chromosome randomly to a member of the number system being used. For instance, in a base-2 genetic algorithm, we could mutate

1010111

to

1011111

where the fourth bit was mutated to one. For a base-10 number system you would simply pick a number at random to replace a digit with if you are going to mutate a digit location.

Besides the fact that this helps to model mutation in a biological system, why should the genetic algorithm perform mutation? Basically, it provides random excursions into new parts of the search space. It is possible that we will get lucky and mutate to a good solution. It is the mechanism that tries to make sure that we do not get stuck at a local maxima and that we seek to explore other areas of the search space to help find a global maximum for $J(\theta)$. Usually, the mutation probability is chosen to be quite small (e.g., less than 0.01) since this will help guarantee that all the individuals in the mating pool are not mutated so that any search progress that was made is lost (i.e., we keep it relatively low to avoid degradation to exhaustive search via a random walk in the search space).

After mutation we get a modified mating pool at time k, $M(k)$. To form the next generation for the population, we let

$$P(k+1) = M(k)$$

where this $M(k)$ is the one that was formed by selection and modified by crossover and mutation. Then the above steps repeat, successive generations are produced, and we thereby model evolution (of course it is a very crude model).

Optional Features: There have been many different options used in the definition of a genetic algorithm. These include the following:

- There is the possibility of using other genetic operators. For instance, there is an operation called "elitism" where the most fit individual in $P(k)$ is copied directly to $P(k+1)$ without being changed by the other operations. This operator is sometimes used to try to make sure that there will be a reasonably fit individual present in the population at every time step; it helps to avoid having all the strings get modified by crossover and mutation in a way so that no good solution exists at some time k.

- Some use a "population splitting" approach where the population of S members is partitioned into subsets and the genetic operations are constrained to only mix within these partitions. This can produce different subpopulations that will seek different solutions.

- There are many options for the crossover operation. For instance, some consider crossing over at every site in the chromosome. Others will perform crossover for each separate parameter (trait) on the chromosome.

- Some will grow and shrink the population.

There are many other options besides the ones listed above. The interested reader should consult Section 8.8, For Further Study, for more details about genetic algorithms.

Termination Conditions, Initialization, and Implementation Issues

The above discussion showed how to produce successive generations and thereby simulate evolution. While the biological evolutionary process continues, perhaps indefinitely, there are many times when we would like to terminate our artificial one and find the following:

- The population individual—say, $\theta^*(k)$—that best maximizes the fitness function. Notice that to determine this we also need to know the generation number k where the most fit individual existed (it is not necessarily in the last generation). You may want to design the computer code that implements the genetic algorithm to always keep track of the highest J value and the generation number and individual that achieved this value of J.
- The value of the fitness function $J(\theta^*(k))$. While for some applications this value may not be important, for others it is critical (e.g., in many function optimization problems).
- Information about the way that the population has evolved, which areas of the search space were visited, and how the fitness function has evolved over time. You may want to design the code that implements the genetic algorithm to provide plots or printouts of all the relevant genetic algorithm data.

There is then the question of how to terminate the genetic algorithm. There are many ways to terminate a genetic algorithm, many of them similar to termination conditions used for conventional optimization algorithms. To introduce a few of these, let $\epsilon > 0$ be a small number and $M_1 > 0$ and $M_2 > 0$ be integers. Consider the following options for terminating the genetic algorithm:

- Stop the algorithm after generating generation $P(M_2)$—that is, after M_2 generations.
- Stop the algorithm after at least M_2 generations have occurred and at least M_1 steps have occurred where the maximum (or average) value of J for all population members has increased by no more than ϵ.
- Stop the algorithm once J takes on a value above some fixed value.

Of course, there are other possibilities for termination conditions. The above ones are easy to implement on a computer but sometimes you may want to watch the parameters evolve and decide yourself when to stop the algorithm.

Initialization of the genetic algorithm is done by first choosing the representation to be used (including the structure of the chromosomes, the number base,

and the number of digits to be used). Next, you need to specify the size of the population, decide which genetic operations will be used, specify the crossover and mutation probabilities p_c and p_m, and pick a termination method (if it is needed).

Sometimes for problems that are solved by the genetic algorithm it is known that the parameters that are manipulated by the genetic algorithm will lie in a certain fixed range (e.g., you may know that you would never want to make the proportional gain of a PID controller negative). Suppose, for the sake of discussion, that θ is a scalar and we know a priori that it will stay in a certain interval—say, $[\theta_{min}, \theta_{max}]$. It is important to note that crossover and mutation can generate strings that are out of a fixed range even if parameters all start within the proper ranges (provide an example of this). Due to this there is a problem when it comes to implementing the genetic algorithm of what to do when the algorithm generates a chromosome that is out of range. There are several approaches to solving this problem. For instance, if a scalar parameter θ is to lie in $[\theta_{min}, \theta_{max}]$, and at time k crossover or mutation makes $\theta(k) > \theta_{max}$, then simply choose $\theta(k) = \theta_{max}$. If at time k crossover or mutation makes $\theta(k) < \theta_{min}$, then simply choose $\theta(k) = \theta_{min}$. An alternative approach would be to simply repeat the crossover or mutation operation again and hope that the newly generated parameters will be in range. Of course, this may not solve the problem since the next time they are generated they may also be out of range (and the number of tries that it takes to get in range is random).

8.4.2 Genetic Algorithms for Fuzzy System Design and Tuning

There are basically two ways that the genetic algorithm can be used in the area of fuzzy systems: They can be used for the off-line design of fuzzy systems and in their on-line tuning. Both of these options are considered next.

Computer-Aided Design of Fuzzy Systems

The genetic algorithm can be used in the (off-line) computer-aided design of control systems since it can artificially evolve an appropriate controller that meets the performance specifications to the greatest extent possible. To do this, the genetic algorithm maintains a population of strings that each represent a different controller (digits on the strings characterize parameters of the controller), and it uses a fitness measure that characterizes the closed-loop specifications. Suppose, for instance, that the closed-loop specifications indicate that you want, for a step input, a (stable) response with a rise-time of t_r^*, a percent overshoot of M_p^*, and a settling time of t_s^*. We need to define the fitness function so that it measures how close each individual in the population at time k (i.e., each controller candidate) is to meeting these specifications. Suppose that we let t_r, M_p, and t_s denote the rise-time, overshoot, and settling time, respectively, for a given individual (we compute these for an individual in the population by performing a simulation of the closed-loop system with the candidate controller and a model of the plant). Given these values, we let

(for each individual and every time step k)

$$\bar{J} = w_1(t_r - t_r^*)^2 + w_2(M_p - M_p^*)^2 + w_3(t_s - t_s^*)^2$$

where $w_i > 0$, $i = 1, 2, 3$, are positive weighting factors. The function $\bar{J}$ characterizes how well the candidate controller meets the closed-loop specifications where if $\bar{J} = 0$ it meets the specifications perfectly. The weighting factors can be used to prioritize the importance of meeting the various specifications (e.g., a high value of w_2 relative to the others indicates that the percent overshoot specification is more important to meet than the others).

Now, we would like to minimize $\bar{J}$, but the genetic algorithm is a maximization routine. To minimize $\bar{J}$ with the genetic algorithm, we can choose the fitness function

$$J = \frac{1}{\bar{J} + \epsilon}$$

where $\epsilon > 0$ is a small positive number. Maximization of J can only be achieved by minimization of $\bar{J}$, so the desired effect is achieved. Another way to define the fitness function is to let

$$J(\theta(k)) = -\bar{J}(\theta(k)) + \max_{\theta(k)}\{\bar{J}(\theta(k))\}$$

The minus sign in front of the $\bar{J}(\theta(k))$ term turns the minimization problem into a maximization problem (to see this, consider $\bar{J}(\theta) = (\theta)^2$, where θ is a scalar, as an example). The $\max_{\theta(k)}\{\bar{J}(\theta(k))\}$ term is needed to shift the function up so that $J(\theta(k))$ is always positive. We need it positive since in selection, Equation (8.7) defines a probability that must always be positive and between one and zero.

This completes the definition of how to use a genetic algorithm for computer-aided control system design. Note that the above approach depends in no way on whether the controller that is evolved is a conventional controller (e.g., a PID controller) or a fuzzy system or neural network. For instance, you could use a Takagi-Sugeno fuzzy system or a standard fuzzy system for the controller and let the genetic algorithm tune the appropriate parameters. Moreover, we could take any of the controllers described in Chapters 6 or 7 and parameterize them and use the above approach to tune these adaptive or supervisory controllers. We have used the genetic algorithm to tune direct, adaptive, and supervisory controllers for several applications, and while this approach is computationally intensive, and we did have to make some application-dependent modifications to the above fitness evaluation approach, it did produce successful results (see, e.g., [27]).

The above approach can also be used in system identification and for the construction of estimators and predictors, just as we used gradient optimization for these in Chapter 5. The genetic algorithm can be used for the tuning of fuzzy system parameters that enter in a nonlinear fashion and can be used in conjunction with other methods from Chapter 5 to form hybrid approaches such as those de-

scribed in Section 5.7 on page 291. To use the genetic algorithm for tuning fuzzy systems as estimators, we could choose $\bar{J}$ to be Equation (5.7) or (5.6) on page 238. Then we would choose the fitness function J similarly to how we did above. We have used such an approach to construct a Takagi-Sugeno fuzzy system that acted as a gain scheduler. One possible advantage that the GA approach could offer over, for example, a gradient method is that it may be able to better avoid local optima and hence find the global optimum.

On-Line Tuning of Fuzzy Systems

Traditionally, genetic algorithms have been used for off-line design, search, and optimization. There are ways, however, to evolve controllers (fuzzy or conventional) while the system is operating, rather than in off-line design. Progress in this direction has been made by the introduction of the "genetic model reference adaptive controller" (GMRAC) shown in Figure 8.6. As in the FMRLC, the GMRAC uses a reference model to characterize the desired performance. For the GMRAC there is a genetic algorithm that maintains a population of strings each of which represents a candidate controller. This genetic algorithm uses a process model (e.g., a linear model of the process) and data from the process to evaluate the fitness of each controller in the population. It does this evaluation at each time step by simulating out into the future with each candidate controller and forming a fitness function based on the error between the predicted output for each controller and that of the reference model. Using this fitness evaluation, the genetic algorithm propagates controllers into the next generation via the standard genetic operations. The controller that is the most fit one in the population at each time step is used to control the system.

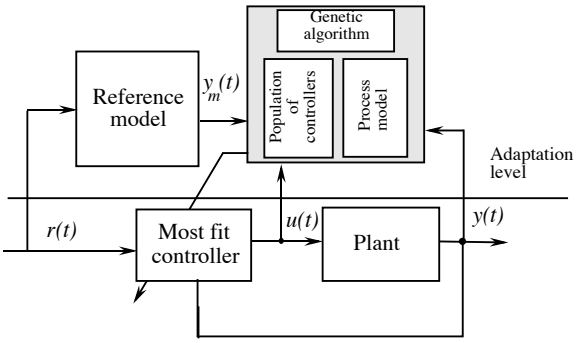


FIGURE 8.6 Genetic model reference adaptive controller (figure taken from [158], © IEEE).

This allows the GMRAC to automatically evolve a controller from generation to generation (i.e., from one time step to the next, but of course multiple generations could occur between time steps) and hence to tune a controller in response

to changes in the process or due to user change of the specifications in the reference model. Overall, the GMRAC provides unique features where alternative controllers can be quickly applied to the problem if they appear useful (e.g., the process (re)enters a new operating condition) and since it has some inherent capabilities to learn via evolution of its population of controllers. It is also possible to use the genetic algorithm in on-line tuning of estimators. The closest analogy to such an approach is the use of the gradient method for on-line estimator tuning. You can adapt the GMRAC approach above for such a purpose.

8.5 Knowledge-Based Systems

In this section we will introduce two types of knowledge-based approaches to control that can be viewed as more general forms of controllers than the basic (knowledge-based) fuzzy controller. First, we provide an overview of how to use an expert system as a controller (i.e., "expert control"); then we highlight ideas on how to use planning systems for control.

8.5.1 Expert Control

For the sake of our discussion, we will simply view the expert system that is used here as a controller for a dynamic system, as is shown in Figure 8.7. Here, we have an expert system serving as feedback controller with reference input r and feedback variable y. It uses the information in its knowledge-base and its inference mechanism to decide what command input u to generate for the plant. Conceptually, we see that the expert controller is closely related to the fuzzy controller.

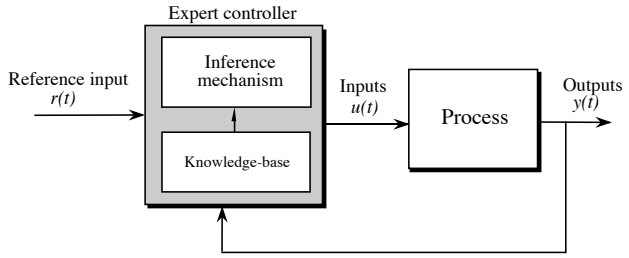


FIGURE 8.7 Expert control system.

There are, however, several differences:

1. The knowledge-base in the expert controller could be a rule-base but is not necessarily so. It could be developed using other knowledge-representation structures, such as frames, semantic nets, causal diagrams, and so on.

2. The inference mechanism in the expert controller is more general than that of the fuzzy controller. It can use more sophisticated matching strategies to

determine which rules should be allowed to fire. It can use more elaborate inference strategies. For instance, some expert systems use (a) "refraction," where if a rule has fired recently it may not be allowed back into the "conflict set" (i.e., the set of rules that are allowed to fire), (b) "recency," where rules that were fired most recently are given priority in being fired again, and (c) various other priority schemes.

It is in fact the case that an expert system is in a sense more general than a fuzzy system since it can be shown that a single rule in an expert controller can be used to represent an entire fuzzy controller [163]. From another perspective, we can "fuzzify" the expert controller components and make it a more general fuzzy system. Regardless, it is largely a waste of time to concern ourselves with which is more general. What is of concern is whether the traditional ideas from expert systems offer anything on how to design fuzzy systems. The answer is certainly affirmative. Clearly, certain theory and applications may dictate the need for different knowledge-representation schemes and inference strategies.

Next, we should note that Figure 8.7 shows a direct expert controller. As we pointed out in Chapter 7, it is also possible to use an expert system in adaptive or supervisory control systems. Expert systems can be used in a supervisory role for conventional controllers or for the supervision of fuzzy controllers (e.g., for supervision of the learning mechanism and reference model in an adaptive fuzzy controller). Expert systems themselves can also be used as the basis for general learning controllers.

8.5.2 Planning Systems for Control

Artificially intelligent planning systems (computer programs that emulate the way experts plan) have been used in path planning and high-level decisions about control tasks for robots. A generic planning system can be configured in the architecture of a standard control system, as shown in Figure 8.8. Here, the "problem domain" (the plant) is the environment that the planner operates in. There are measured outputs y_k at step k (variables of the problem domain that can be sensed in real time), control actions u_k (the ways in which we can affect the problem domain), disturbances d_k (which represent random events that can affect the problem domain and hence the measured variable y_k), and goals g_k (what we would like to achieve in the problem domain). There are closed-loop specifications that quantify performance specifications and stability requirements.

It is the task of the planner in Figure 8.8 to monitor the measured outputs and goals and generate control actions that will counteract the effects of the disturbances and result in the goals and the closed-loop specifications being achieved. To do this, the planner performs "plan generation," where it projects into the future (usually a finite number of steps, and often using a model of the problem domain) and tries to determine a set of candidate plans. Next, this set of plans is pruned to one plan that is the best one to apply at the current time (where "best" can be determined based on, e.g., consumption of resources). The plan is then executed,

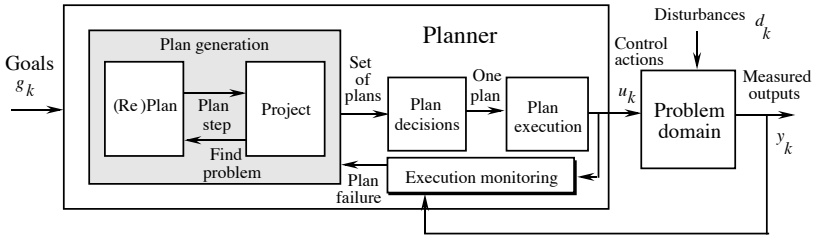


FIGURE 8.8 Closed-loop planning system (figure taken from [162], © Hemisphere Publishing Corp.).

and during execution the performance resulting from the plan is monitored and evaluated. Often, due to disturbances, plans will fail, and hence the planner must generate a new set of candidate plans, select one, then execute that one.

While not pictured in Figure 8.8, some planning systems use "situation assessment" to try to estimate the state of the problem domain (this can be useful in execution monitoring and plan generation); others perform "world modeling," where a model of the problem domain is developed in an on-line fashion (similarly to on-line system identification), and "planner design" uses information from the world modeler to tune the planner (so that it makes the right plans for the current problem domain). The reader will, perhaps, think of such a planning system as a general adaptive controller.

The role of planning systems in fuzzy control could be any one of the following: (1) the use of a fuzzy planner as a controller, (2) the use of fuzzy "situation assessment" in determining control actions, (3) the use of fuzzy "world modeling" to generate a model of the plant that is useful in making control decisions, (4) the use of a fuzzy adaptive planning system (e.g., a fuzzified version of the adaptive planner in [162]), or (5) the use of a planning system in a supervisory control role.

8.6 Intelligent and Autonomous Control

Autonomous systems have the capability to independently perform complex tasks with a high degree of success. Consumer and governmental demands for such systems are frequently forcing engineers to push many functions normally performed by humans into machines. For instance, in the emerging area of intelligent vehicle and highway systems (IVHS), engineers are designing vehicles and highways that can fully automate vehicle route selection, steering, braking, and throttle control to reduce congestion and improve safety. In avionic systems a "pilot's associate" computer program has been designed to emulate the functions of mission and tactical planning that in the past may have been performed by the copilot. In manufacturing systems, efficiency optimization and flow control are being automated, and robots are replacing humans in performing relatively complex tasks.

From a broad historical perspective, each of these applications began at a low level of automation, and through the years each has evolved into a more autonomous

system. For example, today's automotive cruise controllers are the ancestors of the controllers that achieve coordinated control of steering, braking, and throttle for autonomous vehicle driving. And the terrain following, terrain avoidance control systems for low-altitude flight are ancestors of an artificial pilot's associate that can integrate mission and tactical planning activities. The general trend has been for engineers to incrementally "add more intelligence" in response to consumer, industrial, and government demands and thereby create systems with increased levels of autonomy. In this process of enhancing autonomy by adding intelligence, engineers often study how humans solve problems, then try to directly automate their knowledge and techniques to achieve high levels of automation. Other times, engineers study how intelligent biological systems perform complex tasks, then seek to automate "nature's approach" in a computer algorithm or circuit implementation to solve a practical technological problem (e.g., in certain vision systems). Such approaches where we seek to emulate the functionality of an intelligent biological system (e.g., the human) to solve a technological problem can be collectively named "intelligent systems and control techniques." It is by using such techniques that some engineers are trying to create highly autonomous systems such as those listed above.

In this section we will explain how "intelligent" control methods can be used to create autonomous systems. First we will define "intelligent control." Next, we provide a framework for the operation of autonomous systems to clarify the ultimate goal of achieving autonomous behavior in complex technological systems.

8.6.1 What Is "Intelligent Control"?

Since the answer to this question can get rather philosophical, let us focus on a working definition that does not dwell on definitions of "intelligence" (since there is no widely accepted one partly because biological intelligence seems to have many dimensions and appears to be very complex) and issues of whether we truly model or emulate intelligence, but instead focuses on (1) the methodologies used in the construction of controllers and (2) the ability of an artificial system to perform activities normally performed by humans.

"Intelligent control" techniques offer alternatives to conventional approaches by borrowing ideas from intelligent biological systems. Such ideas can either come from humans who are, for example, experts at manually solving the control problem, or by observing how a biological system operates and using analogous techniques in the solution of control problems. For instance, we may ask a human driver to provide a detailed explanation of how she or he manually solves an automated highway system intervehicle distance control problem, then use this knowledge directly in a fuzzy controller. In another approach, we may train an artificial neural network to remember how to regulate the intervehicle spacing by repeatedly providing it with examples of how to perform such a task. After the neural network has learned the task, it can be implemented on the vehicle to regulate the intervehicle distance by recalling the proper throttle input for each value of the intervehicle distance that is sensed. In another approach, genetic algorithms may be used to automatically

synthesize and tune a control algorithm for the intervehicle spacing control problem by starting with a population of candidate controllers and then iteratively allowing the most fit controller, which is determined according to the performance specifications, to survive in an artificial evolution process implemented in a computer. In this way the controller evolves over time, successively improving its performance and adapting to its environment, until it meets the prespecified performance objectives.

Such intelligent control techniques may exploit the information represented in a mathematical model or may heavily rely on heuristics on how best to control the process. The primary difference from conventional approaches, such as PID control, is that intelligent control techniques are motivated by the functionality of intelligent biological systems, either in how they perform the control task or in how they provide an innovative solution to another problem that can be adapted to solve a control problem. This is not to say that systems that are not developed using intelligent systems and control techniques such as those listed above cannot be called "intelligent"; traditionally, we have often called any system intelligent if it is designed to perform a task that has normally been performed by humans (e.g., we use the term "intelligent" vehicle and highway systems).

A full discussion on defining intelligent control involves considering additional issues in psychology, human cognition, artificial intelligence, and control. The interested reader is referred to the articles listed at the end of this chapter for a more detailed exposition that considers these issues.

8.6.2 Architecture and Characteristics

Figure 8.9 shows a functional architecture for an intelligent autonomous controller with an interface to the process involving sensing (e.g., via conventional sensing technology, vision, touch, smell, etc.), actuation (e.g., via hydraulics, robotics, motors, etc.), and an interface to humans (e.g., a driver, pilot, crew, etc.) and other systems.

The "execution level" has low-level numeric signal processing and control algorithms (e.g., PID, optimal, adaptive, or intelligent control; parameter estimators, failure detection and identification (FDI) algorithms). The "coordination level" provides for tuning, scheduling, supervision, and redesign of the execution-level algorithms, crisis management, planning and learning capabilities for the coordination of execution-level tasks, and higher-level symbolic decision making for FDI and control algorithm management. The "management level" provides for the supervision of lower-level functions and for managing the interface to the human(s) and other systems. In particular, the management level will interact with the users in generating goals for the controller and in assessing the capabilities of the system. The management level also monitors performance of the lower-level systems, plans activities at the highest level (and in cooperation with humans), and performs high-level learning about the user and the lower-level algorithms.

Intelligent systems or intelligent controllers (e.g., fuzzy, neural, genetic, expert, and planning) can be employed as appropriate in the implementation of various functions at the three levels of the intelligent autonomous controller. For example,

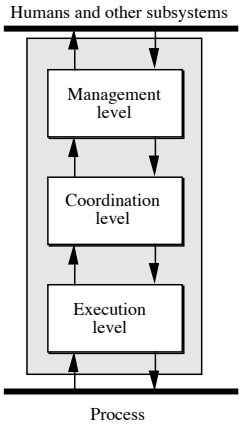


FIGURE 8.9 Intelligent autonomous controller.

adaptive fuzzy control may be used at the execution level for adaptation, genetic algorithms may be used in the coordination level to pick an optimal coordination strategy, and planning systems may be used at the management level for sequencing operations. Hierarchical controllers composed of a hybrid mix of intelligent and conventional systems are commonly used in the intelligent control of complex dynamic systems. This is because to achieve high levels of autonomy, we often need high levels of intelligence, which calls for incorporating a diversity of decision-making approaches for complex dynamic learning and reasoning.

There are several fundamental characteristics that have been identified for intelligent autonomous control systems. For example, there is generally a successive delegation of duties from the higher to lower levels, and the number of distinct tasks typically increases as we go down the hierarchy. Higher levels are often concerned with slower aspects of the system's behavior and with its larger portions, or broader aspects. There is then a smaller contextual horizon at lower levels—that is, the control decisions are made by considering less information. Higher levels are typically concerned with longer time horizons than lower levels. It is said that there is "increasing intelligence with decreasing precision as one moves from the lower to the higher levels" (see [179]). At the higher levels there is typically a decrease in time-scale density, a decrease in bandwidth or system rate, and a decrease in the decision (control action) rate. In addition, there is typically a decrease in the granularity of models used—or, equivalently, an increase in model abstractness at the higher levels.

Finally, we note that there is an ongoing evolution of the intelligent functions of an autonomous controller so that by the time you implement its functions, they no longer appear intelligent, just algorithmic. It is because of this evolution principle, doubts about our ability to implement "artificial intelligence," and the fact that

implemented intelligent controllers are nonlinear controllers, that many researchers feel more comfortable focusing on enhancing autonomy rather than achieving intelligent behavior.

8.6.3 Autonomy

Next, we explain how to incorporate the notion of autonomy into the conventional manner of thinking about control problems. Consider the general control system shown in Figure 8.10 where P is a model of the plant, C represents the controller, and T represents specifications on how we would like the closed-loop system to behave (i.e., closed-loop specifications). For some classical control problems, the scope is limited so that C and P are linear and T simply represents, for example, stability, robustness, rise-time, and overshoot specifications. In this case intelligent control techniques may not be needed. As engineers, we must remember that the simplest solution that works is the best one. We tend to need more complex controllers for more complex plants (where, for example, there is a significant amount of uncertainty) and more demanding closed loop specifications T. Consider the case where the following statements hold:

- P is so complex that it is most convenient to represent it with ordinary differential equations and discrete-event system (DES) models [70] (or some other hybrid mix of models), and for some parts of the plant the model is not known (or is too expensive to determine).
- T is used to characterize the desire to make the system perform well and act with high degrees of autonomy (i.e., via [6] "so that the system performs well under significant uncertainties in the system and its environment for extended periods of time, and compensates for significant system failures without external intervention").

The general control problem is how to construct C, given P, so that T holds. The intelligent autonomous controller described briefly in the previous section provides a general architecture for C to achieve highly autonomous behavior specified by T for very complex plants P.

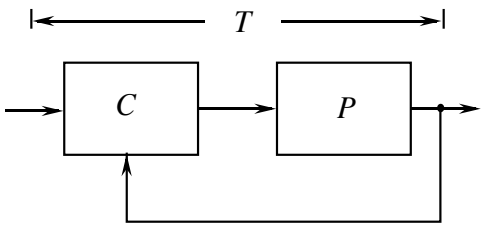


FIGURE 8.10 General control system.

From a control engineer's perspective, we are trying to solve the general control problem (i.e., we are trying to find C to enhance autonomy). Often, in practice, engineers in intelligent (and conventional) control are often examining portions of the above general control problem and trying to make incremental progress toward a solution. For example, a simple direct fuzzy controller could, perhaps, be called an "intelligent controller" (although some would never call a controller without adaptation capabilities an intelligent controller) but not an "autonomous controller," as most of them do not achieve high levels of autonomous operation but merely help enhance performance as many conventional controllers do (adaptive and supervisory approaches slightly increase performance but typically do not achieve full autonomy). It is important to note that researchers in intelligent control have been naturally led to focus on the very demanding general control problem described above for two reasons: (1) in order to address pressing needs for practical applications, and (2) since often there is a need to focus on representing more aspects of the process so that they can be used to reduce the uncertainty in making high-level decisions about how to perform control functions that are normally performed by humans.

Have we achieved autonomous control via intelligent control or any other methods? This is a difficult question to answer since certain levels of autonomy have certainly been achieved but there are no rigorous definitions of "degrees of autonomy." For instance, relatively autonomous robots and autonomous vehicles have been implemented. It is clear that current intelligent systems only roughly model their biological counterparts, and hence from one perspective they can achieve relatively little. What will we be able to do if we succeed in emulating the functions of their biological counterparts? Achieve full autonomy via the correct orchestration of intelligent control?

8.6.4 Example: Intelligent Vehicle and Highway Systems

To make the operation of autonomous systems and the notion of autonomy more concrete, let us examine an intelligent vehicle and highway systems (IVHS) problem of automating a highway system. One possible general functional architecture for automated highway systems is shown in Figure 8.11. Here, suppose that we have many vehicles operating on a large roadway system in the metropolitan area of a large city.

Execution Level

Each vehicle is equipped with a (1) *vehicle control system* that can control the brakes, throttle, and steering to automate the driving task (for normal operation or collision avoidance). In addition, suppose that there is a (2) *vehicle information system* in each vehicle that provides information to the driver (e.g., platoon lead vehicle information; vehicle health status; information on traffic congestion, road construction, accidents, weather, road conditions, lodging, and food; etc.) and information to the overall system about the vehicle (e.g., if the vehicle has had an accident or if the vehicle's brakes have failed). For the roadway there are (3)

the *traffic signal controllers* (e.g., for intersections and ramp metering) and (4) the *roadway information systems* that provide information to the driver and other subsystems (e.g., automatic signing systems that provide rerouting information in case of congestion, road condition warning systems, accident information, etc.). It is these four components that form the "execution level" in the intelligent autonomous controller, and clearly these components will be physically distributed across many vehicles, roadways, and areas of the metropolitan area.

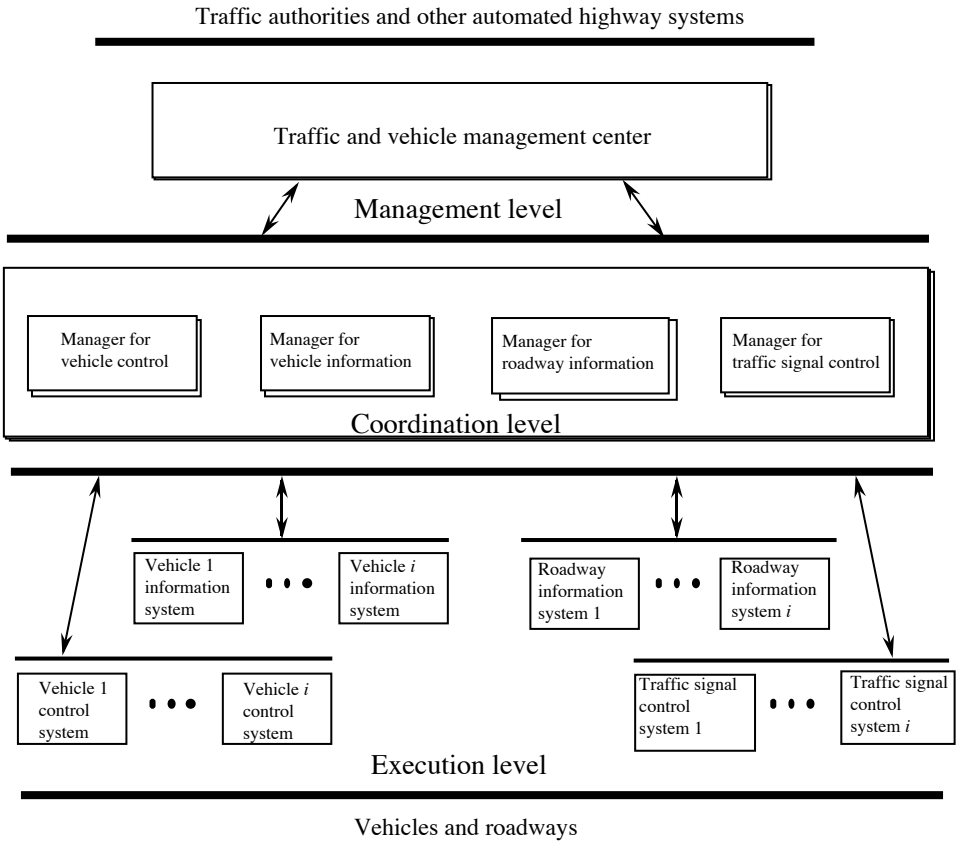


FIGURE 8.11 Intelligent autonomous controller for an intelligent vehicle and highway system (figure taken from [158], © IEEE).

Coordination Level

In the coordination level shown in Figure 8.11, there is a manager for vehicle control that (1) may coordinate the control of vehicles that are in close proximity to form "platoons," maneuver platoons, and avoid collisions; and (2) provide information

about such control activities to the rest of the system. In addition, there is a manager for vehicle information that (1) makes sure that appropriate vehicles get the correct information about road, travel, and traffic conditions; and (2) manages and distributes the information that comes in from vehicles on accidents and vehicle failures (e.g., so that the control manager can navigate platoons to avoid collisions). The manager for traffic signal control could (1) utilize information from the roadway information system (e.g., on accidents or congestion) to adaptively change the traffic light sequences at several connected intersections to reduce congestion; and (2) provide information to the other subsystems about signal control changes (e.g., to the vehicle information systems). The manager for roadway information (1) provides information on road conditions, accidents, and congestion to the other subsystems; and (2) provides information from the other subsystems to the roadway for changeable message signs (e.g., rerouting information from the traffic signal control manager). As indicated in Figure 8.11, there are multiple copies of each of the managers and the entire coordination level as needed for different areas in the metropolitan region.

Management Level

The management level is the traffic and vehicle management center, which provides for high-level management of traffic flow. It provides an interface to other automated highway systems (perhaps in rural areas or other nearby metropolitan areas) and to traffic authorities (e.g., to provide information to police and emergency services on accidents and to input information on construction, weather predictions, and other major events that affect traffic flow). It can interact with traffic authorities to advise them on the best way to avoid congestion given current weather conditions, construction, and expected traffic loads. It can monitor the performance of all the lower-level subsystems in the coordination and execution levels, and suggest corrective actions if there are problems.

Fundamental Characteristics

Notice that in terms of the fundamental characteristics of intelligent autonomous control systems discussed in Section 8.6.2, we find a successive delegation of duties as we go down the hierarchy of the controller in Figure 8.11. For example, high-level tasks at the management level may involve reconfiguring traffic signaling due to construction and weather. The coordination-level manager for roadway information and traffic signal control may develop a new signaling strategy. This strategy would be implemented in the execution level on the changeable message signs (to inform drivers) and the traffic signal control strategy. The higher levels of the hierarchy are often concerned with slower and broader aspects of the system behavior (of course, in an accident situation the traffic and vehicle management center would react as quickly as possible to alert emergency vehicles). The lower levels of the system have a smaller "contextual horizon" since they consider much less information in making decisions. Also, the decision rate tends to be higher at the lower levels (e.g., the rate at which control corrections are made as a vehicle automatically

steers around a curve may be on the order of milliseconds, while the decision rate at the management level may be on the order of minutes or hours).

Clearly, there is the need for a significant amount of interdisciplinary activity to implement such a complex control system that involves a wide range of technologies and falls beyond the traditional scope of control problems. There is no single control technique (conventional or intelligent) that can be used to solve the diversity of problems that is found in a complex automated highway system problem. While conventional systems and control technologies will certainly find wide use in IVHS, it seems likely that intelligent systems and control techniques will prove to be useful for at least some functions, especially considering the focus on automating what has traditionally been largely a human control activity. Similar statements seem to hold for many complex autonomous systems.

8.7 Summary

In this chapter we have provided an overview of the relationships between fuzzy control and intelligent control. Our overall objective is twofold. First, we wish to provide the reader who understands the basics of fuzzy control with a view of the other areas of intelligent control, as this tends to strengthen one's understanding of fuzzy control. Second, we wish to provide the reader who knows some other area of intelligent control with a view of fuzzy control. We provided an overview of a conventional control-engineering perspective on fuzzy control. Next, we highlighted some ideas from expert control, planning systems, neural networks, genetic algorithms, and intelligent autonomous control.

Upon completing this chapter, the reader should understand the following:

- The general ideas in how conventional control compares to fuzzy control.
- The basics of the operation of the multilayer perceptron and the radial basis function neural network.
- The general ideas on how fuzzy systems and neural networks are related and how techniques from each of the fields can be used in the other.
- The basic mechanics of the operation of the genetic algorithm.
- How genetic algorithms can be used for the design of fuzzy estimators or control systems, and some basic ideas on how a genetic algorithm can be used in an adaptive controller to tune a fuzzy or conventional controller.
- The connections to expert control, particularly in how expert systems use more general knowledge representation and inference.
- The basics of how a planning system operates and how a planning system might be used in a fuzzy control system.
- A general definition of intelligent controllers.

- The general hierarchical functional architecture for intelligent autonomous control systems.
- Some of the basic characteristics of intelligent autonomous control systems.
- How to view the autonomous control problem as a general control problem.
- Basic ideas in how to form an intelligent autonomous controller for an IVHS application.

Essentially, this is a checklist of the major topics of this chapter. This chapter serves as the concluding remarks for the entire book and tries to motivate the reader to branch out into the many other interesting areas in intelligent control.

8.8 For Further Study

The section on fuzzy versus conventional control has developed over the years and has been woven throughout many of the papers published by our Ohio State group. Section 8.2 appeared, however, in an earlier form in [157]. Other articles on the relationships between conventional and intelligent control can be found in [160, 156, 159]. There are many books and articles on neural networks [68, 67, 96]. For neural control, consider, for example, the books [142, 234, 64, 26] and the articles [150, 74]. Nice introductions to neural control and learning control are contained in [50, 51]. For more details on genetic algorithms, see the books [58, 139] or article [204]. Computer-aided design of fuzzy controllers via genetic algorithms has been studied in a variety of places including [88, 79, 155, 222, 116]. An example of how to use a genetic algorithm to design direct, adaptive, and supervisory fuzzy controllers for a robot is given in [27]. The genetic model reference adaptive controller was first introduced in [169]. For more details on (direct) expert control, see [163], or [131] for a control-engineering analysis of the feedback loop that is inherent in the expert system inference process. The idea of using expert systems to supervise adaptive control systems was first introduced in [8] and is also reported on in [6]. The section on planning systems is based on [162], where the authors also discuss situation assessment, world modeling, and adaptive planning systems. For an artificial intelligence perspective on planning systems that attempts to relate planning ideas to control theory, see [44]. For a general introduction to intelligent control, see the books [6, 64, 219, 234] or articles [5, 2, 205]. For a particularly easy-to-read introduction that provides a brief overview of many of the areas of intelligent control, see [158] (several parts of this chapter are based on that article, e.g., the IVHS example).

8.9 Exercises

Exercise 8.1 (Conventional Versus Fuzzy Control): In this problem you will compare and contrast conventional versus fuzzy control.

(a) When and how is a mathematical model used in the fuzzy control design methodology?

(b) What are the disadvantages of not using a mathematical model in the control design process?

(c) Why is the verification of the behavior of a fuzzy control system important? When is it unimportant?

(d) Can a fuzzy controller be implemented on a conventional microprocessor?

Exercise 8.2 (Back-Propagation Training of Multilayer Perceptrons): Suppose that you use a three-layer multilayer perceptron and the sigmoidal activation function given in Equation (8.2) on page 445 for all neurons in the network.

(a) Use the gradient approach to training that is developed in Chapter 5, Section 5.4, on page 260, to develop training algorithms for the weights, biases, and b parameters of the activation functions (i.e., their slopes).

(b) Choose $n = n_1 = n_2 = 2$ and $m = 1$, and use the algorithm in (a) to train the network to map the data set G used in Chapter 5, Equation (5.3), on page 236. Test the interpolation capabilities of the network by seeing how it will map an input that is not in the data set G.

Exercise 8.3 (Gradient Training of Radial Basis Function Neural Networks): Consider training the radial basis function shown in Figure 8.3 on page 448.

(a) Use the gradient approach to develop update laws for $\underline{c}_i = [c_1^i, c_2^i, \ldots, c_n^M]^\top$, σ_i, and $\bar{y}_i$, $i = 1, 2, \ldots, M$. Develop the parameter update laws for the following four cases: (1) For the output, choose Equation (8.3) on page 447 to compute its output and use "choice 1" for the receptive field unit; (2) For the output, choose Equation (8.3) on page 447 to compute its output and use "choice 2" for the receptive field unit; (3) For the output, choose Equation (8.4) on page 448 to compute its output and use "choice 1" for the receptive field unit; and (4) For the output, choose Equation (8.4) on page 448 to compute its output and use "choice 2" for the receptive field unit.

(b) Suppose our unknown function is

$$y = 2x^2 = g(x)$$

where x and y are scalars. Suppose that we know that $f(x|\theta) = \theta x^2$, which is not a fuzzy system (i.e., we know that our unknown system squares its input, but we do not know how it scales it). Develop a gradient algorithm that will estimate θ given

$$G = \{(i, g(i)) : i = 5, 6, 7, \ldots, 60\}$$

Cycle through the data once and use only one gradient update step for each data pair. Hint: You will need a small step size.

(c) You will use the RBF training formulas obtained in part (a), case 1, to train an RBF $f(x|\theta)$ where θ holds the parameters of the RBF neural network. Use the training data set G from (b). Use the same basic training approach as in (b) except cycle through the data repeatedly until the error between the output of the RBF neural network and the output portion of the training data is less than 10 for all the training data.

(d) Repeat (c) but for $g(x) = (7/500)x^2$.

Exercise 8.4 (Genetic Algorithms for Optimization): In this problem you will use the genetic algorithm to solve some simple optimization problems. You will need to write a computer program that simulates the genetic algorithm.

(a) Suppose that you are given the function

$$f(x) = x\sin(10\pi x) + 1$$

which is taken from [139]. Design and implement on a computer a genetic algorithm for finding the maximum of this function over the range $x \in [-0.5, 1]$. Plot the best individual, best fitness, and average fitness against the generation. Plot the function to verify the results.

(b) Suppose that you are given the function

$$f(x) = \text{sinc}(x+2) = \frac{\sin(x+2)}{x+2}.$$

Design and implement on a computer a genetic algorithm for finding the maximum of this function over the range $x \in [-10, 10]$. Plot the best individual, best fitness, and average fitness against the generation. Plot the function to verify the results.

(c) Suppose that you are given the function

$$\begin{aligned} z = {} & 0.8x\exp\left(-x^2-(y+1.3)^2\right) + x\exp\left(-x^2-(y-1)^2\right) \\ & + 1.15x\exp\left(-x^2-(y+3.25)^2\right). \end{aligned}$$

Design and implement on a computer a genetic algorithm for finding the maximum of this function over the range $x \in [-5, 2]$, $y \in [-2, 2]$. Plot the best individual, best fitness, and average fitness against the generation. Plot the function to verify the results.

(d) Suppose that you are given the function

$$z = 1.5\text{sinc}(x) + 2\text{sinc}(y) + 3\text{sinc}(x+8) + \text{sinc}(y+8) + 2.$$

Design and implement on a computer a genetic algorithm for finding the maximum of this function over the range $x \in [-12, 12]$, $y \in [-12, 12]$. Plot the best individual, best fitness, and average fitness against the generation. Plot the function to verify the results.

Overall, the objective of this exercise is for you to get a genetic algorithm operating in the computer. From there you can apply the genetic algorithm to the design or tuning of fuzzy systems, as we discussed in the chapter.

Exercise 8.5 (Knowledge-Based Control): In this problem you will study some general issues in knowledge-based control.

(a) Describe an inference strategy that is not used in a standard fuzzy system but may be useful in expert control.

(b) Draw the block diagrams for a planning system–based controller that uses a "situation assessor." Draw the block diagrams for a planning system–based controller that uses "world modeling" in conjunction with a "planner designer." In each case build on the planning system–based controller shown in Figure 8.8 on page 463.

Exercise 8.6 (Defining Intelligent Control): Based on your reading of this book, and particularly Section 8.6 of this chapter, provide a definition for the field of "intelligent control." Your definition should clearly reflect whether you believe that current computers can exhibit intelligent behavior. Moreover, you should pay special attention to how the word "intelligent" is defined.

Exercise 8.7 (Intelligent Autonomous Controller Functional Architectures)*: In this problem you will design a functional hierarchy for an intelligent autonomous controller for two different applications.

(a) Draw the block diagram for the functional architecture for a multilayer hierarchical controller for solving a robot control problem (one where planning, learning, and low-level control is used).

(b) Repeat (a) but for an autonomous land or underwater vehicle problem.

Bibliography

[1] S. Abe and M.-S. Lan. Fuzzy rules extraction directly from numerical data for function approximation. *IEEE Trans. on Systems, Man, and Cybernetics*, 25(1):119–129, January 1995.

[2] J. S. Albus. Outline for a theory of intelligence. *IEEE Trans. on Systems, Man, and Cybernetics*, 21(3):473–509, May/Jun. 1991.

[3] B. D. O. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1990.

[4] A. Angsana and K. M. Passino. Distributed fuzzy control of flexible manufacturing systems. *IEEE Trans. on Control Systems Technology*, 2(4):423–435, December 1994.

[5] P. J. Antsaklis and K. M. Passino. Towards intelligent autonomous control systems: Architecture and fundamental issues. *Journal of Intelligent and Robotic Systems*, 1:315–342, 1989.

[6] P. J. Antsaklis and K. M. Passino, editors. *An Introduction to Intelligent and Autonomous Control*. Kluwer Academic Publishers, Norwell, MA, 1993.

[7] J. Aracil, A. Ollero, and A. Garcia-Cerezo. Stability indices for the global analysis of expert control systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 19(5):998–1007, Sep./Oct. 1989.

[8] K. J. Åström, J. J. Anton, and K. E. Arzen. Expert control. *Automatica*, 22(3):277–286, March 1986.

[9] K. J. Åström and T. Hägglund, editors. *PID Control: Theory, Design, and Tuning*. Instrument Society of America Press, Research Triangle Park, NC, second edition, 1995.

[10] K. J. Åström and B. Wittenmark. *Computer Controlled Systems: Theory and Design*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[11] K. J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, Reading, MA, 1995.

[12] M. Athans and P. L. Falb. *Optimal Control*. McGraw-Hill, New York, 1966.

[13] D. P. Atherton. *Nonlinear Control Engineering: Describing Function Analysis and Design*. Van Nostrand Reinhold, Berkshire, England, 1982.

[14] D. P. Atherton. A describing function approach for the evaluation of fuzzy logic control. In *Proc. of the American Control Conf.*, pages 765–766, San Francisco, CA, June 1993.

[15] W. G. Barie. Design and implementation of a nonlinear state-space controller for a magnetic levitation system. Master's thesis, University of Pittsburgh, 1994.

[16] M. Barrére, A. Jaumotte, B. Veubeke, and J. Vandenkerckhove, editors. *Rocket Propulsion*. Elsevier, New York, 1960.

[17] G. Bartolini, G. Casalino, F. Davoli, R. Minciardi M. Mastretta, and E. Morten. Development of performance adaptive fuzzy controllers with applications to continuous casting plants. In M. Sugeno, editor, *Industrial Applications of Fuzzy Control*, pages 73–86. Elsevier, Amsterdam, The Netherlands, 1985.

[18] C. Batur, A. Srinivasan, and C.-C. Chan. Automated rule-based model generation for uncertain complex dynamic systems. *Engineering Applications in Artificial Intelligence*, 4(5):359–366, 1991.

[19] C. Batur, A. Srinivasan, and C. C. Chan. Fuzzy model based fuzzy predictive control. In *Proc. of the 1st Int. Conf. on Fuzzy Theory and Technology*, pages 176–180, 1992.

[20] M. Bech and L. Smitt. Analogue simulation of ship maneuvers. Technical report, Hydro-og Aerodynmisk Laboratorium, Lyngby, Denmark, 1969.

[21] J. Bernard. Use of a rule-based system for process control. *IEEE Control Systems Magazine*, 8(5):3–13, October 1988.

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

[23] J. C. Bezdek. A convergence theorem for the fuzzy isodata clustering algorithms. In J. C. Bezdek and S. K. Pal, editors, *Fuzzy Models for Pattern Recognition*, pages 130–137. IEEE Press, New York, 1992.

[24] J. C. Bezdek, R. J. Hathaway, M. J. Sabin, and W. T. Tucker. Convergence theory for fuzzy c-means: Counterexamples and repairs. In J. C. Bezdek and S. K. Pal, editors, *Fuzzy Models for Pattern Recognition*, pages 138–142. IEEE Press, New York, 1992.

[25] P. P. Bonissone, V. Badami, K. H. Chiang, P. S. Khedkar, K. W. Marcelle, and M. J. Schutten. Industrial applications of fuzzy logic at General Electric. *Proc. of the IEEE, Special Issue on Fuzzy Logic in Engineering Applications*, 83(3):450–465, March 1995.

[26] M. Brown and C. Harris. *Neurofuzzy Adaptive Modeling and Control*. Prentice-Hall, Englewood Cliffs, NJ, 1994.

[27] S. C. Brown and K. M. Passino. Intelligent control for an acrobot. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 18:209–248, 1997.

[28] J .J. Buckley. Fuzzy controller: Further limit theorems for linear control rules. *Fuzzy Sets and Systems*, 36:225–233, 1990.

[29] J. J. Buckley and H. Ying. Fuzzy controller theory: Limit theorems for linear fuzzy control rules. *Automatica*, 25(3):469–472, March 1989.

[30] S. Chand and S. Hansen. Energy based stability analysis of a fuzzy roll controller design for a flexible aircraft wing. In *Proc. of the IEEE Conf. on Decision and Control*, pages 705–709, Tampa, FL, December 1989.

[31] C. T. Chen. *Linear System Theory and Design*. Saunders College Publishing, Fort Worth, TX, 1984.

[32] Y.-Y. Chen. *The Global Analysis of Fuzzy Dynamical Systems*. PhD thesis, University of California at Berkeley, 1989.

[33] Y. Y. Chen and T. C. Tsao. A description of the dynamical behavior of fuzzy systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 19(4):745–755, Jul./Aug. 1989.

[34] S. Chiu and S. Chand. Fuzzy controller design and stability analysis for an aircraft model. In *Proc. of the American Control Conf.*, pages 821–826, Boston, June 1991.

[35] S. Chiu, S. Chand, D. Moore, and A. Chaudhary. Fuzzy logic for control of roll and moment for a flexible wing aircraft. *IEEE Control Systems Magazine*, 11(4):42–48, June 1991.

[36] E. Czogała and W. Pedrycz. On identification in fuzzy systems and its applications in control problems. *Fuzzy Sets and Systems*, 6:73–83, 1981.

[37] E. Czogała and W. Pedrycz. Control problems in fuzzy systems. *Fuzzy Sets and Systems*, 7:257–273, 1982.

[38] S. Daley and K. F. Gill. A design study of a self-organizing fuzzy logic controller. *Proc. Institute of Mechanical Engineers*, 200(C1):59–69, 1986.

[39] S. Daley and K. F. Gill. Atitude control of a spacecraft using an extended self-organizing fuzzy logic controller. *Proc. Institute of Mechanical Engineers*, 201(C2):97–106, 1987.

[40] S. Daley and K. F. Gill. Comparison of a fuzzy logic controller with a P+D control law. *Trans. ASME, Journal of Dynamical System, Measurement, and Control*, 111:128–137, June 1989.

[41] J. H. D'Azzo and C. H. Houpis. *Linear Control System Analysis and Design: Conventional and Modern*. McGraw-Hill, New York, 1995.

[42] C. W. de Silva. An analytical framework for knowledge-based tuning of servo controllers. *Engineering Applications in Artificial Intelligence*, 4(3):177–189, 1991.

[43] C. W. de Silva. Considerations of hierarchical fuzzy control. In H. T. Nguyen, M. Sugeno, R. Tong, and R. R. Yager, editors, *Theoretical Aspects of Fuzzy Control*, pages 183–234. Wiley, New York, 1995.

[44] T. Dean and M. P. Wellman. *Planning and Control*. Morgan Kaufman, San Mateo, CA, 1991.

[45] R. C. Dorf and R. H. Bishop. *Modern Control Systems*. Addison-Wesley, Reading, MA, 1995.

[46] J. C Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback Control Theory*. Macmillan, New York, 1992.

[47] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An Introduction to Fuzzy Control*. Springer-Verlag, New York, 1993.

[48] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, Orlando, FL, 1980.

[49] S. S. Farinwata and G. Vachtsevanos. Stability analysis of the fuzzy controller designed by the phase portrait assignment algorithm. In *Proc. of the 2nd IEEE Int. Conf. on Fuzzy Systems*, pages 1377–1382, San Francisco, CA, March 1993.

[50] J. Farrell. Neural control. In W. Levine, editor, *The Control Handbook*, pages 1017–1030. CRC Press, Boca Raton, FL, 1996.

[51] J. A. Farrell and W. Baker. Learning control systems. In P. J. Antsaklis and K. M. Passino, editors, *An Introduction to Intelligent and Autonomous Control Systems*, pages 237–262. Kluwer Academic Publishers, Norwell, MA, 1993.

[52] J. Fei and C. Isik. The analysis of fuzzy knowledge-based systems using cell-to-cell mapping. In *Proc. of the IEEE Int. Symp. on Intelligent Control*, pages 633–637, Philadelphia, September 1990.

[53] R. E. Fenton and R. J. Mayhan. Automated highway studies at The Ohio State University: An overview. *IEEE Trans. on Vehicular Technology*, 40(1):100–113, February 1991.

[54] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Wesley, Reading, MA, 1994.

[55] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, Reading, MA, 1990.

[56] B. Friedland. *Control System Design*. McGraw-Hill, New York, 1986.

[57] E. Garcia-Benitez, S. Yurkovich, and K. M. Passino. Rule-based supervisory control of a two-link flexible manipulator. *Journal of Intelligent and Robotic Systems*, 7(2):195–213, 1993.

[58] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[59] G. H. Golub and J. H. Ortega. *Scientific Computing and Differential Equations: An Introduction to Numerical Methods*. Academic Press, Boston, 1992.

[60] G. C. Goodwin and K. S. Sin. *Adaptive Filtering Prediction and Control*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[61] P. Graham and R. Newell. Fuzzy identification and control of a liquid level rig. *Fuzzy Sets and Systems*, 26:255–273, 1988.

[62] P. Graham and R. Newell. Fuzzy adaptive control of a first-order process. *Fuzzy Sets and Systems*, 31:47–65, 1989.

[63] M. J. Grimble and M. A. Johnson. *Optimal Control and Stochastic Estimation*. Wiley, New York, 1988.

[64] M. Gupta and N. Sinha, editors. *Intelligent Control: Theory and Practice*. IEEE Press, Piscataway, NJ, 1995.

[65] C. J. Harris and C. G. Moore. Phase plane analysis tools for a class of fuzzy control systems. In *Proc. of the IEEE Int. Conf. on Fuzzy Systems*, pages 511–518, San Diego, CA, March 1992.

[66] J. K. Harvey. Implementation of fixed and self-tuning controllers for wheel torque regulation. Master's thesis, The Ohio State University, 1993.

[67] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[68] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Reading, MA, 1991.

[69] K. L. Hillsley and S. Yurkovich. Vibration control of a two-link flexible robot arm. *Dynamics and Control*, 3:261–280, 1993.

[70] L. Ho, editor. *Discrete Event Dynamic Systems: Analyzing Complexity and Performance in the Modern World*. IEEE Press, New York, 1992.

[71] C. S. Hsu. A theory of cell-to-cell mapping dynamical systems. *Trans. of the ASME, Journal of Applied Mechanics*, 47:931–939, December 1980.

[72] C. S. Hsu and R. S. Guttalu. An unraveling algorithm for global analysis of dynamical systems: An application of cell-to-cell mapping. *Trans. of the ASME, Journal of Applied Mechanics*, 47:940–948, December 1980.

[73] K. J. Hunt. Induction of decision trees for rule based modelling and control. In *Proc. of the IEEE Int. Symp. on Intelligent Control*, pages 306–311, Glasgow, Scotland, August 1992.

[74] K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop. Neural networks for control systems: A survey. In M. M. Gupta and D. H. Rao, editors, *Neuro-Control Systems: Theory and Applications*, pages 171–200. IEEE Press, Piscataway, NJ, 1994.

[75] J. K. Hurtig, S. Yurkovich, K. M. Passino, and D. Littlejohn. Torque regulation with the General Motors ABS VI electric brake system. In *Proc. of the American Control Conf.*, pages 1210–1211, Baltimore, MD, June 1994.

[76] H. Ichihashi. Efficient algorithms for acquiring fuzzy rules from examples. In H. T. Nguyen, M. Sugeno, R. Tong, and R. R. Yager, editors, *Theoretical Aspects of Fuzzy Control*, pages 261–280. Wiley, New York, 1995.

[77] P. A. Ioannou and J. Sun. *Robust Adaptive Control*. Prentice-Hall, Englewood Cliffs, NJ, 1996.

[78] S. Isaka, A. Sebald, A. Karimi, N. Smith, and M. Quinn. On the design and performance evaluation of adaptive fuzzy controllers. In *Proc. of the IEEE Conf. on Decision and Control*, pages 1068–1069, Austin, TX, December 1988.

[79] H. Ishibuchi, K. Nozaki, and N. Yamamoto. Selecting fuzzy rules by genetic algorithm for classification problems. In *Proc. of the 2nd IEEE Int. Conf. on Fuzzy Systems*, pages 1119–1124, San Francisco, CA., March 1993.

[80] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, New York, third edition, 1995.

[81] J.-S. R. Jang. ANFIS: Adaptive-Network-Based Fuzzy Inference System.

IEEE Trans. on Systems, Man, and Cybernetics, 23(3):665–685, May/Jun. 1993.

[82] J.-S. R. Jang and C.-T. Sun. Neuro-fuzzy modeling and control. *Proc. of the IEEE, Special Issue on Fuzzy Logic in Engineering Applications*, 83(3):378–406, March 1995.

[83] D. L. Jenkins and K. M. Passino. Introduction to nonlinear analysis of fuzzy control systems. *Unpublished paper*, 1996.

[84] T. A. Johansen. Stability, robustness, and performance of fuzzy model based control. In *Proc. of the IEEE Conf. on Decision and Control*, pages 604–609, Kobe, Japan, December 1996.

[85] Tor A. Johansen. Fuzzy model based control: Stability, robustness, and performance issues. *IEEE Trans. on Fuzzy Systems*, 2(3):221–234, August 1994.

[86] A. Juditsky, H. Hjalmarsson, A. Benveniste, B. Deylon, L. Ljung, J. Sjoberg, and Q. Zhang. Nonlinear black-box modeling in system identification: Mathematical foundations. *Automatica*, 31(12):1691–1724, December 1995.

[87] A. Kandel and G. Langholz, editors. *Fuzzy Control Systems*. CRC Press, Boca Raton, FL, 1993.

[88] C. Karr and E. Gentry. Fuzzy control of pH using genetic algorithms. *IEEE Trans. on Fuzzy Systems*, 1(1):46–53, 1993.

[89] J. M. Keller, M. R. Gray, and J. A. Givens. A fuzzy k-nearest neighbor algorithm. *IEEE Trans. on Systems, Man, and Cybernetics*, 15(4):580–585, July 1985.

[90] H. K. Khalil. *Nonlinear Systems*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 1996.

[91] W. Kickert and H. Van Nauta Lemke. Application of a fuzzy controller in a warm water plant. *Automatica*, 12(4):301–308, April 1976.

[92] W. J. M. Kickert and E. H. Mamdani. Analysis of a fuzzy logic controller. *Fuzzy Sets and Systems*, 1:29–44, 1978.

[93] J. B. Kiszka, M. M. Gupta, and P. N. Nikiforuk. Energetistic stability of fuzzy dynamic systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 15(6):783–792, Nov./Dec. 1985.

[94] G. J. Klir and T. A. Folger. *Fuzzy Sets, Uncertainty and Information*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[95] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[96] B. Kosko. *Neural Networks and Fuzzy Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1992.

[97] V. Krishnaswami, G. C. Luh, and G. Rizzoni. Fault detection in IC engines using nonlinear parity equations. In *Proc. of the American Control Conf.*, pages 2001–2005, Baltimore, MD, 1994.

[98] V. Krishnaswami and G. Rizzoni. Nonlinear parity equation residual generation for fault detection and isolation. In *Proc. of the IFAC/IMACS Symp. on Fault Detection, Supervision and Saftey for Technical Processes–SAFEPROCESS 1994*, pages 317–322, Espoo, Finland, 1994.

[99] M. Krstic, I. Kanellakopoulos, and P. Kokotovic. *Nonlinear and Adpative Control Design*. Wiley, New York, 1995.

[100] P. R. Kumar and T. I. Seidman. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Trans. Automatic Control*, 35(3):289–298, March 1990.

[101] P. R. Kumar and P. P. Varaiya. *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[102] B. C. Kuo. *Automatic Control Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1991.

[103] W. A. Kwong and K. M. Passino. Dynamically focused fuzzy learning control. *IEEE Trans. on Systems, Man, and Cybernetics*, 26(1):53–74, February 1996.

[104] W. A. Kwong, K. M. Passino, E. G. Laukonen, and S. Yurkovich. Expert supervision of fuzzy learning systems for fault tolerant aircraft control. *Proc. of the IEEE, Special Issue on Fuzzy Logic in Engineering Applications*, 83(3):466–483, March 1995.

[105] G. Langari. *A Framework for Analysis and Synthesis of Fuzzy Linguistic Control Systems*. PhD thesis, University of California at Berkeley, 1990.

[106] G. Langari and M. Tomizuka. Stability of fuzzy linguistic control systems. In *Proc. of the IEEE Conf. on Decision and Control*, pages 2185–2190, Honolulu, HI, December 1990.

[107] E. Laukonen and K. M. Passino. Tools for computer simulation of fuzzy systems. Control Research Laboratory Report CRL-1081-SP93-R, Department of Electrical Engineering, The Ohio State University, July 1993.

[108] E. G. Laukonen and K. M. Passino. Training fuzzy systems to perform estimation and identification. *Engineering Applications of Artificial Intelligence*, 8(5):499–514, 1995.

[109] E. G. Laukonen, K. M. Passino, V. Krishnaswami, G.-C. Luh, and G. Rizzoni. Fault detection and isolation for an experimental internal combustion engine via fuzzy identification. *IEEE Trans. on Control Systems Technology*, 3(3):347–355, September 1995.

[110] J. Layne. Fuzzy model reference learning control. Master's thesis, Department of Electrical Engineering, The Ohio State University, 1992.

[111] J. R. Layne and K. M. Passino. Fuzzy model reference learning control. In *IEEE Conf. on Control Applications*, pages 686–691, Dayton, OH, September 1992.

[112] J. R. Layne and K. M. Passino. Fuzzy model reference learning control for cargo ship steering. *IEEE Control Systems Magazine*, 13(6):23–34, December 1993.

[113] J. R. Layne and K. M. Passino. Fuzzy model reference learning control. *Journal of Intelligent and Fuzzy Systems*, 4(1):33–47, 1996.

[114] J. R. Layne, K. M. Passino, and S. Yurkovich. Fuzzy learning control for anti-skid braking systems. *IEEE Trans. on Control Systems Technology*, 1(2):122–129, June 1993.

[115] C. Lee. Fuzzy logic in control systems: Fuzzy logic controller–Parts I–II. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(2):404–435, Mar./Apr. 1990.

[116] M. A. Lee and H. Takagi. Integrating design stages of fuzzy systems using genetic algorithms. In *Proc. of the 2nd IEEE Int. Conf. on Fuzzy Systems*, pages 612–617, San Francisco, CA, March 1993.

[117] Y.-C. Lee, C. Hwang, and Y.-P. Shih. A combined approach to fuzzy model identification. *IEEE Trans. on Systems, Man, and Cybernetics*, 24(5):736–744, May 1994.

[118] W. K. Lennon and K. M. Passino. Intelligent control for brake systems. In *Proc. IEEE Int. Symp. on Intelligent Control*, pages 499–504, Monterey, CA, August 1995.

[119] I. J. Leontaritis and S. A. Billings. Input-output parametric models for non-linear systems, Part I: Deterministic nonlinear systems. *International Journal of Control*, 41(2):303–328, 1985.

[120] I. J. Leontaritis and S. A. Billings. Input-output parametric models for non-linear systems, Part II: Stochastic nonlinear systems. *International Journal of Control*, 41(2):329–344, 1985.

[121] W. Levine, editor. *The Control Handbook*. CRC Press, Boca Raton, FL, 1996.

[122] F. L. Lewis. *Optimal Control*. Wiley, New York, 1986.

[123] F. L. Lewis. *Optimal Estimation*. Wiley, New York, 1986.

[124] F. L. Lewis and K. Liu. Towards a paradigm for fuzzy logic control. *Automatica*, 32(2):167–181, February 1995.

[125] Y. Li and C. Lau. Development of fuzzy algorithms for servo systems. *IEEE Control Systems Magazine*, 9(2):65–72, April 1989.

[126] C. T. Lin and C. S. G. Lee. Neural network-based fuzzy logic control and decision system. *IEEE Trans. on Computers*, 40(12):1320–1336, December 1991.

[127] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[128] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, Reading, MA, 1984.

[129] G.-C. Luh. *Multi-Input Multi-Output Modelling of Nonlinear Systems with Application to Internal Combustion Engine Modelling*. PhD thesis, The Ohio State University, 1994.

[130] G.-C. Luh and G. Rizzoni. Identification of a nonlinear MIMO IC engine model during IM/240 driving cycle for on-board diagnosis. In *Proc. of the American Control Conf.*, pages 1581–1584, Baltimore, MD, June 1994.

[131] A. D. Lunardhi and K. M. Passino. Verification of qualitative properties of rule-based expert systems. *Int. Journal of Applied Artificial Intelligence*, 9(6):587–621, Nov./Dec. 1995.

[132] J. M. Maciejowski. *Multivariable Feedback Design*. Addison-Wesley, Reading, MA, 1989.

[133] A. T. Magan. Fuzzy parameter estimation for failure identification. Master's thesis, The Ohio State University, 1992.

[134] E. Mamdani. Advances in the linguistic synthesis of fuzzy controllers. *Int. Journal of Man-Machine Studies*, 8(6):669–678, 1976.

[135] E. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[136] G. Mandell, G. Caporaso, and W. Bengen, editors. *Topics in Advanced Model Rocketry*. The MIT Press, Cambridge, MA, 1973.

[137] R. J. Marks, editor. *Fuzzy Logic Technology and Applications*. IEEE Press, New York, 1994.

[138] J. M. Mendel. Fuzzy logic systems for engineering: A tutorial. *Proc. of the IEEE, Special Issue on Fuzzy Logic in Engineering Applications*, 83(3):345–377, March 1995.

[139] Z. Michalewicz. *Genetic Algorithms + Data Structure = Evolution Programs*. Springer-Verlag, Berlin, 1992.

[140] A. N. Michel and R. K. Miller. *Qualitative Analysis of Large Scale Dynamical Systems*. Academic Press, New York, 1977.

[141] R. K. Miller and A. N. Michel. *Ordinary Differential Equations*. Academic Press, New York, 1982.

[142] W. T. Miller, R. S. Sutton, and P. J. Werbos, editors. *Neural Networks for Control*. The MIT Press, Cambridge, MA, 1991.

[143] S. R. Morris. Fuzzy estimation for failure detection and identification. Master's thesis, The Ohio State University, 1994.

[144] V. G. Moudgal, W. A. Kwong, K. M. Passino, and S. Yurkovich. Fuzzy learning control for a flexible-link robot. *IEEE Trans. on Fuzzy Systems*, 3(2):199–210, May 1995.

[145] V. G. Moudgal, K. M. Passino, and S. Yurkovich. Rule-based control for a flexible-link robot. *IEEE Trans. on Control Systems Technology*, 2(4):392–405, December 1994.

[146] G. C. Mouzouris and J. M. Mendel. Nonsingleton fuzzy logic systems: Theory and application. *IEEE Trans. on Fuzzy Systems*, 5(1):56–71, February 1997.

[147] T. F. Murphy, S. Yurkovich, and S.-C. Chen. Intelligent control for paper machine moisture control. In *Proc. of the IEEE Conf. on Control Applications*, pages 826–833, Dearborn, MI, September 1996.

[148] K. Narendra and J. Taylor. *Frequency Domain Criteria for Absolute Stability*. Academic Press, New York, 1973.

[149] K. S. Narendra and A. M. Annaswamy. *Stable Adaptive Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[150] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, 1(1):4–27, 1990.

[151] H. T Nguyen, M. Sugeno, R. Tong, and R. R. Yager, editors. *Theoretical Aspects of Fuzzy Control*. Wiley, New York, 1995.

[152] A. Ollero and A. J. Garcia-Cerezo. Direct digital control, auto-tuning and supervision using fuzzy logic. *Fuzzy Sets and Systems*, 12:135–153, 1989.

[153] R. Palm. Sliding mode fuzzy control. In *Proc. of the IEEE Int. Conf. on Fuzzy Systems*, pages 519–526, San Diego, CA, March 1992.

[154] R. Palm, D. Driankov, and H. Hellendoorn. *Model Based Fuzzy Control.* Springer-Verlag, New York, 1997.

[155] D. Park, A. Kandel, and G. Langholz. Genetic-based new fuzzy reasoning models with application to fuzzy control. *IEEE Trans. on Systems, Man and Cybernetics*, 24(1):39–47, 1994.

[156] K. M. Passino. Bridging the gap between conventional and intelligent control. *IEEE Control Systems Magazine, Special Issue on Intelligent Control*, 13(3):12–18, June 1993.

[157] K. M. Passino. Fuzzy vs. conventional control. In *Proc. of the IEEE Int. Symp. on Intelligent Control*, pages 511–512, Chicago, August 1993.

[158] K. M. Passino. Intelligent control for autonomous systems. *IEEE Spectrum*, 32(6):55–62, June 1995.

[159] K. M. Passino. Intelligent control. In W. Levine, editor, *The Control Handbook*, pages 994–1101. CRC Press, Boca Raton, FL, 1996.

[160] K. M. Passino. Towards bridging the perceived gap between conventional and intelligent control. In M. Gupta and N. Sinha, editors, *Intelligent Control: Theory and Practice*, pages 1–27. IEEE Press, Piscataway, NJ, 1996.

[161] K. M. Passino and P. J. Antsaklis. Fault detection and identification in an intelligent restructurable controller. *Journal of Intelligent and Robotic Systems*, 1(1):145–161, June 1988.

[162] K. M. Passino and P. J. Antsaklis. A system and control theoretic perspective on artificial intelligence planning systems. *Int. Journal of Applied Artificial Intelligence*, 3:1–32, 1989.

[163] K. M. Passino and A. D. Lunardhi. Qualitative analysis of expert control systems. In M. Gupta and N. Sinha, editors, *Intelligent Control: Theory and Practice*, pages 404–442. IEEE Press, Piscataway, NJ, 1996.

[164] K. M. Passino, M. Sartori, and P. J. Antsaklis. Neural computing for numeric to symbolic conversion in control systems. *IEEE Control Systems Magazine*, 9(3):44–52, April 1989.

[165] K. M. Passino and S. Yurkovich. Fuzzy control. In W. Levine, editor, *The Control Handbook*, pages 1001–1017. CRC Press, Boca Raton, FL, 1996.

[166] R. Patton, P. Frank, and R. Clark. *Fault Diagnosis in Dynamic Systems: Theory and Applications.* Prentice-Hall, New York, 1989.

[167] W. Pedrycz. *Fuzzy Control and Fuzzy Systems*. Wiley, New York, second edition, 1993.

[168] J. R. Perkins and P. R. Kumar. Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems. *IEEE Trans. on Automatic Control*, 34(2):139–148, February 1989.

[169] L. Porter and K. M. Passino. Genetic model reference adaptive control. In *Proc. of the IEEE Int. Symp. on Intelligent Control*, pages 219–224, Chicago, August 1994.

[170] T. Procyk and E. Mamdani. A linguistic self-organizing process controller. *Automatica*, 15(1):15–30, January 1979.

[171] K. S. Ray, A. M. Ghosh, and D. D. Majumder. L_2-stability and the related design concept for SISO linear systems associated with fuzzy logic controllers. *IEEE Trans. on Systems, Man, and Cybernetics*, 14(6):932–939, Nov./Dec. 1984.

[172] K. S. Ray and D. D. Majumder. Application of circle criteria for stability analysis of linear SISO and MIMO systems associated with fuzzy logic controllers. *IEEE Trans. on Systems, Man, and Cybernetics*, 14(2):345–349, Mar./Apr. 1984.

[173] F. Van Der Rhee, H. Van Nauta Lemke, and J. Dijkman. Knowledge based fuzzy control of systems. *IEEE Trans. on Automatic Control*, 35(2):148–155, February 1990.

[174] G. Rizzoni and P. S. Min. Detection of sensor failures in automotive engines. *IEEE Trans. on Vehicular Technology*, 40(2):487–500, May 1991.

[175] T. Ross. *Fuzzy Logic in Engineering Applications*. McGraw-Hill, New York, 1995.

[176] W. Rugh. Analytical framework for gain scheduling. *IEEE Control Systems Magazine*, 11(1):79–84, January 1991.

[177] M. J. Sabin. Convergence and consistency of fuzzy c-means/isodata algorithms. In J. C. Bezdek and S. K. Pal, editors, *Fuzzy Models for Pattern Recognition*, pages 143–150. IEEE Press, New York, 1992.

[178] I. W. Sandberg and K. K. Johnson. Steady state errors in nonlinear control systems. *IEEE Trans. on Automatic Control*, 37(12):1985–1989, December 1992.

[179] G. N. Saridis. Analytical formulation of the principle of increasing precision with decreasing intelligence for intelligent machines. *Automatica*, 25(3):461–467, March 1989.

[180] S. Sastry and M. Bodson. *Adaptive Control: Stability, Convergence, and Robustness*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[181] E. Scharf and N. Mandic. The application of a fuzzy controller to the control of a multi-degree-of-freedom robot arm. In M. Sugeno, editor, *Industrial Applications of Fuzzy Control*, pages 41–62. Elsevier, Amsterdam, The Netherlands, 1985.

[182] J. S. Shamma and M. Athans. Analysis of nonlinear gain-scheduled control systems. *IEEE Trans. on Automatic Control*, 35(8):898–907, August 1990.

[183] J. S. Shamma and M. Athans. Gain scheduling: Potential hazards and possible remedies. *IEEE Control Systems Magazine*, 12(2):101–107, April 1992.

[184] C. Y. Shieh and S. S. Nair. A new self tuning fuzzy controller design and experiements. In *Proc. of the 2nd IEEE Int. Conf. on Fuzzy Systems*, pages 309–314, San Francisco, CA, March 1993.

[185] S. E. Shladover. Longitudinal control of automotive vehicles in close-formation platoons. *Trans. ASME, Journal of Dynamic Systems, Measurement and Control*, 113:231–241, June 1991.

[186] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, W.-B. Zhang, D. H. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown. Automatic vehicle control developments in the PATH program. *IEEE Trans. on Vehicular Technology*, 40(1):114–130, February 1991.

[187] S. K. Sin and R. J. P. Defigueiredo. Fuzzy system design through fuzzy clustering and optimal predefuzzification. In *Proc. of the 2nd IEEE Conf. on Fuzzy Systems*, pages 190–195, San Francisco, CA, March 1993.

[188] J. Sjoberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: A unified overview. *Automatica*, 31(12):1725–1750, December 1995.

[189] J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall, Englewood Cliffs, NJ, 1991.

[190] M. Spong. Swing up control of the acrobot. In *Proc. of the IEEE Conf. on Robotics and Automation*, pages 2356–2361, San Diego, CA, May 1994.

[191] M. Spong. The swing up control problem for the acrobot. *IEEE Control Systems Magazine*, 15(1):49–55, February 1995.

[192] M. Spong and M. Vidyasagar. *Robot Dynamics and Control*. Wiley, New York, 1989.

[193] M. W. Spong, F. L. Lewis, and C. T. Abdallah, editors. *Robot Control*. IEEE Press, New York, 1992.

[194] J. T. Spooner, R. Ordonez, and K. M. Passino. Stable direct adaptive control of a class of discrete time nonlinear systems. In *Proc. of the IFAC World Congress, Vol. K*, pages 343–348, San Francisco, CA, July 1996.

[195] J. T. Spooner and K. M. Passino. Stable adaptive control using fuzzy systems and neural networks. IVHS-OSU Report 95-01, The Ohio State University, February 1995.

[196] J. T. Spooner and K. M. Passino. Stable adaptive fuzzy control for an automated highway system. In *Proc. of the IEEE Int. Symp. on Intelligent Control*, pages 531–536, Monterey, CA, August 1995.

[197] J. T. Spooner and K. M. Passino. Stable direct adaptive control using fuzzy systems and neural networks. In *Proc. of the IEEE Conf. on Decision and Control*, pages 249–254, New Orleans, LA, December 1995.

[198] J. T. Spooner and K. M. Passino. Stable indirect adaptive control using fuzzy systems and neural networks. In *Proc. of the IEEE Conf. on Decision and Control*, pages 243–248, New Orleans, LA, December 1995.

[199] J. T. Spooner and K. M. Passino. Adaptive control of a class of decentralized nonlinear systems. *IEEE Trans. on Automatic Control*, 41(2):280–284, February 1996.

[200] J. T. Spooner and K. M. Passino. Stable adaptive control using fuzzy systems and neural networks. *IEEE Trans. on Fuzzy Systems*, 4(3):339–359, August 1996.

[201] Jeffrey T. Spooner, Raul Ordonez, and Kevin M. Passino. Direct adaptive fuzzy control for a class of discrete time systems. In *Proceedings of the American Control Conf.*, pages 1814–1818, Albuquerque, NM, June 4-6 1997.

[202] Jeffrey T. Spooner, Raul Ordonez, and Kevin M. Passino. Indirect adaptive fuzzy control for a class of discrete time systems. In *Proceedings of the American Control Conf.*, pages 3311–3315, Albuquerque, NM, June 4-6 1997.

[203] Jeffrey T. Spooner and Kevin M. Passino. Adaptive prediction using fuzzy systems and neural networks. In *Proceedings of the American Control Conf.*, pages 1266–1270, Albuquerque, NM, June 4-6 1997.

[204] M. Srinivas and L. M. Patnaik. Genetic algorithms: A survey. *IEEE Computer Magazine*, pages 17–26, June 1994.

[205] R. F. Stengel. Toward intelligent flight control. *IEEE Trans. on Systems, Man, and Cybernetics*, 23(6):1699–1717, Nov./Dec. 1993.

[206] M. Sugeno, editor. *Industrial Applications of Fuzzy Control*. Elsevier, Amsterdam, The Netherlands, 1985.

[207] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. on Systems, Man, and Cybernetics*, 15(1):116–132, January 1985.

[208] H. Takahashi. Automatic speed control device using self-tuning fuzzy logic. In *Proc. of the IEEE Workshop on Automotive Applications of Electronics*, pages 65–71, Dearborn, MI, October 1988.

[209] K. Tanaka. Design of model-based fuzzy controller using Lyapunov's stability approach and its application to trajectory stabilization of a model car. In H. T. Nguyen, M. Sugeno, R. Tong, and R. R. Yager, editors, *Theoretical Aspects of Fuzzy Control*, pages 31–50. Wiley, New York, 1995.

[210] K. Tanaka, T. Ikeda, and H. O. Wang. Robust stabilization of a class of uncertain nonlinear systems via fuzzy control: Quadratic stabilizability, h^∞ control theory, and linear matrix inequalities. *IEEE Trans. on Fuzzy Systems*, 4(1):1–13, 1996.

[211] K. Tanaka and M. Sano. Concept of stability margin for fuzzy systems and design of robust fuzzy controllers. In *Proc. of the IEEE Int. Conf. on Fuzzy Systems*, pages 29–34, San Francisco, CA, March 1993.

[212] K. Tanaka and M. Sano. A robust stabilization problem of fuzzy control systems and its application to backing up control of a truck-trailer. *IEEE Trans. on Fuzzy Systems*, 2(2):119–134, 1994.

[213] K. Tanaka and M. Sugeno. Stability analysis and design of fuzzy control systems. *Fuzzy Sets and Systems*, 45:135–156, 1992.

[214] R. Tanscheit and E. Scharf. Experiments with the use of a rule-based self-organising controller for robotics applications. *Fuzzy Sets and Systems*, 26:195–214, 1988.

[215] S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, England, 1992.

[216] S. Tzafestas and N. Papanikolopoulos. Incremental fuzzy expert PID control. *IEEE Trans. on Industrial Electronics*, 37(5):365–371, October 1990.

[217] V. I. Utkin. *Sliding Modes in Control Optimization*. Springer-Verlag, Berlin, 1992.

[218] G. Vachtsevanos, S. S. Farinwata, and D. K. Pirovolou. Fuzzy logic control of an automotive engine. *IEEE Control Systems Magazine*, 13(3):62–68, June 1993.

[219] K. Valavanis and G. Saridis. *Intelligent Robotic Systems: Theory, Design, and Applications*. Kluwer Academic Publishers, Norwell, MA, 1992.

[220] J. Van-Amerongen and A. J. Udink ten Cate. Model reference adaptive autopilots for ships. *Automatica*, 11:441–449, 1975.

[221] H. R. van Nauta Lemke and D.-Z. Wang. Fuzzy PID supervisor. In *Proc. of the IEEE Conf. on Decision and Control*, pages 602–608, Fort Lauderdale, FL, December 1985.

[222] A. Varšek, T. Ubančič, and B. Filipič. Genetic algorithms in controller design and tuning. *IEEE Trans. on Systems, Man and Cybernetics*, 23(5):1330–1339, Sept./Oct. 1993.

[223] M. Vidyasagar. *Nonlinear Systems Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[224] B. H. Wang and G. Vachtsevanos. Learning fuzzy logic control: An indirect control approach. In *Proc. 1st IEEE Int. Conf. on Fuzzy Systems*, pages 297–304, San Diego, CA, March 1992.

[225] H. O. Wang and K. Tanaka. An LMI-based stable fuzzy control of nonlinear systems and its application to the control of chaos. In *Proc. of the IEEE Int. Conf. on Fuzzy Systems*, pages 1433–1439, New Orleans, LA, September 1996.

[226] H. O. Wang, K. Tanaka, and M. F. Griffin. An approach to fuzzy control of nonlinear systems: Stability and design issues. *IEEE Trans. on Fuzzy Systems*, 4(1):14–23, 1996.

[227] L.-X. Wang. Fuzzy systems are universal approximators. In *Proc. of the 1st IEEE Conf. on Fuzzy Systems*, pages 1163–1170, San Deigo, CA, March 1992.

[228] L.-X. Wang. Training of fuzzy logic systems using nearest neighborhood clustering. In *Proc. of the 2nd IEEE Int. Conf. on Fuzzy Systems*, pages 13–17, San Francisco, CA, March 1993.

[229] L.-X. Wang. *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1994.

[230] L.-X. Wang. *A Course in Fuzzy Systems and Control*. Prentice-Hall, Englewood Cliffs, NJ, 1997.

[231] L.-X. Wang and J. M. Mendel. Back-propagation fuzzy system as nonlinear dynamic system identifiers. In *Proc. of the 1st IEEE Int. Conf. on Fuzzy Systems*, pages 1409–1418, San Diego, CA, March 1992.

[232] L.-X. Wang and J. M. Mendel. Fuzzy basis functions, universal approximation

and orthogonal least-squares learning. *IEEE Trans. on Neural Networks*, 3(5):1–8, September 1992.

[233] L.-X. Wang and J. M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Trans. on Systems, Man, and Cybernetics*, 22(6):1414–1427, November 1992.

[234] D. White and D. Sofge, editors. *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, New York, 1992.

[235] M. Widjaja and S. Yurkovich. Intelligent control for swing up and balancing of an inverted pendulum system. In *Proc. of the IEEE Conf. on Control Applications*, pages 534–542, Albany, NY, September 1995.

[236] M. P. Windham. Geometrical fuzzy clustering algorithms. In J. C. Bezdek and S. K. Pal, editors, *Fuzzy Models for Pattern Recognition*, pages 123–129. IEEE Press, New York, 1992.

[237] Q. M. Wu and C. W. de Silva. Model identification for fuzzy dynamic systems. In *Proc. of the American Control Conf.*, pages 2246–2247, San Francisco, CA, June 1993.

[238] R. R. Yager and L. A. Zadeh. *An Introduction to Fuzzy Logic Applications in Intelligent Systems*. Kluwer Academic Publishers, New York, 1992.

[239] T. Yamazaki. *An Improved Algorithm for a Self-Organizing Controller and Its Experimental Analysis*. PhD thesis, London University, 1982.

[240] J. Yen, R. Langari, and L. Zadeh, editors. *Industrial Applications of Fuzzy Logic and Intelligent Systems*. IEEE Press, New York, 1995.

[241] H. Ying, W. Siler, and J. J. Buckley. Fuzzy control theory: A nonlinear case. *Automatica*, 26(3):513–520, March 1990.

[242] S. Yurkovich, A. Tzes, and K.Hillsley. Controlling coupled flexible links rotating in the horizontal plane. In *Proc. of the American Control Conf.*, pages 362–366, San Diego, CA, May 1990.

[243] S. Yurkovich, A. Tzes, I. Lee, and K. Hillsley. Control and system identification of a two-link flexible manipulator. In *Proc. of the IEEE Conf. on Robotics and Automation*, pages 1626–1631, Cincinnati, OH, May 1990.

[244] S. Yurkovich and M. Widjaja. Fuzzy controller synthesis for an inverted pendulum system. *IFAC Control Engineering Practice*, 4(4):455–469, 1996.

[245] L. A. Zadeh. Fuzzy sets. *Informat. Control*, 8:338–353, 1965.

[246] L. A. Zadeh. Outline of a new approach to the analysis of complex systems

and decision processes. *IEEE Trans. on Systems, Man, and Cybernetics*, 3(1):28–44, 1973.

[247] J. Zhao, R. Gorez, and V. Wertz. Synthesis of fuzzy control systems based on linear Takagi-Sugeno fuzzy models. In R. Murray-Smith and T. A. Johansen, editors, *Multiple Model Approaches to Nonlinear Modeling and Control*, pages 311–340. Taylor and Francis, UK, 1996.

[248] J. Zhao, R. Gorez, and V. Wertz. Synthesis of fuzzy control systems with desired performances. In *Proc. of the IEEE Int. Symp. on Intelligent Control*, pages 115–120, Dearborn, MI, September 1996.

[249] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice-Hall, Englewood Cliffs, NJ, 1996.

[250] H. J. Zimmerman. *Fuzzy Set Theory–and Its Applications*. Kluwer Academic Press, Boston, 2nd edition, 1991.

[251] J. Zumberge and K. M. Passino. A case study in intelligent control for a process control experiment. In *Proc. IEEE Int. Symp. on Intelligent Control*, pages 37–42, Dearborn, MI, September 1996.

Index