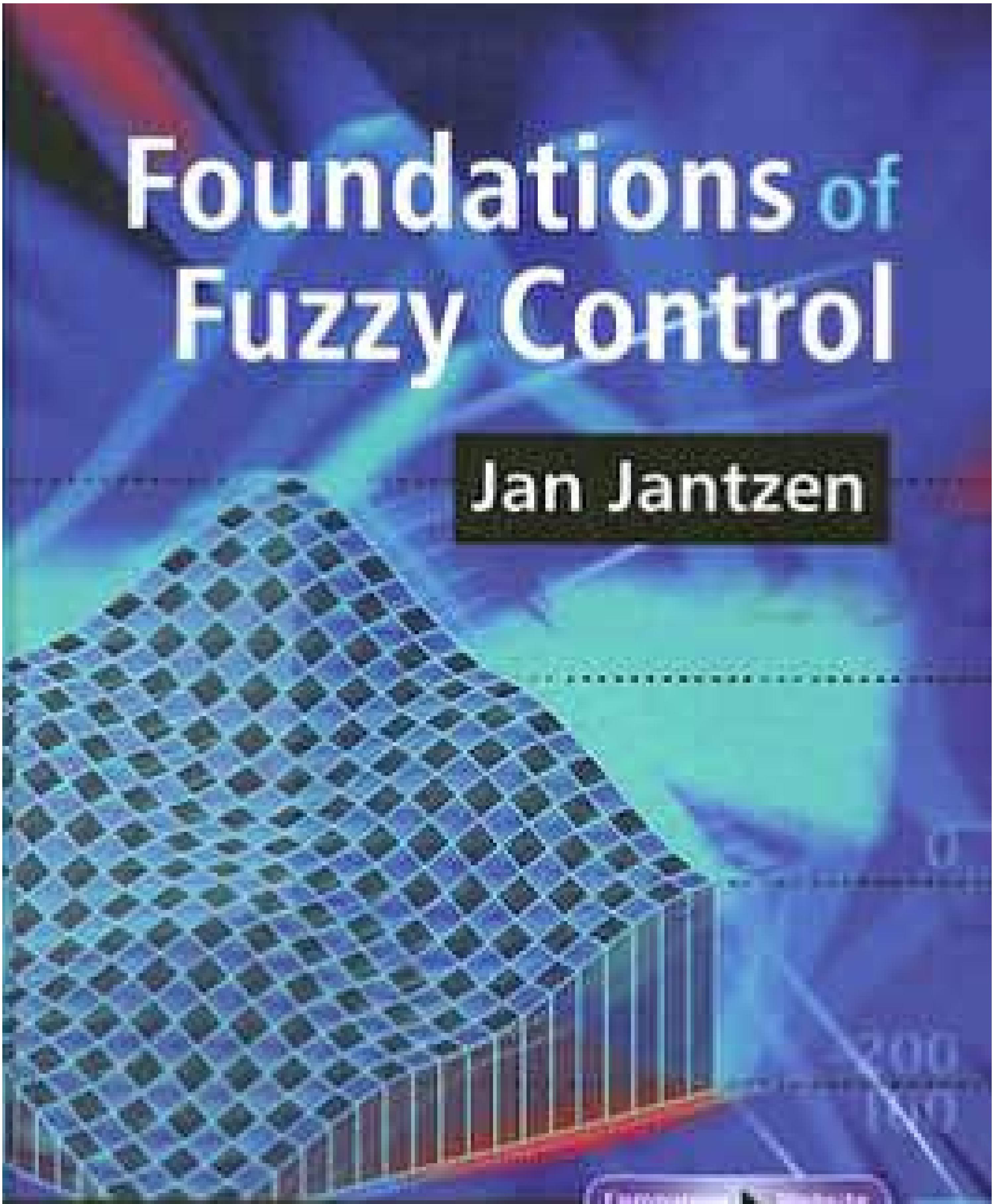
Foundations of Fuzzy Control

Jan Jantzen



WILEY

Foundations of Fuzzy Control

Foundations of Fuzzy Control Jan Jantzen
© 2007 John Wiley & Sons, Ltd. ISBN: 0-470-02963-3

Foundations of Fuzzy Control

Jan Jantzen
Technical University of Denmark

BICENTENNIAL 1807 WILEY 2007 BICENTENNIAL

Library of Congress Cataloging-in-Publication Data

Jantzen, Jan.
Foundations of fuzzy control / Jan Jantzen.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-02963-3 (cloth : alk. paper)
1. Automatic control. 2. Fuzzy systems. 3. Fuzzy automata. I. Title.
TJ213.J32145 2007
629.8–dc22

2006102035

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 978-0-470-02963-3 (HB)

Typeset in 10/12pt Times by Laserwords Private Limited, Chennai, India
Printed and bound in Great Britain by Antony Rowe Ltd, Chippenham, Wiltshire
This book is printed on acid-free paper responsibly manufactured from sustainable forestry
in which at least two trees are planted for each one used for paper production.

In memory of Lauritz Peter Holmblad (23 August 1944–30 March 2005)

Contents

*Sections marked by an asterisk (*) may be skipped in a first reading (please refer to note in Preface).

Foreword

Since the objective of Foundations of Fuzzy Control is to explain why fuzzy controllers behave the way they do, I would like to contribute a historical perspective.

Before the 1960s, a cement kiln operator controlled a cement kiln by looking into its hot end, the burning zone, and watching the smoke leaving the chimney. The operator used a blue glass to protect his eyes. He controlled the fuel/air ratio in order to achieve steady operation of the kiln.

Central control was introduced in the cement industry in the 1960s. PID controllers were installed, mainly for uniform feeding of the raw materials and the fuel. Computers for process supervision and control were introduced in the cement industry in the late 1960s.

During experimental work in the 1970s, the fuel control strategy was programmed as a two-dimensional decision table with an error signal and the change in error as inputs.

The first time we heard about fuzzy logic was at the fourth IFAC/IFIP International Conference on Digital Computer Applications to Process Control, held in Zürich, Switzerland, in 1974. As a postscript to a paper on learning controllers, Seto Assilian and Abe Mamdani proposed fuzzy logic as an alternative approach to human-like controllers.

Experimental work was carried out at the Technical University of Denmark. The theoretical understanding and inspiration in relation to process control was gained mainly from papers written by Lotfi Zadeh and Abe Mamdani, and control experiments were performed in laboratory-scale processes such as, for example, a small heat exchanger. The rule-based approach that underlies the decision tables was also inspired by the instructions we found in a textbook for cement kiln operators, which contained 27 basic rules for manual operation of a cement kiln.

The first experiments using a real cement kiln were carried out at the beginning of 1978 at an FL Smidth cement plant in Denmark. At this stage of the development work, the attitude of the management was sceptical, partly because of the strange name 'fuzzy'. Other names were suggested, but eventually, with an increasing understanding by the management of the concept, it was decided to stay with the word fuzzy, a decision that has never been regretted since.

In 1980, FL Smidth launched the first commercial computer system for automatic kiln control based on fuzzy logic. To date, hundreds of kilns, mills, and other processes have been equipped with high-level fuzzy control by FL Smidth and other suppliers of similar systems.

Jens-Jørgen Østergaard
FL-Soft, Copenhagen

Preface

Abstract The objective of this textbook is to explain the behaviour of fuzzy-logic controllers. Under certain conditions, a fuzzy controller is equivalent to a proportional-integral-derivative (PID) controller. The equivalence enables the use of analysis methods from linear and nonlinear control theory. In the linear domain, PID tuning methods and stability criteria can be transferred to linear fuzzy controllers. The Nyquist plot shows the robustness of different settings of the fuzzy gain parameters. As a result, it can be guaranteed that a fuzzy controller will perform as well as any PID controller. In the nonlinear domain, the stability of four standard control surfaces can be analysed by means of describing functions and Nyquist plots. The self-organizing controller (SOC) has been shown to be a model reference adaptive controller. There is a possibility that a nonlinear fuzzy PID controller performs better than a linear PID controller, but there is no guarantee for the same. Even though a fuzzy controller is nonlinear in general, and commonly built in a trial-and-error fashion, we can conclude that control theory does provide tools for explaining the behaviour of fuzzy-control systems. Further studies are required, however, to find a design method such that a fuzzy-control system exhibits a particular behaviour in accordance with a set of performance specifications.

Fuzzy control is an attempt to make computers understand natural language and behave like a human operator. The first laboratory application (mid-1970s) was a two-input-two-output steam engine controller by Ebrahim (Abe) Mamdani and Seto Assilian, United Kingdom, and the first industrial application was a controller for a cement kiln by Holmblad and Østergaard, FL Smidth, Denmark. Today there is a tendency to combine the technology with other techniques. Fuzzy control together with artificial neural networks provides both the natural language interface from fuzzy logic and the learning capabilities of neural networks. Lately hybrid systems, including machine learning and artificial intelligence methods, increase the potential for intelligent systems.

As a follow-up on the pioneering work by Holmblad and Østergaard, which started at the Technical University of Denmark in the 1970s, I have taught fuzzy control over the Internet to students in more than 20 different countries since 1996. The course is primarily for graduate students, but senior undergraduates and PhD students also take the course. The material, a collection of downloadable lecture notes at 10–30 pages each, formed the basis for this textbook.

A fuzzy controller is in general nonlinear, and therefore the design approach is commonly one of trial and error. The objective of this book is to explain the behaviour of fuzzy-logic controllers in order to reduce the amount of trial-and-error content in a design phase.

Much material has been developed by applied mathematicians, especially with regard to stability analysis. Sophisticated mathematics is often required, which unfortunately makes the material inaccessible to most of the students of the Internet course. On the other hand, application-oriented textbooks that are easily accessible and have a wide coverage of the area are available. The design approach is nevertheless one of trial and error. The present book is positioned between mathematics and heuristics; it is a blend of control theory and trial-and-error methods. The key features of the book are summarized in the following four items.

- *Fundamental* The chapter on fuzzy reasoning presents not only fuzzy logic, but also classical set theory, two-valued logic, and two-valued rules of inference. The chapters concerning nonlinear fuzzy control rely on phase plane analysis, describing functions, and model reference adaptive control. Thus, the book presents the parts of control theory that are the most likely candidates for a theoretical foundation for fuzzy control, it links fuzzy control concepts backwards to the established control theory, and it presents new views of fuzzy control as a result.

- *Coherent* The analogy with PID control is the starting point for the analytical treatment of fuzzy control, and it pervades the whole book. Fuzzy controllers can be designed, equivalent to a P controller, a PD controller, a PID controller, or a PI controller. The PD control table is equivalent to a phase plane, and the stability of the nonlinear fuzzy controllers can be compared mutually, with their linear approximation acting as a reference. The self-organizing controller is an adaptive PD controller or PI controller. In fact, the title of the book could also have been *Fuzzy PID Control*.

- *Companion web site*[1] Many figures in the book are programmed in Matlab (trademark of The MathWorks, Inc.), and the programs are available on the companion web site. For each such figure, the name of the program that produced the figure is provided, in parentheses, in the caption of the figure. They can be recognized by the syntax *.m, where the asterisk stands for the name of the program. The list of figures provides a key and an overview of the programs.

- *Companion Internet course* The course concerns the control of an inverted pendulum problem or, more specifically, rule-based control by means of fuzzy logic. The inverted pendulum is rich in content, and therefore a good didactic vehicle used in courses around the world. In this course, students design and tune a controller that balances a ball on top of a moving cart. The course is based on a simulator, which runs in the Matlab environment, and the case is used throughout the whole course. The objectives of the course are as follows: to teach the basics of fuzzy control, to show how fuzzy logic is applied, and to teach fuzzy controller design. The core means of communication is email, and the didactic method is through email tutoring. An introductory course in automatic control is a prerequisite.

The introductory chapter of the book shows the design approach by means of an example. The book then presents set theory and logic as a basis for fuzzy logic and

[1]http://www.oersted.dtu.dk/31361

fuzzy reasoning, especially the so-called generalized modus ponens. A block-diagram of controller components and a list of design choices lead to the conditions for obtaining a linear fuzzy controller, the prerequisite for the fuzzy PID controller.

The following step is into the nonlinear domain, where everything gets more difficult, but also more interesting. The methods of phase plane analysis, model reference adaptive control, and describing functions provide a foundation for the design and fine-tuning of a nonlinear fuzzy PID controller.

The methods are demonstrated in a simulation of the inverted pendulum problem, the case study in the above-mentioned course on the Internet. Finally, a short chapter presents ideas for supervisory control based on experience in the process industry.

The book aims at an audience of senior undergraduates, first-year graduate students, and the practising control engineer. The book and the course assume that the student has an elementary background in linear differential equations and control theory corresponding to an introductory course in automatic control. Chapters 1, 2, 3, and 9 can be read with few prerequisites, however. Chapter 4 requires knowledge of PID control and Laplace transform, and chapters 5, 6, and 7 increasingly require background knowledge. Even the simulation study in chapter 8 requires some knowledge of state-space modelling for being fully appreciated. Mathematical shortcuts have been taken to preserve simplicity and avoid formalism.

Sections marked by an asterisk (*) may be skipped on a first reading; they are either very mathematical or very practically oriented, and thus off the main track of the book.

It is of course impossible to cover the entire spectrum of topic areas in one volume. I have drawn the line between fuzzy control and neuro-fuzzy control. The latter encompasses topics such as neural networks, learning, and model identification that could be included in a future edition.

Acknowledgements. I am pleased to acknowledge the many helpful suggestions I received from the late Lauritz Peter Holmblad, who acted as external supervisor on Masters projects at the Technical University of Denmark, and Jens-Jørgen Østergaard. They have contributed process knowledge, sound engineering solutions, and a historical continuity. Thanks to Peer Martin Larsen, I inherited all reports from the early days of fuzzy control at the university. I also had the opportunity to browse the archives of Abe Mamdani, then at Queen Mary College, London. I am also pleased to acknowledge the many helpful suggestions from Derek Atherton and Frank Evans, both in the United Kingdom, concerning nonlinear control, and, in particular, state-space analysis and describing functions. Last, but not least, former and present students at the university and on the Internet have contributed collectively with ideas and suggestions.

Jan Jantzen
Technical University of Denmark

List of Figures

1

Introduction

Fuzzy controllers appear in consumer products – such as washing machines, video cameras, cars – and in industry, for controlling cement kilns, underground trains, and robots. A *fuzzy controller* is an *automatic controller*, a self-acting or self-regulating mechanism that controls an object in accordance with a desired behaviour. The object can be, for instance, a robot set to follow a certain path. A fuzzy controller acts or regulates by means of rules in a more or less natural language, based on the distinguishing feature: fuzzy logic. The rules are invented by plant operators or design engineers, and fuzzy control is thus a branch of intelligent control.

1.1 What Is Fuzzy Control?

Traditionally, computers make rigid *yes* or *no* decisions, by means of decision rules based on two-valued logic: *true–false*, *yes–no*, or 1 – 0. An example is an air conditioner with thermostat control that recognizes just two states: above the desired temperature or below the desired temperature. *Fuzzy logic,* on the other hand, allows a graduation from *true* to *false*. A fuzzy air conditioner may recognize ‘warm’ and ‘cold’ room temperatures. The rules behind this are less precise, for instance:

If the room temperature is warm and slightly increasing, then increase the cooling.

Many classes or *sets* have *fuzzy* rather than sharp boundaries, and this is the mathematical basis of fuzzy logic; the set of ‘warm’ temperature measurements is one example of a fuzzy set.

The core of a fuzzy controller is a collection of *verbal* or *linguistic* rules of the *if–then* form. Several variables may occur in each rule, both on the *if*-side and the *then*-side. Reflecting expert opinions, the rules can bring the reasoning used by computers closer to that of human beings.

In the example of the fuzzy air conditioner, the controller works on the basis of a temperature measurement. The room temperature is just a number, and more information

Foundations of Fuzzy Control Jan Jantzen
© 2007 John Wiley & Sons, Ltd. ISBN: 0-470-02963-3

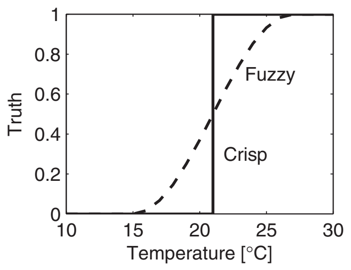


Figure 1.1: A warm room. The crisp air conditioner considers any temperature above 21 °C warm. The fuzzy air conditioner considers gradually warmer temperatures. (figwarm.m)

is necessary to decide whether the room is warm. Therefore the designer must incorporate a human's perception of warm room temperatures. A straightforward implementation is to evaluate beforehand all possible temperature measurements. For example, on a scale from 0 to 1, *warm* corresponds to 1 and *not warm* corresponds to 0:

Measurements (°C):	...	15	17	19	21	23	25	27	...
		|	|	|	|	|	|	|	
Grade:	...	0	0.1	0.3	0.5	0.7	0.9	1	...

The example uses *discrete* temperature measurements, whereas Figure 1.1 shows the same idea graphically in the form of a *continuous* mapping of temperature measurements to truth-values. The mapping is arbitrary, that is, based on preference and not mathematical reason.

1.2 Why Fuzzy Control?

If PID control (proportional-integral-derivative control) is inadequate – for example, in the case of higher-order plants, systems with a long deadtime, or systems with oscillatory modes (Åström and Hägglund 1995) – fuzzy control is an option. But first, let us consider why one would *not* use a fuzzy controller:

- The PID controller is well understood, easy to implement – both in its digital and analog forms – and it is widely used. By contrast, the fuzzy controller requires some knowledge of fuzzy logic. It also involves building arbitrary membership functions.
- The fuzzy controller is generally nonlinear. It does not have a simple equation like the PID, and it is more difficult to analyse mathematically; approximations are required, and it follows that stability is more difficult to guarantee.
- The fuzzy controller has more tuning parameters than the PID controller. Furthermore, it is difficult to trace the data flow during execution, which makes error correction more difficult.

On the other hand, fuzzy controllers are used in industry with success. There are several possible reasons:

- Since the control strategy consists of *if–then* rules, it is easy for a plant operator to read. The rules can be built from a vocabulary containing everyday words such as ‘high’, ‘low’, and ‘increasing’. Plant operators can embed their experience directly.

- The fuzzy controller accommodates many inputs and many outputs. Variables can be combined in an *if–then* rule with the connectives *and* and *or*. Rules are executed in parallel, implying a recommended action from each. The recommendations may be in conflict, but the controller resolves conflicts.

Fuzzy logic enables non-specialists to design control systems, and this may be the main reason for its success.

1.3 Controller Design

Established design methods such as pole placement, optimal control, and frequency response shaping only apply to linear systems, whereas fuzzy control is generally nonlinear. Since our knowledge of the behaviour of nonlinear systems is limited, compared with the situation in the linear domain, this book is based on a design procedure founded on linear control:

1. Design a PID controller

2. Replace it with a linear fuzzy controller

3. Make it nonlinear

4. Fine-tune it.

The idea is to exploit the design methods within PID control and carry them forward to fuzzy control. The design procedure is feasible, only because it is possible to build a linear fuzzy controller that functions exactly as any PID controller does. The following example introduces the design procedure.

1.4 Introductory Example: Stopping a Car

Consider this problem: Model the behaviour of a human driver stopping a car at a red stop light by using the brake pedal. Figure 1.2 defines the symbols.

Model the car first. Disregarding engine dynamics, skidding, slip, and friction – other than the frictional forces in the brake pads – the force F causes an acceleration a according to Newton’s second law of motion $F = ma$. Acceleration is the derivative of velocity $\dot{y}$, which is, in turn, the derivative of the position y. Thus $a = \ddot{y}$, and we can write the differential equation that models the motion of the car as

$$F = m\ddot{y} \Leftrightarrow \ddot{y} = \frac{F}{m} \tag{1.1}$$

For a Volkswagen Caddy Van (diesel, 2-L engine) the mass, without load and including the driver, is approximately 1500 kg. Assume that the stop light changes to red when the car

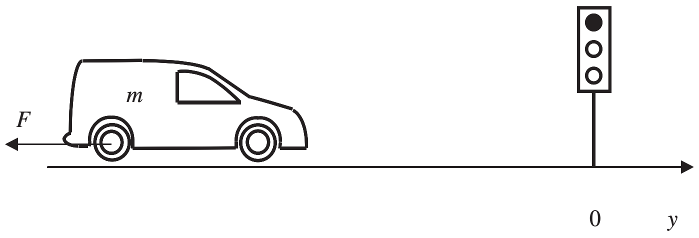


Figure 1.2: Stopping a car. Position y is positive towards the right, with zero at the stop light. The brakes act with a negative force F on the mass m.

is 25 m (82 ft) away at a speed of 10 m/s (36 km/h or 22.7 mph). We have thus identified the following constants:

$$\begin{aligned} m &= 1500 \\ y(0) &= -25 \\ \dot{y}(0) &= 10 \end{aligned}$$

Once the speed is zero, the car will not move anymore. The variable F is thus negative or zero, since the brake is our only means of control. According to specifications, the distance at which the brakes have to be applied when the car is at a speed of from 80 km/h (49.7 mph.) to bring the speed to zero is $y = 27.3$ m. As all the kinetic energy is converted to work, we have, on the average,

$$\frac{1}{2}m\dot{y}^2 = Fy$$

and thus

$$\begin{aligned} F &= \frac{1}{y}\frac{1}{2}m\dot{y}^2 \\ &= \frac{1}{27.3}\frac{1}{2}1500\left(\frac{80\,000}{3600}\right)^2 \\ &\approx 13\,600 \end{aligned}$$

We shall therefore assume that the automatic brake system limits the magnitude of the brake force to 13 600 N (newton). The control signal is thus subject to the constraints

$$-13\,600 \leq F \leq 0$$

We can now simulate the system, and the objective is to study the behaviour related to various control strategies.

Step 1: Design a PID controller

Our first choice is to model the driver as a proportional controller,

$$F = K_p e \tag{1.2}$$

where K_p is the proportional gain, that can be adjusted. To interpret, the driver presses the brake pedal hard when the distance is large, relaxes the pressure as the distance decreases, and finally releases the pedal when arriving at the stop light. The error $e \geq 0$ is the position error between the stop line *Ref* and the current position $y \leq 0$,

$$e = Ref - y \tag{1.3}$$

The force F arises not from the engine, but from an opposite friction force in the brakes. Therefore K_p must be negative. The closed-loop system equations are obtained by inserting Equation (1.2) into Equation (1.1):

$$\ddot{y} = \frac{K_p e}{m} = \frac{K_p}{m} Ref - \frac{K_p}{m} y = -\frac{K_p}{m} y \tag{1.4}$$

since $Ref = 0$.

It turns out that only a particular controller setting ($K_p = -240$) will work such that the car stops at the stop light after about 10 s. A smaller magnitude of K_p stops the car after the light (overshoot), and a larger magnitude of K_p stops the car short of the light. The solution to Equation (1.4) shows why:

$$y(t) = \left(-\frac{5}{\sqrt{\frac{|K_p|}{m}}} - \frac{25}{2} \right) \exp^{-\sqrt{\frac{|K_p|}{m}}t} + \left(\frac{5}{\sqrt{\frac{|K_p|}{m}}} - \frac{25}{2} \right) \exp^{\sqrt{\frac{|K_p|}{m}}t}$$

Setting $K_p = -240$ suppresses the second exponential term, but if K_p is slightly different, the second exponential will diverge to plus or minus infinity.

Because of the long stopping time, and the highly sensitive solution, we conclude that proportional control is unrealistic.

Our second choice is to model the driver as a proportional-derivative (PD) controller, since it is likely that the driver takes the velocity into account. Figure 1.3 shows a Simulink (trademark of The MathWorks, Inc.) model, which includes the constraint on the brake force, and the model is therefore nonlinear. The controller is

$$F = K_p \left(e + T_d \dot{e} \right) \tag{1.5}$$

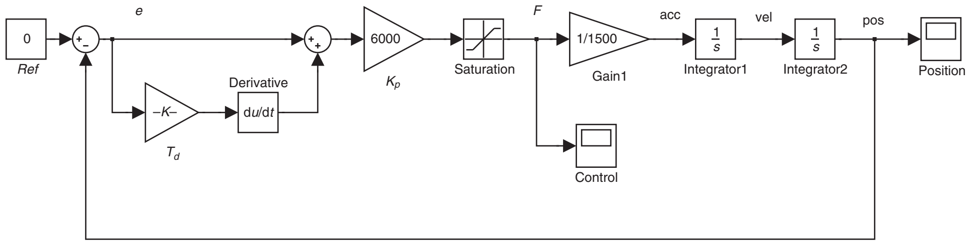


Figure 1.3: Simulink block diagram. A PD controller brakes the car from initial conditions $y(0) = -25$, $\dot{y}(0) = 10$. (figcarpd.mdl)

where T_d is the derivative gain, which can be adjusted. To interpret, the derivative action calls for extra brake force when the velocity is high. The closed-loop system equations are obtained by inserting Equation (1.5) into Equation (1.1):

$$\ddot{y} = \frac{K_p\,(e + T_d\dot{e})}{m} = -\frac{K_pT_d}{m}\dot{y} - \frac{K_p}{m}y \tag{1.6}$$

There will be a steady state solution, since in steady state the system is at rest, that is, $\ddot{y} = \dot{y} = 0$, and insertion yields the solution $y = 0$; this is in accordance with the problem specification. The transfer function of the closed-loop system is

$$\frac{y(s)}{Ref} = \frac{\frac{K_p}{m}}{s^2 + \frac{K_p}{m}T_ds + \frac{K_p}{m}} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_ns + \omega_n^2} \tag{1.7}$$

The last expression is a general second-order system with natural frequency ω_n – the frequency of oscillation of the system without damping – and damping ratio ζ. It is useful, because we are looking for a solution without overshoot, which is as fast as possible; this corresponds to the case $\zeta = 1$, which yields a critically damped response. From Equation (1.7) we derive

$$\zeta = \frac{1}{2}\sqrt{\frac{K_p}{m}}T_d$$

Applying $\zeta = 1$ gives us an optimal tuning relationship

$$T_d = \frac{2}{\sqrt{\frac{K_p}{m}}}$$

It ensures that the response has no overshoot and there is a horizontal tangent at $y = 0$; consequently the velocity will be zero when the car arrives at the stop light.

Figure 1.4 shows the response with

$$K_p = 6000$$
$$T_d = 1$$

The figure also shows a fuzzy controller response for comparison, but we shall refer to that later. During the first 1.5 s, the control signal is zero, because the proportional action is positive and larger in magnitude than the derivative action, which is negative. Since the resultant action is positive, the saturation limits the signal to zero (the car moves owing to the initial velocity of 10 m/s). At time $t = 1.5$, the derivative action takes over and starts to brake the car. To interpret, the controller waits 1.5 s until it kicks in, applies less than maximum force about half a second later, and then relaxes the brake gently. It takes about 5 s to stop the car.

Step 2: Replace it with a linear fuzzy controller

A fuzzy controller consists of *if–then* rules describing the action to be taken in various situations. We will consider the situations where the distance to the stop light is long or

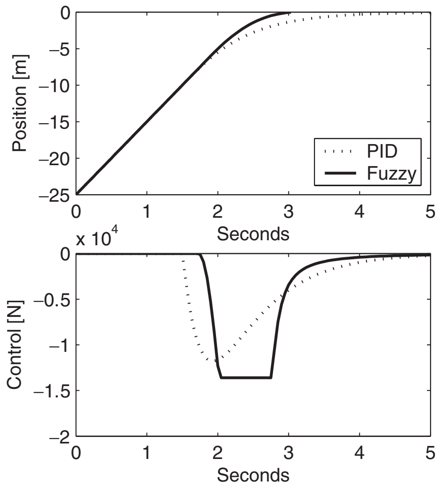


Figure 1.4: Stopping a car. Comparison between a PID controller and a fuzzy controller. (figstopcar.m)

short and situations where the car is approaching fast or slowly. A complete rule base of all possible combinations contains four rules:

if the distance is long and the approach is fast, then brake zero; (1.8)

if the distance is long and the approach is slow, then brake zero; (1.9)

if the distance is short and the approach is fast, then brake hard; (1.10)

if the distance is short and the approach is slow, then brake zero. (1.11)

The linguistic terms must be specified precisely for a computer to execute the rules. Figure 1.5 shows how to implement ‘long’, as in ‘distance is long’, as a fuzzy *membership function*, shaped like the letter ‘s’. The x-axis is the *universe,* the interval [0,100] % of the full range of 25 m. The y-axis is the *membership grade*, that is, how compatible a distance measurement is with our perception of the term ‘long’. For instance, a distance of 25 m (100 %) has membership 1, because the distance is definitely long, while half that distance is long to a degree of just 0.5. Note that the x-axis corresponds to the previously defined error e, scaled onto a standard range of percentages relative to the maximum distance.

The term ‘*fast*’, as in ‘approach is fast’, is another membership function. The x-axis is again percentages of the full range (10 m/s), but the numbers are negative to emphasize that the speed is decreasing rather than increasing. The x-axis corresponds to the previously defined time derivative $\dot{e}$ scaled onto the universe; when the speed decreases, $\dot{e}$ is negative. The -100 % corresponds to the maximum speed of 10 m/s.

Similarly, the membership function for ‘short’ is just a reflection of the membership function ‘long’, and the membership function ‘slow’ is a reflection of ‘fast’.

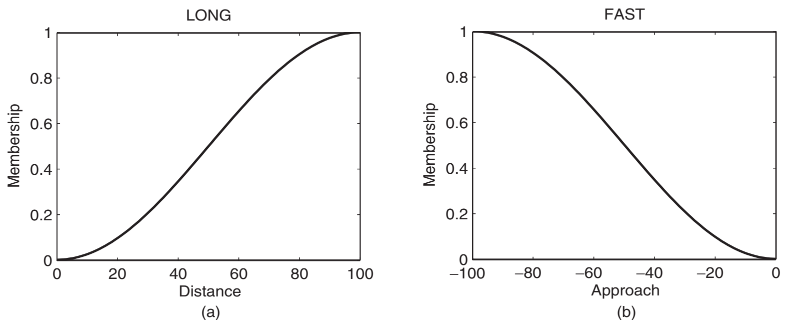


Figure 1.5: Fuzzy membership functions. The (a) specifies a LONG distance, and the (b) specifies a FAST approach. The curves correspond to the first rule (1.8). (figmfcar.m)

Turning to the *then*-side of the rules, the term '*zero*' is brake force 0, a constant. The term '*hard*' is the full brake force of -100 %, which is also a constant.

The following chapters will show how to design a *linear* fuzzy controller, with a performance that is exactly the same as the PD controller in the previous step. It is a design aid, because the PD controller, with its tuning, settles many design choices for the fuzzy controller. One requirement is that the membership functions should be linear.

At the end of this step, we have a fuzzy controller (not shown), with a response exactly as the dotted PD response in Figure 1.4.

Step 3: Make it nonlinear

Stepping into the nonlinear domain usually entails a trial and error design procedure, but the following chapters provide methods such that at least some *analysis* is possible.

The nonlinear fuzzy controller uses the nonlinear membership functions in Figure 1.5. The response is close to the PD controller response, and is therefore not shown.

Step 4: Fine-tune it

After adjusting one tuning factor (input gain on the error, GE, increased from 10 to 13), the response is as in Figure 1.4. The response is better than in PD control. The lower plot with the control signals shows what happens: The fuzzy controller waits longer before it kicks in, and then it uses all the available brake force. Thereafter it releases the brake quicker than the PD controller.

The example shows that good knowledge of the plant to be controlled is beneficial; to analyse stability, a mathematical model is even necessary. But more importantly, the PD controller design step gave us a tuning (K_p and T_d) that we could transfer to the fuzzy controller. Thereby, the PD controller constitutes a reference for the assessment of the performance of the fuzzy controller.

1.5 Summary

It is quite difficult to design a fuzzy controller, because it is in general nonlinear, and nonlinear systems are more or less unpredictable. Instead we propose to stay as long as possible in the linear domain, reflected in the proposed design procedure.

The idea is to start from PID control and design a linear fuzzy controller that is equivalent to the pre-designed PID controller. At this point, all the results from linear control theory can be applied, including tuning methods and stability calculations. In the next phase, the fuzzy controller is made nonlinear.

The design procedure has limited scope in the sense that it requires a PID controller be built to control the plant. As a compensation for the limited scope, the design procedure provides reliability: it guarantees that the fuzzy controller performs at least as well as its pre-designed PID controller. There is a possibility, but no guarantee, that it will perform better.

Some of the following chapters provide tools for analysing the nonlinear fuzzy controller, in particular, phase plane analysis and describing functions. Yet, trial and error is still a characteristic of the design procedure.

1.6 Notes and references

In the mid-1960s, Lotfi A. Zadeh (born in 1921 in Azerbaijan) of the University of California at Berkeley, USA, invented the theory of fuzzy sets. He argued that, more often than not, the classes of objects encountered in the real physical world have imprecisely defined criteria for membership (Zadeh 1965). For example, the 'class of numbers that are much greater than 1', or the 'class of tall human beings' have ill-defined boundaries. Yet, such imprecisely defined classes play an important role in human reasoning and communication.

Ebrahim (Abe) H. Mamdani, a control engineer at Queen Mary College in London (now Emeritus Professor at Imperial College), was attempting to develop an adaptive system that could learn to control an industrial process (Figure 1.6). He used a steam engine as a laboratory model, and with his colleagues set up a program that would teach the computer to control the steam engine by monitoring a human operator. At this point, Mamdani's

Figure 1.6: E.H. Mamdani

research student, Seto Assilian, tried to apply fuzzy logic. He created a set of simple rules in fuzzy terms, and Mamdani and Assilian then studied ways to use fuzzy rules of thumb directly in automating process controls. A few years later, Mamdani and Procyk managed to develop a linguistic self-organizing controller (Procyk and Mamdani 1979). It was an adaptive controller that was able to learn how to control a wide variety of processes, nonlinear and multi-variable, in a relatively short time. It was called '*self-organizing*' because at that point in time the meaning of the words 'adaptive' and 'learning' had not yet been agreed upon.

The work of the pioneers led to a growing literature in fuzzy control and wide-ranging applications, as Table 1.1 illustrates.

In Japan, Michio Sugeno (1989) developed a self-learning fuzzy controller. Twenty control rules determined the motion of a model car. Each rule recommends a specific change in direction, based on the car's distance from the walls of a corridor. The controller drives the car through angled corridors, after a learning session where a 'driving instructor' pulls it through the route a few times. Self-learning controllers that derive their own rules automatically are interesting because they could reduce the effort needed for translating human expertise into a rule base.

The first industrial application was in 1978, where a fuzzy controller was operating in closed loop on a rotary cement kiln in Denmark. Fuzzy control then became a commercial product of the Danish cement company F.L. Smidth & Co. The fuzzy control research program in Denmark was initiated in 1974 (Larsen 1981).

Table 1.1: Milestones in early fuzzy history.

Year	Event	Reference
1965	First article on fuzzy sets	Zadeh (1965)
1972	A rationale for fuzzy control	Zadeh (1972)
1973	Linguistic approach	Zadeh (1973)
1974	Fuzzy-logic controller	Assilian and Mamdani (1974)
1976	Warm-water plant	Kickert and van Nauta Lemke (1976)
1977	Table-based controller	Mamdani (1977)
1977	Heat exchanger	Østergaard (1977)
1977	Self-organizing controller	Procyk and Mamdani (1979)
1980	Fuzzy conditional inference	Fukami *et al.* (1980)
1980	Cement kiln controller	Holmblad and Østergaard (1982)
1983	Train operation	Yasunobu *et al.* (1983)
1984	Parking control of a model car	Sugeno *et al.* (1989)
1985	Fuzzy chip	Togai and Watanabe (1985)
1986	Fuzzy controller hardware system	Yamakawa and Miki (1986)
1987	Sendai subway in operation	Yasunobu *et al.* (1983)
1989	Fuzzy home appliances sold in Japan	
1989	The LIFE project is started in Japan	
1990	Rule learning by neural nets	Kosko (1992)
1990	Hierarchical controller	Østergaard (1990), (1996)

The Laboratory for International Fuzzy Engineering (*LIFE*), Yokohama, was set up by the Japanese Ministry of International Trade and Industry in 1989. It had a 6-year budget of 5 billion yen and a research staff of around 30. LIFE conducted basic research with universities and member Japanese companies and subsidiaries of U.S. and European companies, including Matsushita, Hitachi, Omron, and VW, about 50 companies in all. The research program was trimmed to five major projects: image understanding, fuzzy associative memory, fuzzy computing, intelligent interface, and the intelligent robot. They were all carried out with two themes in view: a navigation system for the blind and home computing.

A European network of excellence called *ERUDIT*[1] was initiated in 1995 with support from the European Commission. ERUDIT, which lasted 6 years, was an open network for uncertainty modelling and fuzzy technology, aimed at putting European industry at the leading edge. The network was followed by another network EUNITE[2] with a broader scope: smart adaptive systems. And this was in turn followed by a coordinated action NISIS[3] with an even broader scope: nature-inspired systems.

For further information

Beginners may start with two articles in Institute of Electrical and Electronics Engineers (IEEE) Spectrum (Zadeh 1984, Self 1990) and then move on to the more advanced textbook by Zimmermann (1993); half of it is dedicated to fuzzy set theory and the other half to applications.

The terms ‘rule base’ and ‘inference engine’ are loans from the field of expert systems, and Lee (1990) uses these to give a wide survey of the whole area of fuzzy control. The article lists 150 references. The book by Ross (1995) is oriented towards applications in engineering and technology with many calculated examples.

A major reference on fuzzy control is the book by Driankov *et al.* (1996). It is explicitly targeted at the control engineering community, in particular, engineers in industry, and university students. Chapter 3 gives more specific references related to fuzzy control.

Industrial applications are described in a special issue of the journal *Fuzzy Sets and Systems*, for instance, the fuzzy car by Sugeno *et al.* (1989) and an arc welding robot by Murakami *et al.* (1989). There are more early applications in the classical book by Sugeno (1985). Ten years later, Constantin von Altrock (1995) described more than 30 case studies from companies that employed fuzzy and neuro-fuzzy methods. The FL Smidth controller is described in detail in Holmblad and Østergaard (1982).

There are more than 10 journals related to fuzzy sets. Two of the major journals are *Fuzzy Sets and Systems* and *International Journal of Approximate Reasoning*, both published by Elsevier, and a third one is *Journal of Intelligent and Fuzzy Systems*, published by IOS Press, Netherlands. The Institute of Electrical and Electronics Engineers, *IEEE,* started a journal in 1992 called *IEEE Transactions on Fuzzy Systems*. The *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems* is published four times per year by World Scientific Publishing Co. It is a forum for research on imprecise, vague, uncertain, and incomplete knowledge. Other journals that occasionally have fuzzy control articles are

[1]http://www.erudit.de
[2]http://www.eunite.org
[3]http://www.nisis.de

Automatica, the control section of *IEE Proceedings*, *IEEE Transactions on Systems Man and Cybernetics*, *IEEE Transactions on Computers, Control Engineering Practice,* and the *International Journal of Man-Machine Studies*.

There is an active newsgroup called `comp.ai.fuzzy`[4]. It supplies useful news, conference announcements, and discussions.

There are two major professional organizations. The International Fuzzy Systems Association (IFSA) is a worldwide organization dedicated to fuzzy sets. IFSA publishes the *International Journal of Fuzzy Sets and Systems* (24 issues per year), holds international conferences, establishes chapters, and sponsors activities. The other organization is the North American Fuzzy Information Processing Society, NAFIPS[5], with roughly the same purpose.

[4]http://groups.google.com/group/comp.ai.fuzzy

[5]http://nafips.ece.ualberta.ca/

2

Fuzzy Reasoning

In classical logic, an assertion is either true or false – not something in between – and fuzzy logic extends classical logic by allowing intermediate truth values between zero and one. An assertion can be more or less true in fuzzy logic.

A computer can interpret a linguistic statement such as

if the washing machine is half full, then use less water.

Fuzzy logic adds intelligence to the washing machine since the computer infers an action from a set of if–then rules. Fuzzy logic is 'computing with words', to quote the creator of fuzzy logic, Lotfi A. Zadeh.

The objective of this chapter is to select and emphasize the concepts of fuzzy logic that are necessary and sufficient from the point of view of a control engineer.

2.1 Fuzzy Sets

Fuzzy sets are a further development of mathematical set theory, first studied formally by the German mathematician Georg Cantor (1845–1918). It is possible to express most of mathematics in the language of set theory, and researchers are today looking at the consequences of 'fuzzifying' set theory, resulting in, for example, fuzzy logic, fuzzy numbers, fuzzy intervals, fuzzy arithmetic, and fuzzy integrals. Fuzzy logic is based on fuzzy sets, and with fuzzy logic a computer can process words from natural language, such as 'small', 'large', and 'approximately equal'.

Although elementary, the following sections include the basic definitions of classical set theory. This is to shed light on the original ideas. But only those basic definitions that are necessary and sufficient will be presented; students interested in delving deeper into set theory and logic can, for example, read the comprehensive treatment by Stoll (1979).

Classical sets

According to Cantor, a *set* $\mathcal{X}$ is a collection of definite, distinguishable objects of our intuition that can be treated as a whole. The objects are the *members* of $\mathcal{X}$. The concept 'objects of our intuition' gives us great freedom of choice, even with respect to sets with infinitely many members. Objects must be 'definite': given an object and a set, it must be possible to determine whether the object is, or is not, a member of the set. Objects must also be 'distinguishable': given a set and its members, it must be possible to determine whether any two members are different or the same.

The members completely define a set. To determine membership, it is necessary that the sentence 'x is a member of $\mathcal{X}$', where x is replaced by an object and $\mathcal{X}$ by the name of a set, is either true or false. We use the symbol $\in$ and write $x \in \mathcal{X}$ if object x is a member of the set $\mathcal{X}$. The assumption that the members determine a set is equivalent to saying, 'two sets $\mathcal{X}$ and $\mathcal{Y}$ are equal, $\mathcal{X} = \mathcal{Y}$, iff (if and only if) they have the same members'. The set whose members are the objects $x_1, x_2, \ldots, x_n$ is written as

$$\{x_1, x_2, \ldots, x_n\}\,.$$

In particular, the set with no members is the *empty set* symbolized by $\emptyset$. The set $\mathcal{X}$ is included in $\mathcal{Y}$,

$$\mathcal{X} \subseteq \mathcal{Y}$$

iff each member of $\mathcal{X}$ is a member of $\mathcal{Y}$. We also say that $\mathcal{X}$ is a *subset* of $\mathcal{Y}$, and it means that, for all x, if $x \in \mathcal{X}$, then $x \in \mathcal{Y}$. The empty set is a subset of every set.

Almost anything called a *set* in ordinary conversation is acceptable as a mathematical set, as the next example indicates.

Example 2.1.1 *Classical sets*
The following are lists or collections of definite and distinguishable objects, and therefore sets in the mathematical sense:

(a) The set of non-negative integers less than 3. This is a finite set with three members $\{0, 1, 2\}$.

(b) The set of live dinosaurs in the basement of the British Museum. This set has no members and is the empty set $\emptyset$.

(c) The set of measurements greater than 10 volts. Even though this set is infinite, it is possible to determine whether a given measurement is a member.

(d) The set $\{0, 1, 2\}$ *is the set from (a). Since* $\{0, 1, 2\}$ *and* $\{2, 1, 0\}$ *have the same members, they are equal sets. Moreover,* $\{0, 1, 2\} = \{0, 1, 1, 2\}$ *for the same reason.*

(e) The members of a set may themselves be sets. The set

$$\mathcal{X} = \{\{1, 3\}, \{2, 4\}, \{5, 6\}\}$$

is a set with three members, namely, $\{1, 3\}$, $\{2, 4\}$, *and* $\{5, 6\}$. *Matlab supports sets of sets, or nested sets, in cell arrays.*

(f) It is possible in Matlab to assign an empty set, for instance, $x = \{[\,]\}$.

Although the brace notation $\{\cdot\}$ is practical for listing sets of a few elements, it is impractical for large sets and impossible for infinite sets. How do we then define a set with a large number of members?

For an answer we require a few more concepts to be defined. A *proposition* is an assertion (declarative statement) that can be classified as either true or false. By a *predicate* in x we understand an assertion formed using a formula in x. For instance, '$0 < x \leq 3$', or '$x > 10$ volts' are predicates. They are not propositions, however, since they are not necessarily true or false. Only if we assign a value to the variable x, each predicate becomes a proposition. A predicate $P(x)$ in x defines a set $\mathcal{X}$ by the convention that the members of $\mathcal{X}$ are exactly those objects a such that $P(a)$ is true. In mathematical notation,

$$\{x \mid P(x)\},$$

is 'the set of all x such that $P(x)$' is true. Thus $a \in \{x \mid P\ (x)\}$ iff $P(a)$ is a true proposition.

Fuzzy sets

A system in which propositions must be either true or false, but not both, uses a two-valued logic. As a consequence, what is not true is false and vice versa; this is the *law of the excluded middle*. But two-valued logic is only an approximation to human reasoning, as Zadeh observed:

Clearly, the "class of all real numbers that are much greater than 1," or "the class of beautiful women," or "the class of tall men," do not constitute classes or sets in the usual mathematical sense of these terms. (Zadeh 1965)

We might call it *Zadeh's challenge*, because he focuses on the elasticity in the meaning of terms such as 'much', 'beautiful', and 'tall'. To define the set of tall men as a classical set, one would use a predicate $P(x)$, for instance $x \geq 176$, where x is the height of a person, and the right hand side of the inequality is a threshold value in centimeters (176 cm $\simeq$ 5 ft 9 in.). This is an abrupt approximation to the meaning of 'tall'.

Following Zadeh a *membership grade* allows finer detail, such that the transition from membership to non-membership is gradual rather than abrupt. The membership grade for all members defines a *fuzzy set* (Figure 2.1).

Definition *Fuzzy set*. Given a collection of objects $\mathcal{U}$, a fuzzy set $\mathcal{A}$ in $\mathcal{U}$ is defined as a set of ordered pairs

$$\mathcal{A} \equiv \{\langle x, \mu_{\mathcal{A}}(x)\rangle \mid x \in \mathcal{U}\}, \tag{2.1}$$

where $\mu_{\mathcal{A}}(x)$ is called the *membership function* for the set of all objects x in $\mathcal{U}$.

The symbol '$\equiv$' stands for 'defined as'. The membership function relates to each x a membership grade $\mu_{\mathcal{A}}(x)$, a real number in the closed interval [0, 1]. Notice that it is now necessary to work with pairs $\langle x, \mu_{\mathcal{A}}(x)\rangle$, whereas for classical sets a list of objects suffices, since their membership is understood. An *ordered pair* $\langle x, y\rangle$ is a list of two objects, in which the object x is considered as the first and y as the second (note: in the set $\{x, y\}$ the order is insignificant).

The term *fuzzy* (synonymous with indistinct) suggests a boundary zone, rather than an abrupt frontier. Indeed, fuzzy logicians speak of classical sets being *crisp sets,* to distinguish them from fuzzy sets. As with crisp sets, we are only guided by intuition in deciding which objects are members and which are not; a formal basis for determining the membership grade of a fuzzy set is absent. The membership grade is a precise, but arbitrary, measure; it rests on personal preference, and not on reason.

The definition of a fuzzy set extends the definition of a classical set, because membership values μ are permitted in the interval $0 \leq \mu \leq 1$, and the higher the value, the higher the

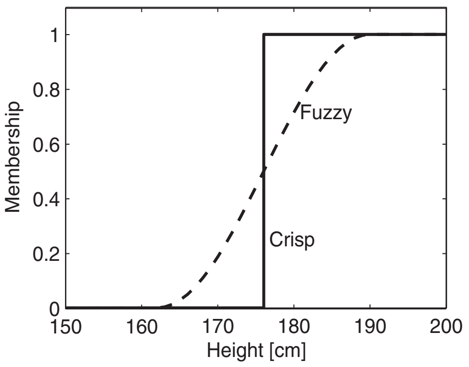


Figure 2.1: Two definitions of the set of "tall men", a crisp set and a fuzzy set. (figtall.m)

membership. A classical set is consequently a special case of a fuzzy set, with membership values restricted to $\mu \in \{0, 1\}$.

A single pair $\langle x, \mu(x)\rangle$ is a fuzzy *singleton*; thus the whole set can be viewed as the union of its constituent singletons.

Example 2.1.2 *Fuzzy sets*
The following are sets that could be described by fuzzy membership functions:

(a) The set of real numbers $x \gg 1$ *(x much greater than one).*

(b) The set of high temperatures, the set of strong winds, or the set of nice days are fuzzy sets in weather reports.

(c) The set of young people. A 1-year-old baby will clearly be a member of the set of young people, and a 100-year-old person will not be a member of this set. A person aged 30 might be young to the degree 0.5.

(d) The set of adults. The Danish railways allow children under the age of 15 to travel at half-price. Adults are thus defined as the set of passengers aged 15 or older. By this definition, the set of adults is a crisp set.

(e) A predicate may be crisp but it may be perceived as fuzzy: a speed limit of 60 km/h is taken to be an elastic range of more or less acceptable speeds within, say, 60–70 km/h ($\simeq$37–44 *mph) by some drivers. Notice how, on the one hand, the traffic law is crisp while, on the other hand, those drivers' understanding of the law is fuzzy.*

Universe

Members of a fuzzy set are taken from a *universe of discourse,* or *universe* for short. The universe consists of all objects that can come into consideration, compare the set $\mathcal{U}$ in Equation (2.1). The universe depends on the context, as the next example shows.

Example 2.1.3 *Universes*

(a) The set $x \gg 1$ *could have as a universe all real numbers, alternatively all positive integers.*

(b) The set of young people could have all human beings in the world as its universe. Alternatively, it could have the numbers between 0 *and* 100*; these would then represent the age in years.*

(c) The universe depends on the measuring unit; a duration in time depends on whether it is measured in hours, days, or weeks.

(d) A non-numerical quantity, for instance, taste*, must be defined on a* psychological continuum*; an example of such a universe is* $\mathcal{U} = \{bitter, sweet, sour, salt, hot\}$.

A programmer can exploit the universe to suppress faulty measurement data, for instance, negative values for a duration of time, by making the program consult the universe.

Membership function

There are two alternative ways to represent a membership function: continuous or discrete. A continuous fuzzy set $\mathcal{A}$ is defined using a continuous membership function $\mu_{\mathcal{A}}(x)$. A *trapezoidal* membership function is a piecewise linear, continuous function, controlled by four parameters $\{a, b, c, d\}$ (Jang *et al.* 1997)

$$\mu_{Trapezoid}(x; a, b, c, d) = \left\{ \begin{array}{ll} 0 & , x \le a \\ \frac{x-a}{b-a} & , a \le x \le b \\ 1 & , b \le x \le c \\ \frac{d-x}{d-c} & , c \le x \le d \\ 0 & , d \le x \end{array} \right\}, x \in \mathbb{R} \tag{2.2}$$

The parameters $a \le b \le c \le d$ define four breakpoints, designated as follows: left footpoint (a), left shoulderpoint (b), right shoulderpoint (c), and right footpoint (d). Figure 2.2 (a) illustrates a trapezoidal membership function.

A *triangular* membership function is piecewise linear, and derived from the trapezoidal membership function by merging the two shoulderpoints into one, that is, setting $b = c$, as in Figure 2.2 (b).

Smooth, differentiable versions of the trapezoidal and triangular membership functions can be obtained by replacing the linear segments corresponding to the intervals $a \le x \le b$ and $c \le x \le d$ by a nonlinear function, for instance, a half period of a cosine function,

$$\mu_{STrapezoid}(x; a, b, c, d) = \left\{ \begin{array}{ll} 0 & , x \le a \\ \frac{1}{2} + \frac{1}{2}\cos(\frac{x-b}{b-a}\pi) & , a \le x \le b \\ 1 & , b \le x \le c \\ \frac{1}{2} + \frac{1}{2}\cos(\frac{x-c}{d-c}\pi) & , c \le x \le d \\ 0 & , d \le x \end{array} \right\}, x \in \mathbb{R}$$

We call it *STrapezoid*, for '*smooth trapezoid*' or '*soft trapezoid*'. Figures 2.2 (c–d) illustrate the smooth membership functions. Other possibilities exist for generating smooth trapezoidal functions, for example, Gaussian, generalized bell, and sigmoidal membership functions (Jang *et al.* 1997).

Discrete fuzzy sets are defined by means of a discrete variable x_i $(i = 1, 2, \ldots)$. A discrete fuzzy set $\mathcal{A}$ is defined by ordered pairs,

$$\mathcal{A} = \{\langle x_1, \mu(x_1)\rangle, \langle x_2, \mu(x_2)\rangle, \ldots \mid x_i \in \mathcal{U}, i = 1, 2, \ldots\}$$

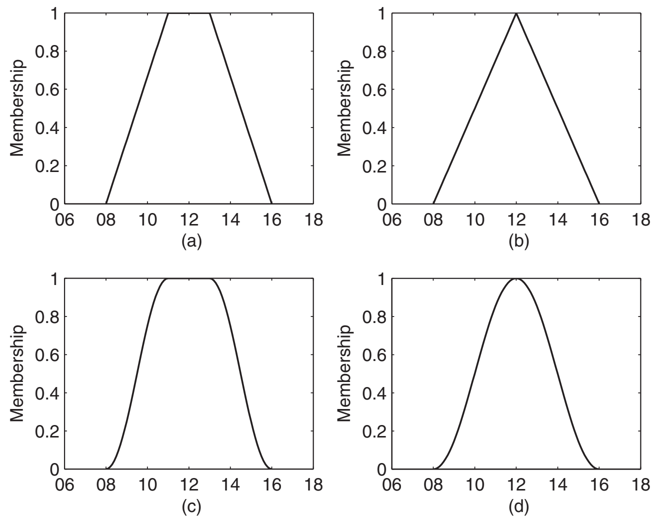


Figure 2.2: Around noon. Four possible membership functions representing the time 'around noon': (a) trapezoidal, (b) triangular, (c) smooth trapezoid, and (d) smooth triangular. The universe is the hours of the day in 24-hour format. (figmf0.m)

Each membership value $\mu(x_i)$ is an evaluation of the membership function μ at a discrete point x_i in the universe $\mathcal{U}$, and the whole set is a collection, usually finite, of pairs $\langle x_i, \mu(x_i)\rangle$.

Example 2.1.4 *Discrete membership function*

To achieve a discrete triangular membership function from the trapezoidal, function Equation (2.2), assume the universe is a vector **u** *of seven elements. In Matlab notation,*

$$\mathbf{u} = [9 \quad 10 \quad 11 \quad 12 \quad 13 \quad 14 \quad 15]$$

Assume the defining parameters are $a = 10$, $b = 12$, $c = 12$, and $d = 14$. Then, by Equation (2.2), the corresponding membership values are a vector of seven elements,

$$0 \quad 0 \quad 0.5 \quad 1 \quad 0.5 \quad 0 \quad 0$$

Each membership value corresponds to one element of the universe, more clearly displayed as

$$\begin{pmatrix} 0 & 0 & 0.5 & 1 & 0.5 & 0 & 0 \\ 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{pmatrix}$$

with the universe in the bottom row and the membership values in the top row. When this is impractical, in a program, the universe and the membership values can be kept in separate vectors.

As a crude rule of thumb, the continuous form is computationally intensive but requires less storage compared to the discrete form.

Possibility

According to Zadeh, a fuzzy set induces a *possibility distribution* on the universe, which implies that one can interpret the membership values as possibilities. How are then possibilities related to probabilities? First of all, probabilities must add up to one, or the area under a density curve must be one. Memberships may add up to anything (discrete case), or the area under the membership function may be anything (continuous case). Secondly, a probability distribution concerns the likelihood for the occurrence of an event, based on observations, whereas a possibility distribution (membership function) is subjective. The word 'probably' is synonymous with 'presumably', 'assumably', 'doubtless', 'likely', or 'presumptively'. The word 'possible' is synonymous with 'doable', 'feasible', 'practicable', 'viable', or 'workable'. Their relationship is best described in the sentence, 'what is probable is always possible, but not vice versa'. This is illustrated next.

Example 2.1.5 *Probability versus possibility*

(a) Consider the statement 'Hans ate x eggs for breakfast', where $x \in \mathcal{U} = \langle 1, 2, \ldots, 8 \rangle$ *(Zadeh in Zimmermann 1993). We may associate a probability distribution p by observing Hans eating breakfast for 100 days:*

$$\begin{pmatrix} 0.1 & 0.8 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

A fuzzy set expressing the grade of ease with which Hans can eat x eggs may be the following possibility distribution π*:*

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0.8 & 0.6 & 0.4 & 0.2 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

where the possibility $\pi(3) = 1$*, and the probability* $p(3) = 0.1$.

(b) Consider a universe of four car models

$$\mathcal{U} = \{Trabant, Fiat\ Uno, BMW, Ferrari\}.$$

We may associate a probability $p(x)$ *of each car model driving* 100 *mph (*161 *km/h) on a motorway, by observing the cars for* 100 *days:*

$$p(Trabant) = 0,\ p(Fiat\ Uno) = 0.1,\ p(BMW) = 0.4,\ p(Ferrari) = 0.5$$

The possibilities may be

$$\pi(Trabant) = 0,\ \pi(Fiat\ Uno) = 0.5,\ \pi(BMW) = 1,\ \pi(Ferrari) = 1$$

Notice that each possibility is at least as high as the corresponding probability.

2.2 Fuzzy Set Operations

In order to compare fuzzy sets, equality and inclusion are defined by means of membership functions.

Definition *Equality and inclusion (subset) of fuzzy sets.* Let $\mathcal{A}$ and $\mathcal{B}$ be two fuzzy sets defined on a mutual universe $\mathcal{U}$. The two fuzzy sets $\mathcal{A}$ and $\mathcal{B}$ are equal iff they have the same membership function:

$$\mathcal{A} = \mathcal{B} \equiv \mu_{\mathcal{A}}(x) = \mu_{\mathcal{B}}(x)$$

for all x. A fuzzy set $\mathcal{A}$ is a subset of (included in) a fuzzy set $\mathcal{B}$ iff the membership of $\mathcal{A}$ is less than or equal to that of $\mathcal{B}$,

$$\mathcal{A} \subseteq \mathcal{B} \equiv \mu_{\mathcal{A}}(x) \leq \mu_{\mathcal{B}}(x) \tag{2.3}$$

for all x.

In order to generate new sets from existing sets, we define two operations, which are in certain respects analogous to addition and multiplication. The (classical) union of the sets $\mathcal{X}$ and $\mathcal{Y}$, symbolized by $\mathcal{X} \cup \mathcal{Y}$ and read '$\mathcal{X}$ union $\mathcal{Y}$', is the set of all objects that are members of $\mathcal{X}$ or $\mathcal{Y}$, or both, that is,

$$\mathcal{X} \cup \mathcal{Y} \equiv \{x \mid x \in \mathcal{X} \text{ or } x \in \mathcal{Y}\}$$

Thus, by definition, $x \in \mathcal{X} \cup \mathcal{Y}$ iff x is a member of at least one of $\mathcal{X}$ and $\mathcal{Y}$. For example,

$$\{1, 2, 3\} \cup \{1, 3, 4\} = \{1, 2, 3, 4\}$$

The (classical) intersection of the sets $\mathcal{X}$ and $\mathcal{Y}$, symbolized by $\mathcal{X} \cap \mathcal{Y}$ and read '$\mathcal{X}$ intersection $\mathcal{Y}$', is the set of all objects that are members of both $\mathcal{X}$ and $\mathcal{Y}$, that is,

$$\mathcal{X} \cap \mathcal{Y} \equiv \{x \mid x \in \mathcal{X} \text{ and } y \in \mathcal{Y}\}$$

For example,

$$\{1, 2, 3\} \cap \{1, 3, 4\} = \{1, 3\}$$

The (classical) complement of a set $\mathcal{X}$, symbolized by $\overline{\mathcal{X}}$ and read 'the complement of $\mathcal{X}$' is

$$\overline{\mathcal{X}} \equiv \{x \mid x \notin \mathcal{X}\}$$

that is, the set of those members of the universe that are not members ($\notin$) of $\mathcal{X}$. Venn diagrams clearly illustrate the set operations (Figure 2.3 (a-c)).

When dealing with fuzzy sets, we must consider gradual membership. Figure 2.3 (d–f) shows an intuitively acceptable modification of the Venn diagrams. The following fuzzy set operations are defined accordingly.

Definition *Fuzzy union, intersection, and complement.* Let $\mathcal{A}$ and $\mathcal{B}$ be fuzzy sets defined on a mutual universe $\mathcal{U}$. The fuzzy union of $\mathcal{A}$ and $\mathcal{B}$ is

$$\mathcal{A} \cup \mathcal{B} \equiv \{\langle x, \mu_{\mathcal{A}\cup\mathcal{B}}(x)\rangle \mid x \in \mathcal{U} \text{ and } \mu_{\mathcal{A}\cup\mathcal{B}}(x) = \max(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x))\}$$

The fuzzy intersection of $\mathcal{A}$ and $\mathcal{B}$ is

$$\mathcal{A} \cap \mathcal{B} \equiv \{\langle x, \mu_{\mathcal{A}\cap\mathcal{B}}(x)\rangle \mid x \in \mathcal{U} \text{ and } \mu_{\mathcal{A}\cap\mathcal{B}}(x) = \min(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x))\}$$

The fuzzy complement of $\mathcal{A}$ is

$$\overline{\mathcal{A}} \equiv \left\{\left\langle x, \mu_{\overline{\mathcal{A}}}(x)\right\rangle \mid x \in \mathcal{U} \text{ and } \mu_{\overline{\mathcal{A}}}(x) = 1 - \mu_{\mathcal{A}}(x)\right\}$$

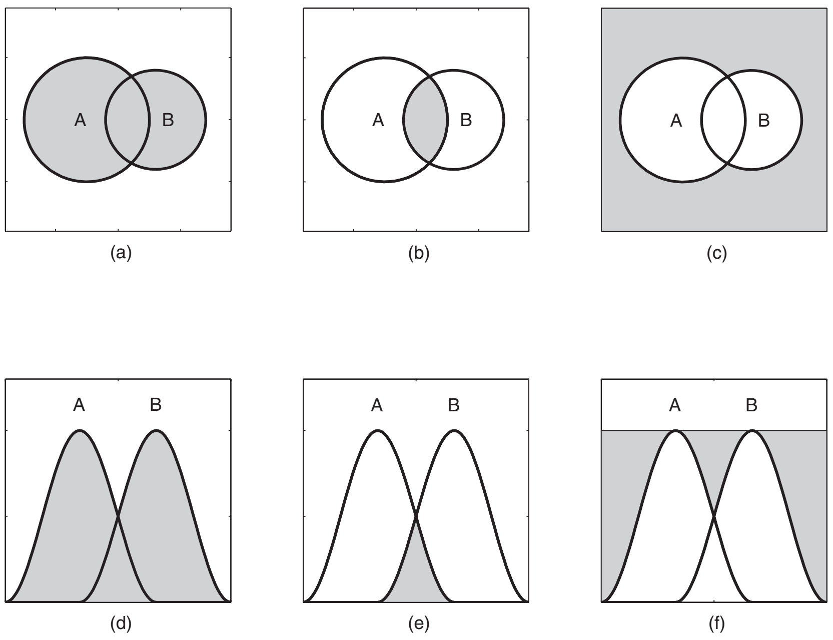


Figure 2.3: Set operations. The top row shows classical Venn diagrams; the universe is represented by the points within the rectangle, and sets by the interiors of the circles. The bottom row their fuzzy equivalents; the universal set is represented by a horizontal line at membership $\mu = 1$, and sets by membership functions. The shaded areas are: union $A \cup B$ (a, d), intersection $A \cap B$ (b, e), and complement $\overline{A \cup B}$ (c, f). (figvenn2.m)

Although the notation looks cumbersome, it is, in practice, easy to apply the fuzzy set operations max, min, and $1 - \mu$.

Example 2.2.1 *Buying a house (after Zimmermann 1993)*

A four-person family wishes to buy a house. An indication of its level of comfort is the number of bedrooms in the house. But it also wishes to have a large house. The universe $\mathcal{U} = \langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \rangle$ *is the set of houses to be considered based on the number of bedrooms. The fuzzy set Comfortable may be described as a vector* **c** *in Matlab:*

$$\boldsymbol{c} = \begin{bmatrix} 0.2 & 0.5 & 0.8 & 1 & 0.7 & 0.3 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Let **l** *describe the fuzzy set Large, defined as*

$$\boldsymbol{l} = \begin{bmatrix} 0 & 0 & 0.2 & 0.4 & 0.6 & 0.8 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The intersection of Comfortable and Large is then ***min(c,l)****,*

$$0 \quad 0 \quad 0.2 \quad 0.4 \quad 0.6 \quad 0.3 \quad 0 \quad 0 \quad 0 \quad 0$$

To interpret, five bedrooms is optimal, having the largest membership value 0.6. *It is, however, not fully satisfactory, since the membership is less than* 1. *The second best solution is* *four bedrooms, with membership* 0.4. *If the market is a buyer's market, the family may wish to wait until a better offer comes up, hoping to obtain full satisfaction (membership 1).*

The union, Comfortable or Large, is ***max(c,l)***

0.2	0.5	0.8	1	0.7	0.8	1	1	1	1

Here four bedrooms is fully satisfactory (membership 1) because it is comfortable. Also 7–10 bedrooms are satisfactory, because the house is large. If the market is a seller's market, the family may wish to buy the house, being content that at least one of the criteria is fulfilled.

If the children are about to move away from the family within the next couple of years, the parents may wish to buy a house that is Comfortable and Not Large, or ***min(c, 1-l)***

0.2	0.5	0.8	0.6	0.4	0.2	0	0	0	0

In this case, three bedrooms is satisfactory to the degree 0.8.

The example indicates how computer-aided decision support systems apply fuzzy sets. Another application area is information search, for instance, the World Wide Web search engines.

Linguistic variables

Whereas an algebraic variable takes numbers as values, a *linguistic variable* takes words or sentences as values (Zadeh in Zimmermann 1993). The name of such a linguistic variable is its *label*. The set of values that it can take is called its *term set*. Each value in the term set is a *linguistic value* or *term* defined over the universe. In short, a linguistic variable takes a linguistic value, which is a fuzzy set defined on the universe.

Example 2.2.2 *Term set Age*

Let x be a linguistic variable labelled 'Age' (Figure 2.4). Its term set $\mathcal{T}$ could be defined as

$$\mathcal{T}(Age) = \{young,\ very\ young,\ not\ very\ young,\ old,\ more\ or\ less\ old\}$$

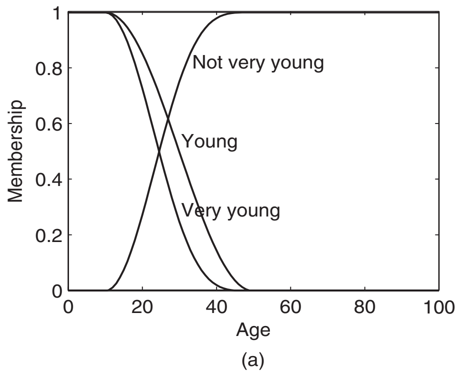
Each term is defined on the universe, for example, the integers from 0 *to* 100 *years.*

A *hedge* is a word that acts on a term and modifies its meaning. For example, in the sentence 'very near zero', the word 'very' modifies the term 'near zero'. Examples of other hedges are 'a little', 'more or less', 'possibly', and 'definitely'. In fuzzy reasoning, a hedge operates on a membership function, and the result is a membership function.

Even though it is difficult to precisely say what effect the hedge 'very' has, it does have an intensifying effect. The hedge 'more or less' (or 'morl' for short) has the opposite effect. Given a fuzzy term with the label $\mathcal{A}$ and membership function $\mu_{\mathcal{A}}(x)$ defined on the universe $\mathcal{X}$, the hedges 'very' and 'morl' are defined as

$$very\ \mathcal{A} \equiv \left\{\langle x, \mu_{very\ \mathcal{A}}(x)\rangle \mid \mu_{very\ \mathcal{A}}(x) = \mu^2_{\mathcal{A}}(x),\ x \in \mathcal{X}\right\}$$

$$morl\ \mathcal{A} \equiv \left\{\langle x, \mu_{morl\ \mathcal{A}}(x)\rangle \mid \mu_{morl\ \mathcal{A}}(x) = \mu^{\frac{1}{2}}_{\mathcal{A}}(x),\ x \in \mathcal{X}\right\}$$



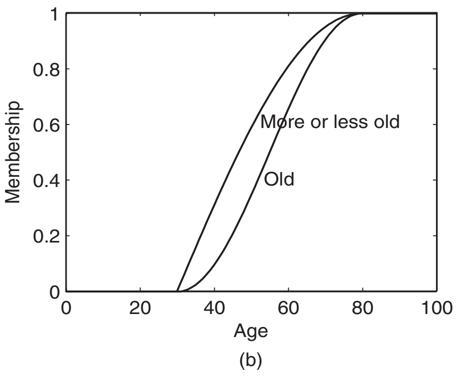


Figure 2.4: The membership functions for *very young* and *not very young* are derived from *young* (a), and the membership functions for *more or less old* from *old* (b). (figage.m)

We have applied the operations squaring and square root, but a whole family of hedges is generated by $\mu^k_{\mathcal{A}}$ or $\mu^{\frac{1}{k}}_{\mathcal{A}}$ (with integer k). For example

$$extremely\ \mathcal{A} \equiv \left\{\langle x, \mu_{very\ \mathcal{A}}(x)\rangle \mid \mu_{very\ \mathcal{A}}(x) = \mu^3_{\mathcal{A}}(x),\ \ x \in \mathcal{X}\right\}$$

$$slightly\ \mathcal{A} \equiv \left\{\langle x, \mu_{very\ \mathcal{A}}(x)\rangle \mid \mu_{very\ \mathcal{A}}(x) = \mu^{\frac{1}{3}}_{\mathcal{A}}(x),\ \ x \in \mathcal{X}\right\}$$

A derived hedge is, for example, *somewhat* $\mathcal{A}$, which is defined as *morl* $\mathcal{A}$ *and not slightly* $\mathcal{A}$. For the special case where $k = 2$, the operation μ^2 is *concentration* and $\mu^{\frac{1}{2}}$ is *dilation*. With $k = \infty$ the hedge $\mu^k_{\mathcal{A}}$ could be named *exactly*, because it would suppress all memberships lower than 1.

Example 2.2.3 *Very on a discrete membership function*
Assume a discrete universe $\mathcal{U} = \langle 0, 20, 40, 60, 80\rangle$ *of ages. In Matlab we can assign*

$$\boldsymbol{u} = [0 \quad 20 \quad 40 \quad 60 \quad 80]$$

and

$$\boldsymbol{young} = [1 \quad .6 \quad .1 \quad 0 \quad 0]$$

The discrete membership function for the set 'very young' is young.^2,

```
1   0.36   0.01   0   0
```

The notation '.^' is Matlab notation for the power operator. The set 'very very young' is, by repeated application, young.^4,

```
1   0.13   0   0   0
```

The derived sets inherit the universe of the primary set.

A *primary term* is a term that must be defined a priori, for example ‘young’ and ‘old’ in Figure 2.4, whereas the sets ‘very young’ and ‘not very young’ are modified sets. The primary terms can be modified by negation (‘not’) or hedges (‘very’, ‘more or less’), and the resulting sets can be connected using connectives (‘and’, ‘or’, ‘implies’, ‘equals’). Long sentences can be built using this vocabulary, and the result is still a membership function.

Relations

In mathematics, the word *relation* is used in the sense of relationship. For example, the predicates: x is less than y, or y is a function of x. A *binary relation* $\mathcal{R}$ is a set of ordered pairs. We write $x\mathcal{R}y$ for ‘x is related to y.’ There are established symbols for various relations, for example, $x = y$, $x < y$. One simple relation is the set of all pairs $\langle x, y\rangle$ such that x is a member of a set $\mathcal{X}$ and y is a member of a set $\mathcal{Y}$. This is the (classical) *Cartesian product* of $\mathcal{X}$ and $\mathcal{Y}$:

$$\mathcal{X} \times \mathcal{Y} \equiv \{\langle x, y\rangle \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$$

In fact, any binary relation $x\mathcal{R}y$ is a subset of the Cartesian product $\mathcal{X} \times \mathcal{Y}$, and we can think of those instances of $\mathcal{X} \times \mathcal{Y}$ that are members of $\mathcal{R}$ as having membership 1.

By analogy, a binary *fuzzy relation* consists of pairs $\langle x, y\rangle$ with an associated fuzzy membership value.

Definition *Fuzzy binary relation.* Let $\mathcal{A}$ and $\mathcal{B}$ be fuzzy sets defined on $\mathcal{X}$ and $\mathcal{Y}$ respectively. Then the fuzzy set in $\mathcal{X} \times \mathcal{Y}$ with the membership function

$$\mathcal{R} \equiv \{\langle\langle x, y\rangle, \mu_{\mathcal{R}}(x, y)\rangle \mid (x, y) \in \mathcal{X} \times \mathcal{Y}\}$$

is a binary fuzzy relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$.

For example, given $\mathcal{X} = \mathcal{Y} = \{1, 2, 3\}$, we can set up a relation ‘approximately equal’ between all pairs of the three numbers, most clearly displayed in a tabular arrangement:

		$\mathcal{Y}$		
		1	2	3
	1	1	0.8	0.3
$\mathcal{X}$	2	0.8	1	0.8
	3	0.3	0.8	1

It is straightforward to generalize the relations to n-ary relations ($n > 2$).

In the fuzzy Cartesian product, each object is defined by a pair: the object, which itself is a pair, and its membership.

Definition *Fuzzy Cartesian product.* Let $\mathcal{A}$ and $\mathcal{B}$ be fuzzy sets defined on $\mathcal{X}$ and $\mathcal{Y}$ respectively. Then the fuzzy set in $\mathcal{X} \times \mathcal{Y}$ with the membership function

$$\mathcal{A}\times\mathcal{B} \equiv \left\{\langle\langle x, y\rangle, \mu_{\mathcal{A}\times\mathcal{B}}(x, y)\rangle \mid x \in \mathcal{X}, y \in \mathcal{Y}, \mu_{\mathcal{A}\times\mathcal{B}}(x, y) = \min\left(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(y)\right)\right\}$$

is the Cartesian product of $\mathcal{A}$ and $\mathcal{B}$.

For example, assume $\mathcal{X}$ and $\mathcal{Y}$ are given as above, and $\mu_{\mathcal{A}}(x_i) = \langle 0, 0.5, 1\rangle$, with $i = 1, 2, 3$, and $\mu_{\mathcal{B}}(y_j) = \langle 1, 0.5, 0\rangle$, with $j = 1, 2, 3$. Then $\mathcal{A} \times \mathcal{B}$ can be arranged as a two-dimensional fuzzy set:

$$\begin{array}{cc|c|c|c|}
 & & & \mathcal{B} & \\
 & & 1 & 0.5 & 0 \\
\hline
 & 0 & 0 & 0 & 0 \\
\hline
\mathcal{A} & 0.5 & 0.5 & 0.5 & 0 \\
\hline
 & 1 & 1 & 0.5 & 0 \\
\hline
\end{array}$$

The element at row i and column j is the intersection of $\mu_{\mathcal{A}}(x_i)$ and $\mu_{\mathcal{B}}(y_j)$. Again we note that a membership $\mu_{\mathcal{A}\times\mathcal{B}}(x_i, y_j)$ is associated with each object $\langle x_i, y_j\rangle$, whereas the classical Cartesian product consists of objects $\langle x_i, y_j\rangle$ only.

In order to understand how relations can be combined, let us look at an example from the cartoon Donald Duck.

Example 2.2.4 *Family resemblance*

Assume that Donald Duck's nephew Huey resembles nephew Dewey to the grade 0.8, and Huey resembles nephew Louie to the grade 0.9. We have therefore a relation between two subsets of the nephews in the family. This is conveniently represented in a matrix with one row and two columns (and additional headings):

$$\boldsymbol{R}_1 = \begin{array}{c|c|c|}
 & \textit{Dewey} & \textit{Louie} \\
\hline
\textit{Huey} & 0.8 & 0.9 \\
\hline
\end{array}$$

Let us assume another relation between nephews Dewey and Louie on the one side, and uncle Donald on the other, a matrix with two rows and one column,

$$\boldsymbol{R}_2 = \begin{array}{c|c|}
 & \textit{Donald} \\
\hline
\textit{Dewey} & 0.5 \\
\hline
\textit{Louie} & 0.6 \\
\hline
\end{array}$$

We wish to find out how much Huey resembles Donald by combining the information in the two matrices:

(i) Huey resembles Dewey ($\boldsymbol{R}_1(1, 1) = 0.8$), and Dewey in turn resembles Donald ($\boldsymbol{R}_2(1, 1) = 0.5$); or

(ii) Huey resembles Louie ($\boldsymbol{R}_1(1, 2) = 0.9$), and Louie in turn resembles Donald ($\boldsymbol{R}_2(2, 1) = 0.6$).

Assertion (i) contains two relationships combined by 'and'; it seems reasonable to take the intersection. With our previous definition, this corresponds to choosing the smallest membership value for the (transitive) Huey–Donald relationship, or $\min(0.8, 0.5) = 0.5$. *Similar results can be obtained with statement (ii). Thus from (i) and (ii), respectively, we deduce the following:*

(iii) Huey resembles Donald to the degree 0.5*; or*

(iv) Huey resembles Donald to the degree 0.6.

Although the results in (iii) and (iv) differ, we are equally confident of either result; we have to choose either one or the other, so it seems reasonable to take the union. With our previous definition, this corresponds to choosing the largest membership value, or max (0.5, 0.6) = 0.6. *Thus, the answer is that Huey resembles Donald to the degree* 0.6.

Mathematically speaking, this was an example of *composition* of relations.

Definition *Fuzzy relational composition.* Let $\mathcal{R}$ and $\mathcal{S}$ be two fuzzy relations defined on $\mathcal{X} \times \mathcal{Y}$ and $\mathcal{Y} \times \mathcal{Z}$, respectively. Then the fuzzy set in $\mathcal{X} \times \mathcal{Z}$ with the membership function

$$\mathcal{R} \circ \mathcal{S} \equiv \left\{ \left\langle \langle x, z \rangle , \bigcup_y \mu_{\mathcal{R}} (x, y) \cap \mu_{\mathcal{S}} (y, z) \right\rangle \mid x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z} \right\}$$

is the composition of $\mathcal{R}$ with $\mathcal{S}$.

When $\mathcal{R}$ and $\mathcal{S}$ are expressed as matrices $\boldsymbol{R}$ and $\boldsymbol{S}$, the composition is equivalent to an *inner product*. The inner product is similar to an ordinary matrix (dot) product, except that multiplication and summation are replaced by other functions. Suppose $\boldsymbol{R}$ is an $m \times p$ matrix and $\boldsymbol{S}$ is a $p \times n$ matrix. Then the inner $\cup - \cap$ product (read 'cup-cap product') is an $m \times n$ matrix $\boldsymbol{T} = (t_{ij})$ whose ijth entry is obtained by combining the ith row of $\boldsymbol{R}$ with the jth column of $\boldsymbol{S}$, such that

$$t_{ij} = \left(r_{i1} \cap s_{1j}\right) \cup \left(r_{i2} \cap s_{2j}\right) \cup \ldots \cup \left(r_{ip} \cap s_{pj}\right) = \bigcup_{k=1}^{p} r_{ik} \cap s_{kj} \tag{2.4}$$

The notation looks unwieldy, but the operation is essentially a matter of combining rows with columns successively, as the next example shows.

Example 2.2.5 *Inner product*

For the matrices $\boldsymbol{R}_1$ *and* $\boldsymbol{R}_2$ *above, the inner product yields*

$$\boldsymbol{R}_1 \circ \boldsymbol{R}_2 = \begin{pmatrix} 0.8 & 0.9 \end{pmatrix} \circ \begin{pmatrix} 0.5 \\ 0.6 \end{pmatrix} = (0.8 \cap 0.5) \cup (0.9 \cap 0.6) = 0.5 \cup 0.6 = 0.6$$

which agrees with the previous result.

With our previous definitions of the set operations, composition is specifically called *max–min composition* (Zadeh in Zimmermann 1993). If the *min* operation is replaced by * for multiplication, it is called *max–star composition*.

2.3 Fuzzy Logic

The study of logic began as a study of language in arguments and persuasion, and it can be used to judge the correctness of a chain of reasoning – in a mathematical proof for instance. The goal of the theory is to reduce principles of reasoning to a code. Similar to a programming language, logic builds on truth-values (constants), connectives (operators), and rules of inference (programming constructs).

Truth-values

The 'truth' or 'falsity' assigned to a proposition is its *truth-value*. In two-valued logic, a proposition can be either true (1) or false (0), that is, the truth values belong to the set {0, 1}. The binary logical relations called *connectives* can be defined by means of $2^{2^2} = 16$ possible tables of truth-values, the *truth-tables*.

A possible extension is to include an intermediate truth-value 'undecided' (0.5). The result is a three-valued logic. If one tries to identify all possible binary connectives, it would result in $3^{2^3} = 729$ connectives, which is impractical. There are, nevertheless, a multitude of three-valued logics (for instance Lukasiewicz logic in Nguyen and Walker 2000), differing in the specification of truth-tables and the implication connective. Extensions to more truth-values, finite in number, lead to multi-valued logics (Rescher in Nguyen and Walker 2000).

Fuzzy logic extends the range of truth-values to the continuous interval [0, 1] of real numbers (Nguyen and Walker 2000). In *fuzzy logic* a proposition may be true or false, or any intermediate truth-value. For instance, the sentence 'John is tall' may assume an intermediate truth-value depending on the circumstances.

Originally, Zadeh interpreted a truth-value in fuzzy logic as a fuzzy set, for instance, *Very true* (Zadeh 1988). Thus, Zadeh based fuzzy (linguistic) logic on the treatment of *Truth* as a linguistic variable that takes words or sentences, rather than numbers, as values (Zadeh 1975). Our approach differs, as it is built on scalar truth-values rather than membership functions.

Classical connectives

In daily conversation and mathematics, sentences are connected with the words *and*, *or*, *if–then* (or *implies)*, and *if and only if.* These are called *connectives.* A sentence that is modified by the word 'not' is called the *negation* of the original sentence. The word 'and' is used to join two sentences to form the *conjunction* of the two sentences. The word 'or' is used to join two sentences to form the *disjunction* of the two sentences. From two sentences we may construct one, of the form 'If . . . then . . .'; this is called an *implication.* The sentence following 'If' is the *antecedent*, and the sentence following 'then' is the *consequent.* Other idioms (unclear phrases that must be learnt as a whole unit) that we shall regard as having the same meaning are 'p implies q', 'p only if q', 'q if p', and so on.

Letters and special symbols make the connective structure stand out. Our choice of symbols is

$\neg$	for 'not'
$\wedge$	for 'and'
$\vee$	for 'or'

$\Rightarrow$ for 'if–then' (implication)
$\Leftrightarrow$ for 'if and only if' (equivalence)

A *propositional variable*, denoted by a letter, takes truth-values as values. An assertion that contains at least one propositional variable is called a *propositional formula*. The main difference between *proposition* and *propositional formula* is that every proposition has a truth-value, whereas a propositional formula is an assertion whose truth-value cannot be determined until propositions are substituted for its propositional variables. But when no confusion results, we will refer to propositional formulae as propositions.

The next example illustrates how the symbolic forms expose the underlying logical structure.

Example 2.3.1 *Baseball betting (Stoll 1979)*
Consider the assertion about four baseball teams: if either the Pirates or the Cubs lose and the Giants win, then the Dodgers will be out of first place, and I will lose a bet. Since it is an implication, it may be symbolized in the form $r \Rightarrow s$. The antecedent is composed from the three sentences p *(The Pirates lose),* c *(The Cubs lose), and* g *(The Giants win). The consequent is the conjunction of* d *(The Dodgers will be out of first place) and* b *(I will lose a bet). The original sentence may thus be represented by $((p \vee c) \wedge g) \Rightarrow (d \wedge b)$.*

A *truth-table* summarizes the possible truth-values of an assertion. Take for example the truth-table for the two-valued formula $p \vee q$. The truth-table below (left) lists all possible combinations of truth-values – the Cartesian product – of the variables p and q in the two leftmost columns. The rightmost column holds the truth-values of the formula. Alternatively, the truth-table can be rearranged into a two-dimensional array, a so-called Cayley table (below, right).

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

is equivalent to

Or

$$\begin{array}{c|c|c|l} \multicolumn{3}{c}{p \vee q} & \\ & 0 & 1 & \rightarrow q \\ \hline 0 & 0 & 1 & \\ \hline 1 & 1 & 1 & \\ \hline \downarrow & & & \\ p & & & \end{array}$$

Along the vertical axis in the Cayley table, symbolized by arrow $\downarrow$, are the possible values 0 and 1 of the first argument p. Along the horizontal axis, symbolized by arrow $\rightarrow$, are the possible values 0 and 1 of the second argument q. Above the table, is the symbolic form $p \vee q$ for disjunction. At the intersection of row i and column j (only counting the inside of the box) is the truth-value of the expression $p_i \vee q_j$. By inspection, one entry renders $p \vee q$ false, while three entries render $p \vee q$ true. Truth-tables for binary connectives are thus given by two-by-two matrices. A total of 16 such tables can be constructed, and each is associated with a connective.

We can derive the truth-table for 'Nand' ('not and') from 'or'. By the definition $(\neg p) \vee (\neg q)$ we negate each variable of the previous truth-table, which is equivalent to reversing the axes,

$$\begin{array}{c}\text{Nand}\\ (\neg p) \vee (\neg q)\end{array}$$

$$\begin{array}{c|c|c|l} & 0 & 1 & \to q \\ \cline{2-3} 0 & 1 & 1 & \\ \cline{2-3} 1 & 1 & 0 & \\ \cline{2-3} \downarrow & \multicolumn{3}{l}{} \\ p & \multicolumn{3}{l}{} \end{array}$$

From ‘Nand’ (‘not and’) we derive ‘And’ by negating the entries inside the table (‘not not and’ = ‘And’),

$$\begin{array}{c}\text{And}\\ \neg\, ((\neg p) \vee (\neg q))\end{array}$$

$$\begin{array}{c|c|c|l} & 0 & 1 & \to q \\ \cline{2-3} 0 & 0 & 0 & \\ \cline{2-3} 1 & 0 & 1 & \\ \cline{2-3} \downarrow & \multicolumn{3}{l}{} \\ p & \multicolumn{3}{l}{} \end{array}$$

Notice that it was possible to define ‘and’ by means of ‘or’ and ‘not’, rather than assume its definition given as an axiom.

By means of ‘or’ and ‘not’ we can proceed to define ‘implication’. Classical logic defines implication $\neg p \vee q$, which is called *material implication*. We negate the p-axis of the ‘or’ table, which is equivalent to reversing the axis,

$$\begin{array}{c}\text{Implication}\\ \neg p \vee q\end{array}$$

$$\begin{array}{c|c|c|l} & 0 & 1 & \to q \\ \cline{2-3} 0 & 1 & 1 & \\ \cline{2-3} 1 & 0 & 1 & \\ \cline{2-3} \downarrow & \multicolumn{3}{l}{} \\ p & \multicolumn{3}{l}{} \end{array}$$

Equivalence is taken to mean $(p \Rightarrow q) \wedge (q \Rightarrow p)$, which is equivalent to the conjunction of each entry of the implication table with the elements of the transposed table, element by element:

$$\begin{array}{c}\text{Equivalence}\\ (p \Rightarrow q) \wedge (q \Rightarrow p)\end{array}$$

$$\begin{array}{c|c|c|l} & 0 & 1 & \to q \\ \cline{2-3} 0 & 1 & 0 & \\ \cline{2-3} 1 & 0 & 1 & \\ \cline{2-3} \downarrow & \multicolumn{3}{l}{} \\ p & \multicolumn{3}{l}{} \end{array}$$

It is possible to evaluate, in principle at least, a formula by an exhaustive test of all combinations of truth-values of the variables, as the next example illustrates.

Example 2.3.2 *Array-based logic*
In the baseball example, we derived the relation $((p \vee c) \wedge g) \Rightarrow (d \wedge b)$. The proposition contains five variables, and each variable can take two truth-values. The total number of possible combinations is therefore $2^5 = 32$. Only 23 are legal, in the sense that the proposition is true for these combinations, and $32 - 23 = 9$ cases are illegal, in the sense that the proposition is false for those combinations. If we are interested only in the legal combinations

for which 'I win the bet' ($b = 0$)*, then the following table is obtained:*

p	c	g	d	b
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
1	0	0	0	0
1	0	0	1	0
1	1	0	0	0
1	1	0	1	0

There are thus 10 *winning outcomes out of* 32 *possible.*

An exhaustive test of all possible combinations is the idea behind array-based logic *(Franksen 1979).*

Fuzzy connectives

We would like to define truth-tables for fuzzy logic connectives by analogy, but we face one problem: the truth-values are defined on a continuous interval, that is, they are uncountable. We have to make sure that whatever identity we develop is valid for *all* truth values in [0, 1].

We shall assume a universe of truth values

$$\mathcal{U} = \{0, u, 0.5, v, 1\}$$

with the special constraints that

$$0 < u < 0.5$$
$$0.5 < v < 1$$
$$v = \neg u$$

This is a five-valued logic, but the variable u can take any intermediate value between false and undecided, and at the same time v varies between undecided and true. Thus all truth values are accounted for.

If we start again by defining negation and disjunction, we can derive the truth-tables of other connectives from that point of departure. Let us define *negation* as set complement, that is,

$$\neg p \equiv 1 - p.$$

By this definition $v = 1 - u$, and therefore $u = 1 - v$, and we have

$$v = \neg u$$
$$u = \neg v$$

as specified. By insertion, we can immediately ascertain the *law of involution*: $\neg(\neg u) = u$.

If we define *disjunction* as set union, that is,

$$p \vee q \equiv \max(p, q), \tag{2.5}$$

we can build the truth-table for the fuzzy connective 'or':

Or

$p \vee q$

$p \downarrow$ \ $\rightarrow$q	0	u	0.5	v	1
0	0	u	0.5	v	1
u	u	u	0.5	v	1
0.5	0.5	0.5	0.5	v	1
v	v	v	v	v	1
1	1	1	1	1	1

Like before, the p-axis is vertical and the q-axis horizontal. At the intersection of row i and column j (only regarding the inside of the box) we have the value of the expression $\max(p_i, q_j)$ in accordance with Equation (2.5). When looking for definitions of fuzzy connectives, we will require that such connectives should agree with their classical counterparts for the truth-domain $\{0, 1\}$. In terms of truth-tables, the values in the four corners of the fuzzy Cayley table should agree with the Cayley table for the classical connective.

The truth-table for 'Nand' is derived from 'Or' by the definition $(\neg p) \vee (\neg q)$ by negating the variables. This is equivalent to reversing the axes in the Cayley table. Moving further, 'And' is the negation of the entries in the truth-table for 'Nand'. Thus we have the following:

Nand

$(\neg p) \vee (\neg q)$

$p \downarrow$ \ $\rightarrow$q	0	u	0.5	v	1
0	1	1	1	1	1
u	1	v	v	v	v
0.5	1	v	0.5	0.5	0.5
v	1	v	0.5	u	u
1	1	v	0.5	u	0

And

$\neg((\neg p) \vee (\neg q))$

$p \downarrow$ \ $\rightarrow$q	0	u	0.5	v	1
0	0	0	0	0	0
u	0	u	u	u	u
0.5	0	u	0.5	0.5	0.5
v	0	u	0.5	v	v
1	0	u	0.5	v	1

It is reassuring to observe that even though the truth-table for 'And' is derived from the truth-table for 'Or', the truth-table for 'And' is identical to one generated using the min operation, set intersection.

Example 2.3.3 *Fuzzy baseball*

The baseball example illustrates the difference three-valued logic makes. The proposition

$$((p \vee c) \wedge g) \Rightarrow (d \wedge b)$$

contains five variables, and now each can take three truth-values. There are $3^5 = 243$ *possible combinations;* 148 *of these are legal in the sense that the proposition is true (truth-value* 1*). If we are interested again in the legal combinations for which 'I win the bet'* ($b = 0$)*, then there are* 33 *winning outcomes out of* 148*. Instead of listing all of these, we show one for illustration:*

$$(p, c, g, d, b) = (0.5, 0.5, 0, 1, 0)$$

With two-valued logic, we found 10 *winning outcomes out of the* 32 *possible.*

The example indicates that fuzzy logic provides more solutions, compared to two-valued logic, and requires more computational effort.

2.4 Fuzzy Implication

The *implication*, however, has always troubled fuzzy logicians. If we define it as *material implication*, $\neg p \vee q$, it causes several useful logical laws to break down. We must make a design choice at this point in order to proceed with the definition of implication and equivalence. The choice is to select which logical laws must hold.

Some laws known from two-valued logic do not hold in fuzzy logic. Take for instance the formula

$$p \vee \neg p \Leftrightarrow 1 \tag{2.6}$$

which is equivalent to the law of the excluded middle. Testing with the truth-value $p = 0.5$ (fuzzy logic), the left-hand side of the equivalence symbol $\Leftrightarrow$ yields

$$0.5 \vee \neg 0.5 = \max(0.5, 1 - 0.5) = 0.5.$$

This is different from the right-hand side, and thus the law of the excluded middle does not hold in fuzzy logic.

If a proposition is true with a truth-value of 1, for *any* combination of truth-values assigned to the variables, we shall say it is *valid*. Such a proposition is a *tautology*. If the proposition is true for some, but not all combinations, we shall say it is *satisfiable*. Thus Equation (2.6) is satisfiable, since it is true in two-valued logic, but not in three-valued logic.

One tautology that we definitely wish to apply in fuzzy logic applications is

$$\text{Tautology 1:} \quad \big[p \wedge (p \Rightarrow q)\big] \Rightarrow q \tag{2.7}$$

or, in words, if p, and p implies q, then q. We need it later in connection with the *modus ponens* rule of inference. Another tautology that we definitely wish to apply in fuzzy logic applications is the transitive relationship

$$\text{Tautology 2:} \quad \big[(p \Rightarrow q) \wedge (q \Rightarrow r)\big] \Rightarrow (p \Rightarrow r) \tag{2.8}$$

or, in words, from left to right, if p implies q, which in turn implies r, then p implies r. Whether these propositions are valid in fuzzy logic depends on how we define the connectives, or rather, we must define the connectives, implication in particular, such that those propositions become valid.

Many researchers have proposed implication connectives (e.g. Zadeh 1973, Mizumoto *et al.* 1979, Fukami *et al.* 1980, Wenstøp 1980; see also the survey by Lee 1990). In fact Kiszka *et al.* (1985) list 72 alternatives to choose from and Driankov *et al.* (1996) test nine implications. Nguyen and Walker (2000) define classes of implications based on three basic forms.

One candidate, not necessarily the best, is the so-called *Gödel implication*. It can be defined as

$$p \Rightarrow q \equiv (p \leq q) \vee q \tag{2.9}$$

and the truth-table is

Implication

$(p \leq q) \vee q$

$p \downarrow$ \ $\rightarrow q$	0	u	0.5	v	1
0	1	1	1	1	1
u	0	1	1	1	1
0.5	0	u	1	1	1
v	0	u	0.5	1	1
1	0	u	0.5	v	1

Equivalence

$(p \Rightarrow q) \wedge (q \Rightarrow p)$

$p \downarrow$ \ $\rightarrow q$	0	u	0.5	v	1
0	1	0	0	0	0
u	0	1	0	0	0
0.5	0	0	1	0	0
v	0	0	0	1	0
1	0	0	0	0	1

(2.10)

The truth-table for equivalence ($\Leftrightarrow$) is derived from implication and conjunction, once it is agreed that $p \Leftrightarrow q$ is the same as $(p \Rightarrow q) \wedge (q \Rightarrow p)$. Further truth-tables can be built on the previously defined operations, for example exclusive or $\neg(p \Rightarrow q)$ and 'nor' = 'not or', $\neg(p \vee q)$.

It is straightforward to test whether tautologies 1 and 2 are valid, because it is possible to perform an exhaustive test of all combinations of truth-values of the variables.

Example 2.4.1 *Proof of tautology 1*

Tautology 1 is

$$[p \wedge (p \Rightarrow q)] \Rightarrow q$$

Since the proposition contains two variables p and q and each variable can take five truth-values, there will be $5^2 = 25$ *possible combinations to test. The truth-table has* 25 *rows (Table 2.1). Columns 1 and 2 are the input combinations of p and q. Column 3 is the result of Gödel implication* $(p \leq q) \vee q$*, and column 4 is the left-hand side of tautology 1.*

Since the rightmost column is all ones, the proposition is a tautology.

In fact, the example suggests a new tautology. Compare the truth-values for $[p \wedge (p \Rightarrow q)]$ (column 4 in Table 2.1) with the truth-table for conjunction – they are identical. We thus have

$$\text{Tautology 3:} \quad \big[p \wedge (p \Rightarrow q)\big] \Leftrightarrow p \wedge q \tag{2.11}$$

This is in fact our motivation for choosing the Gödel implication. A test (not shown) with our definitions of $\wedge$ and $\Rightarrow$ confirms that all three tautologies are valid.

Since implication can be defined in many possible ways, one has to determine a design criterion first, namely the tautologies, before choosing a proper definition for the implication connective. The array approach reduces the proof of any tautology to a test that can be programmed on a computer.

Table 2.1: Proof of tautology 1. The tautology is valid since the rightmost column contains purely 1s.

p	q	$p \Rightarrow q$	$[p \wedge (p \Rightarrow q)]$	$[p \wedge (p \Rightarrow q)] \Rightarrow q$
0	0	1	0	1
0	u	1	0	1
0	0.5	1	0	1
0	v	1	0	1
0	1	1	0	1
u	0	0	0	1
u	u	1	u	1
u	0.5	1	u	1
u	v	1	u	1
u	1	1	u	1
0.5	0	0	0	1
0.5	u	0	u	1
0.5	0.5	1	0.5	1
0.5	v	1	0.5	1
0.5	1	1	0.5	1
v	0	0	0	1
v	u	0	u	1
v	0.5	0	0.5	1
v	v	1	v	1
v	1	1	v	1
1	0	0	0	1
1	u	0	u	1
1	0.5	0	0.5	1
1	v	0	v	1
1	1	1	1	1

Example 2.4.2 *Mamdani 'implication'*
The so-called Mamdani 'implication' *(Mamdani 1977), is often used in fuzzy control. Let $\mathcal{A}$ and $\mathcal{B}$ be fuzzy sets defined on $\mathcal{X}$ and $\mathcal{Y}$ respectively. Then the Mamdani 'implication' is a fuzzy set in $\mathcal{X} \times \mathcal{Y}$ with the membership function*

$$\{\langle\langle x, y\rangle, \mu_{\mathcal{A}' \Rightarrow' \mathcal{B}}(x, y)\rangle \mid x \in \mathcal{X}, y \in \mathcal{Y}, \mu_{\mathcal{A}' \Rightarrow' \mathcal{B}}(x, y) = \min(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(y))\}$$

Notice the definition is similar to the definition of the fuzzy Cartesian product. Its Cayley table, in two-valued logic, is

Mamdani 'implication'

$$\min(p, q)$$

$$\begin{array}{c|c|c|l} & 0 & 1 & \rightarrow q \\ \hline 0 & 0 & 0 & \\ \hline 1 & 0 & 1 & \\ \hline \downarrow & & & \\ p & & & \end{array}$$

Only one out of four truth-values matches the truth-table for two-valued implication; therefore it is not an implication.

The logic just employed, is an example of a disproof by the rule of modus tollens. *The argument is as follows: (1) Mamdani 'implication' is an implication; (2) all implications are valid in two-valued logic; and (3) the truth-table shows it is not true, and therefore Mamdani 'implication' is not an implication. It also illustrates the principle of falsification: proving a hypothesis can be very difficult, but a single counterexample can falsify it (after Karl Popper).*

It would be accurate to call it Mamdani 'inference', according to the formal derivation at the end of this chapter, and we shall do so from here onwards.

2.5 Rules of Inference

Logic provides principles of reasoning, by means of *inference,* the drawing of conclusions from assertions. The verb 'to infer' means to conclude from evidence, deduce, or to have as a logical consequence. *Rules of inference* specify conclusions drawn from assertions known or assumed to be true.

One such rule of inference is *modus ponens*. It is often presented in the form of an *argument*:

$$\begin{array}{l} P \\ P \Rightarrow Q \\ \hline Q \end{array}$$

In words, if (1) P is known to be true, and (2) we assume that $P \Rightarrow Q$ is true, then (3) Q must be true. Restricting for a moment to two-valued logic, we observe from the truth-table for implication,

$$\begin{array}{c} \text{Implication} \\ p \Rightarrow q \\ \begin{array}{c|c|c|l} & 0 & 1 & \rightarrow q \\ \hline 0 & 1 & 1 & \\ \hline 1 & 0 & 1 & \\ \hline \downarrow & & & \\ p & & & \end{array} \end{array},$$

that whenever $P \Rightarrow Q$ and P are true, then so is Q; assuming P true takes us to the second row, which contains only a single 1 leaving Q true as the only solution. In such an argument, the assertion P is the *premise*, the assertion $P \Rightarrow Q$ is the *implication*, and the assertion below the line is the *conclusion*. Notice that the premise and the implication are considered as true and only true.

On the other hand, underlying the modus ponens is tautology 1, which expresses the same, but for *all* truth-values. Modus ponens is thus valid in fuzzy logic, if tautology 1 is valid in fuzzy logic.

Example 2.5.1 *Four useful rules of inference*
There are several useful rules of inference, which can be represented by tautological forms. Four such rules are presented here using examples.

(a) Modus ponens. *Its tautological form is*

$$\left[p \wedge (p \Rightarrow q)\right] \Rightarrow p$$

Let p stand for 'altitude sickness,' and let $p \Rightarrow q$ stand for 'altitude sickness causes a headache'. If it is known that John suffers from altitude sickness, p is true, and $p \Rightarrow q$ is assumed to be true in this illustration, then the conclusion q is true, that is, John has a headache.

(b) Modus tollens. *Its tautological form is*

$$\left[\neg q \wedge (p \Rightarrow q)\right] \Rightarrow \neg p$$

Let p and q be as in (a). Thus, if John does not have a headache, then we may infer that John does not suffer from altitude sickness.

(c) Disjunctive syllogism. *Its tautological form is*

$$\left[(p \vee q) \wedge \neg p\right] \Rightarrow q$$

Let p stand for 'altitude sickness' as before, but let q stand for 'dehydration'. Thus, if it is known for a fact that John's headache is due to either altitude sickness or dehydration, and it is not altitude sickness, then we may infer that John suffers from dehydration.

(d) Hypothetical syllogism. *Its tautological form is tautology 2:*

$$\left[(p \Rightarrow q) \wedge (q \Rightarrow r)\right] \Rightarrow (p \Rightarrow r)\,.$$

Let p stand for 'high altitude and fast ascent', let q stand for 'altitude sickness', and let r stand for 'a headache'. Further, assume that high altitude and fast ascent together cause altitude sickness, and in turn that altitude sickness causes a headache. Thus, we may infer that John will get a headache in high altitude if John ascends fast.

Testing with our previous definitions of $\neg$, $\wedge$, $\vee$, and $\Rightarrow$ shows that (a), (c), and (d) are valid, while (b) is only satisfiable. Further logical relationships (31 in total) have been tested earlier (Jantzen 1995).

Provided the tautological forms are valid in fuzzy logic, the inference rules may be applied in fuzzy logic as well.

The inference mechanism in modus ponens can be generalized. The pattern is as follows: given a relation $\mathcal{R}$ connecting logical variables p and q, we infer the possible values of q given a particular instance of p; the procedure is similar to finding the value y_1 of a function $y = f(x)$ given an input value x_1 The modus ponens inference mechanism is recast into arrays, to make it operational, that is, executable by a computer. The next example shows the mechanism.

Example 2.5.2 *Array-based modus ponens*

Switching to vector–matrix representation, with $\mathbf{p}$ *as a (column) vector and* $\boldsymbol{R}$ *as a two-dimensional truth-table, with the p-axis vertical, the inference is defined*

$$\mathbf{q}^t = \mathbf{p}^t \circ \boldsymbol{R}$$

The operation $\circ$ is an inner $\vee - \wedge$ product (read 'or–and product'). The $\wedge$ operation is the same as in $p \wedge (p \Rightarrow q)$ and the $\vee$ operation along the columns yields what can possibly *be implied about q, confer the rightmost implication in $\left[p \wedge (p \Rightarrow q)\right] \Rightarrow p$. Assuming p is true corresponds to setting*

$$\mathbf{p} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

But the scheme is more general, because we could also assume **p** *is false, compose with* **R**, *and study what can be inferred about* **q**. *Take for instance modus ponens. Thus*

$$\boldsymbol{R} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

which is the truth-table for $p \Rightarrow q$. Assigning **p** *as above,*

$$\mathbf{q}^t = \mathbf{p}^t \circ \boldsymbol{R} = \begin{pmatrix} 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

The outcome $\mathbf{q}^t$ is a truth-vector pointing at q true as the only possible conclusion, as expected.

Trying $\mathbf{p} = (1\ 0)^t$ yields

$$\mathbf{q}^t = \mathbf{p}^t \circ \boldsymbol{R} = \begin{pmatrix} 1 & 0 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

Thus q could be anything, true or false, as expected.

The inference could even proceed in the reverse direction, from q to p, but then we must compose from the right side of $\boldsymbol{R}$ *to match the axes. Assume for instance q is true, or $\mathbf{q} = (1\ 0)^t$, then*

$$\mathbf{p} = \boldsymbol{R} \circ \mathbf{q} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

To interpret, if q is false and $p \Rightarrow q$, then p is false (modus tollens).

The array-based inference mechanism is even more general, because $\boldsymbol{R}$ *can be any dimension n ($n > 0$ and integer). Given the values of $n - 1$ variables, the possible outcomes of the remaining variable can be inferred by an n-dimensional inner product. Furthermore, given the values of $n - d$ variables (d integer and $0 < d < n$), then the truth-array connecting the remaining d variables can be inferred. The mechanism is the basis of array-based technology (Møller 1998).*

2.6 Generalized Modus Ponens

Modus ponens generalized to fuzzy logic is the core of fuzzy reasoning. Consider the argument

$$\begin{array}{l} A' \\ A \Rightarrow B \\ \hline B' \end{array} \tag{2.12}$$

It is similar to modus ponens, but the premise A' is slightly different from A and thus the conclusion B' is slightly different from B. Mizumoto and Zimmermann give an example (in Zimmermann 1993):

$$\begin{array}{l} \text{This tomato is very red} \\ \text{If a tomato is red, then the tomato is ripe} \\ \hline \text{This tomato is very ripe} \end{array}$$

A fuzzy rule has the following form:

$$\text{If } x \text{ is } \mathcal{A} \text{ then } y \text{ is } \mathcal{B},$$

in which $\mathcal{A}$ and $\mathcal{B}$ are fuzzy sets, defined on universes $\mathcal{X}$ and $\mathcal{Y}$, respectively. This is an implication, where the antecedent is 'x is $\mathcal{A}$', and the consequent is 'y is $\mathcal{B}$'. The following are some examples of such rules in everyday conversation:

1. If it is dark, then drive slowly.
2. If the tomato is red, then it is ripe.
3. If it is early, then John can study;
4. If the room is cold, then increase the heat.
5. If the washing machine is half full, then wash for a shorter time.

Other forms can be transcribed into the if–then form, for example 'when in Rome, do like the Romans' becomes 'if in Rome, then do like the Romans'. Examples 4 and 5 could be embedded in a computer inside a heating unit or a washing machine.

Example 2.6.1 *Student John*
To understand how a computer infers a conclusion, consider rule 3:

If it is early, then John can study

Assume that 'early' is a fuzzy set defined on the universe

$$\mathcal{U} = \langle 4, 8, 12, 16, 20, 24 \rangle$$

The time of the day is denoted by t, in steps of 4 hours in a 24-hour format to numerically distinguish night from day. Define 'early' as a fuzzy set on $\mathcal{U}$:

$$early = \{\langle 4, 0 \rangle, \langle 8, 1 \rangle, \langle 12, 0.9 \rangle, \langle 16, 0.7 \rangle, \langle 20, 0.5 \rangle, \langle 24, 0.2 \rangle\}$$

Define 'can study' as a singleton fuzzy set $\mu_{study} = 1$. If the hour is truly early, for instance 8 o'clock in the morning, then $\mu_{early}(8) = 1$, and thus John can study to the fullest degree, that is $\mu_{study} = 1$. However, if the hour is 20 (8 p.m.), then $\mu_{early}(20) = 0.5$, and accordingly John can study to the degree 0.5. The degree of fulfillment of the antecedent (the if side) weights the degree of fulfillment of the conclusion – a useful mechanism that enables one rule to cover a range of hours. The procedure is as follows: given a particular time instant t_0, the resulting truth-value is computed as $\min(\mu_{early}(t_0), \mu_{study})$.

To make the inference rule operational, we use the following definition.

Definition *Generalized modus ponens.* Let $\mathcal{A}$ and $\mathcal{A}'$ be fuzzy sets defined on $\mathcal{X}$, and let $\mathcal{B}$ be a fuzzy set defined on $\mathcal{Y}$. Then the fuzzy set $\mathcal{B}'$, induced by 'x is $\mathcal{A}'$' from the fuzzy rule

$$\text{if } x \text{ is } \mathcal{A} \text{ then } y \text{ is } \mathcal{B},$$

represented by the relation $\mu_{\mathcal{A}}(x)\mathcal{R}\mu_{\mathcal{B}}(y)$, is given by

$$\{\langle y, \mu_{\mathcal{B}'}(y)\rangle \mid \mu_{\mathcal{B}'}(y) = \mu_{\mathcal{A}'}(x) \circ (\mu_{\mathcal{A}}(x)\mathcal{R}\mu_{\mathcal{B}}(y)), x \in \mathcal{X}, y \in \mathcal{Y}\}$$

The operation $\circ$ is the $\vee - \wedge$ composition.

The generalized modus ponens is thus closely tied to relational composition. Notice also that x and y are scalars, but the definition applies vectorially as well, if taken element by element. The next example illustrates the calculations numerically.

Example 2.6.2 *Generalized modus ponens*

Given the rule 'if altitude is High, then oxygen is Low'. Let the fuzzy set High be defined on a range of altitudes from 0 to 4000 m (about 12 000 ft),

$$High = \{\langle 0, 0\rangle, \langle 1000, 0.25\rangle, \langle 2000, 0.5\rangle, \langle 3000, 0.75\rangle, \langle 4000, 1\rangle\}$$

and Low be defined on a set of percentages of normal oxygen content,

$$Low = \{\langle 0, 1\rangle, \langle 25, 0.75\rangle, \langle 50, 0.5\rangle, \langle 75, 0.25\rangle, \langle 100, 0\rangle\}$$

As a shorthand notation we write the rule as a logical proposition High $\Rightarrow$ Low, where it is understood that the proposition concerns altitude on the left side and oxygen on the right side. We construct the relation $\boldsymbol{R}$, connecting High and Low, using Gödel implication $(\mu_{High}(x) \le \mu_{Low}(y)) \vee \mu_{Low}(y)$:

$$\boldsymbol{R} = \begin{array}{c|c|c|c|c|c|} & 1 & 0.75 & 0.5 & 0.25 & 0 \\ \hline 0 & 1 & 1 & 1 & 1 & 1 \\ \hline 0.25 & 1 & 1 & 1 & 1 & 0 \\ \hline 0.5 & 1 & 1 & 1 & 0.25 & 0 \\ \hline 0.75 & 1 & 1 & 0.5 & 0.25 & 0 \\ \hline 1 & 1 & 0.75 & 0.5 & 0.25 & 0 \\ \hline \end{array}$$

The boxes and axis annotations make the construction of the table clearer: each element r_{xy} is the evaluation of $\mu_{High}(x) \Rightarrow \mu_{Low}(y)$. The numbers on the vertical axis correspond to μ_{High} and the numbers on the horizontal axis correspond to μ_{Low}. Assuming altitude is High, we find by modus ponens that

$$\begin{aligned}\boldsymbol{\mu}^t &= \boldsymbol{\mu}^t_{High} \circ \boldsymbol{R} \\ &= \begin{pmatrix} 0 & 0.25 & 0.5 & 0.75 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0.25 & 0 \\ 1 & 1 & 0.5 & 0.25 & 0 \\ 1 & 0.75 & 0.5 & 0.25 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0.75 & 0.5 & 0.25 & 0 \end{pmatrix}\end{aligned}$$

The result is identical to Low. Thus modus ponens returns a result as expected in this case.

Assume instead that altitude is Very High,

$$\boldsymbol{\mu}^t_{VeryHigh} = \begin{pmatrix}0 & 0.06 & 0.25 & 0.56 & 1\end{pmatrix},$$

the square of $\boldsymbol{\mu}^t_{High}$. *Modus ponens yields*

$$\begin{aligned}\boldsymbol{\mu}^t &= \boldsymbol{\mu}^t_{VeryHigh} \circ \boldsymbol{R} \\ &= \begin{pmatrix}0 & 0.06 & 0.25 & 0.56 & 1\end{pmatrix} \circ \begin{pmatrix}1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0.25 & 0 \\ 1 & 1 & 0.5 & 0.25 & 0 \\ 1 & 0.75 & 0.5 & 0.25 & 0\end{pmatrix} \\ &= \begin{pmatrix}1 & 0.75 & 0.5 & 0.25 & 0\end{pmatrix}\end{aligned}$$

The result is not identical to the square of μ_{Low}. *Written as an argument, we have in fact*

$$\frac{\begin{array}{l}\textit{Very High} \\ \textit{High} \Rightarrow \textit{Low}\end{array}}{\textit{Low}}$$

This is not as desired; but in fact all implication operators have some inconvenient feature, and we rest the case.

Several rules

Fuzzy controllers apply several rules at a time:

1. if x is $\mathcal{A}_1$ then y is $\mathcal{B}_2$
2. if x is $\mathcal{A}_2$ then y is $\mathcal{B}_2$

...

m. if x is $\mathcal{A}_m$ then y is $\mathcal{B}_m$

The input x and the output y are the same in all m rules. Formally, we extend the interpretation of rule i as

i. if x is $\mathcal{A}_i$ then y is $\mathcal{B}_i$, else

where the 'else' is interpreted as a logical disjunction. That is, if rule i is represented by a relational matrix $\boldsymbol{R}_i$, then formally the whole rule base is the element-by-element union

$$\boldsymbol{R} = \bigvee_{i=1}^{m} \boldsymbol{R}_i$$

Input x induces output y by means of composition. In case of n inputs, that is, if each *if* side contains n variables, the relation matrix $\boldsymbol{R}$ generalizes to an $n + 1$ dimensional array.

It is simpler in practice, as we shall see later.

2.7 Triangular Norms

There are alternative definitions of conjunction and disjunction, and they are always defined in pairs. For example,

$$\begin{array}{lll} x \wedge y: & x * y & \text{algebraic product} \\ x \vee y: & x + y - xy & \text{algebraic sum} \end{array}$$

In fact, any operation that agrees with the truth-table for two-valued conjunction is a candidate. The so-called *t-norms* and *t-conorms* are valid connectives.

For conjunction, a *triangular norm* or *t-norm* is an operation $\triangle$ satisfying (Nguyen and Walker 2000)

$$1 \triangle x = x$$

$$x \triangle y = y \triangle x$$

$$x \triangle (y \triangle z) = (x \triangle y) \triangle z$$

$$\textit{If } w \le x \text{ and } y \le z \text{ then } w \triangle y \le x \triangle z$$

Some examples are

$$(a) \quad \begin{Bmatrix} x \wedge y & \text{if } x \vee y = 1 \\ 0 & \text{otherwise} \end{Bmatrix}$$

$$(b) \quad 0 \vee (x + y - 1)$$

$$(c) \quad \frac{xy}{2 - (x + y - xy)}$$

$$(d) \quad xy$$

$$(e) \quad \frac{xy}{x + y - xy}$$

$$(f) \quad x \wedge y$$

For disjunction, a t-conorm is an operation   satisfying

$$0 \quad x = x$$

$$x \quad y = y \quad x$$

$$x \quad (y \quad z) = (x \quad y) \quad z$$

$$\textit{If } w \le x \text{ and } y \le z \text{ then } w \quad y \le x \quad z$$

A t-conorm   can be generated from a t-norm $\triangle$ by

$$x \quad y = 1 - (x + 1) \triangle (y + 1)$$

This explains why the definition of fuzzy logic operations may vary from application to application.

The implication connective illustrates the difficulties with building a consistent system for fuzzy logic. But leaving implication out, and restricting to negation, conjunction, and disjunction, it is possible to build a so-called Kleene-algebra (Nguyen and Walker 2000). Thereby the conjunctive and disjunctive normal forms, of great importance in digital electronics, can be proved to be valid in fuzzy logic.

2.8 Formal Derivation of the Mamdani Inference*

Mamdani inference is defined by means of the min operation taken as conjunction.

Definition *Mamdani inference*. Let $\mathcal{A}$ and $\mathcal{B}$ be fuzzy sets defined on $\mathcal{X}$ and $\mathcal{Y}$ respectively, then the fuzzy set in $\mathcal{X} \times \mathcal{Y}$ with the membership function

$$\{\langle\langle x, y\rangle, \mu(x, y)\rangle \mid x \in \mathcal{X}, y \in \mathcal{Y}, \mu(x, y) = \min(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(y))\} \tag{2.13}$$

is the Mamdani inference.

Intuitively, the degree of fulfillment of the premise $\mu_{\mathcal{A}}(x)$ weights, by the min operation, the membership function of the conclusion $\mu_{\mathcal{B}}(y)$; we saw the mechanism in the 'Student John' example (Example 2.6.1). Formally it rests on Tautology 3,

$$\left[p \wedge (p \Rightarrow q)\right] \Leftrightarrow p \wedge q \tag{2.14}$$

which shows that the premise combined with the implication of modus ponens is equivalent to just conjunction. The relationship is valid under a suitable choice of implication connective, for instance Gödel. To get into further detail, we will introduce the *compositional rule of inference* (Zadeh 1975) by an example.

Example 2.8.1 *Compositional rule of inference*

Generalizing modus ponens to three-valued array logic, $\mathbf{p}$ *and* $\mathbf{q}$ *are vectors of three elements,* $\boldsymbol{R}$ *is a 3-by-3 matrix, and the inner* $\vee - \wedge$ *product is interpreted as the inner max–min product. The assumption that* $\mathbf{p}$ *is true corresponds to assigning*

$$\mathbf{p} = \begin{pmatrix} 0 \\ 0.5 \\ 1 \end{pmatrix}$$

In modus ponens

$$\boldsymbol{R} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0.5 & 1 \end{pmatrix},$$

which is the truth-table for Gödel implication. With $\mathbf{p}$ *as above,*

$$\mathbf{q}^t = \mathbf{p}^{\mathbf{t}} \circ \boldsymbol{R} = \begin{pmatrix} 0 & 0.5 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0.5 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0.5 & 1 \end{pmatrix}$$

To interpret, if p is true and $p \Rightarrow q$ is true, it implies that q is true, as expected because tautology 1 is valid in three-valued logic.

The inner product $\mathbf{p}^t \circ \boldsymbol{R}$ *can be decomposed into three operations:*

1. *A Cartesian product*

$$\boldsymbol{R}^{(1)} = \mathbf{p} \times \mathbf{1}^t = \begin{pmatrix} 0 \\ 0.5 \\ 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \\ 1 & 1 & 1 \end{pmatrix}$$

*Can be skipped in a first reading.

called the cylindrical extension. *The result is a matrix whose columns are the vector* **p** *repeated as many times as necessary to fit the size of* **R**.

2. *An element-by-element conjunction*

$$\boldsymbol{R}^{(2)} = \boldsymbol{R}^{(1)} \wedge \boldsymbol{R} = \begin{pmatrix} 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \\ 1 & 1 & 1 \end{pmatrix} \wedge \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0.5 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0.5 & 1 \end{pmatrix}$$

This operation does not have a name in the literature.

3. *A disjunction along the columns of* $\boldsymbol{R}^{(2)}$,

$$\mathbf{q}^t = \bigvee_p r^{(2)}_{pq} = \begin{pmatrix} 0 & 0.5 & 1 \end{pmatrix}$$

which is the or-projection *of* $\boldsymbol{R}^{(2)}$ *on the* q*-axis.*

In general, the $\vee - \wedge$ *composition*

$$\mathbf{b}^t = \mathbf{a}^t \circ \mathbf{R}$$

consists of three operations: (1) the cylindrical extension $\boldsymbol{R}^{(1)} = \mathbf{a}^t \times 1_b$*, (2) the element-by-element conjunction* $\boldsymbol{R}^{(2)} = r^{(1)}_{ab} \wedge r_{ab}$*, and (3) the projection onto the b-axis* $\mathbf{b}^t = \bigvee_b r^{(2)}_{ab}$. *This constitutes the* compositional rule of inference*, where* **b** *is inferred by composing* **a** *with* $\boldsymbol{R}$.

In the above example, the matrix $\boldsymbol{R}$ is the truth-table for implication, corresponding to $p \Rightarrow q$ in Equation (2.14). Operations 1) and 2) together are a vectorized conjunction, corresponding to $p \wedge (p \Rightarrow q)$ in Equation (2.14). Clearly, the matrix $\boldsymbol{R}^{(2)}$ is equivalent to the right-hand side of Equation (2.14). Therefore, Mamdani inference is a direct version of modus ponens (not generalized modus ponens). The next example uses the data in the altitude example (Example 2.6.2) to illustrate this point.

Example 2.8.2 *Mamdani inference*

Given the rule 'if altitude is High, then oxygen is Low', let the fuzzy set High be defined on a range of altitudes from 0 to 4000 m (about 12 000 ft),

$$High = \{\langle 0, 0\rangle, \langle 1000, 0.25\rangle, \langle 2000, 0.5\rangle, \langle 3000, 0.75\rangle, \langle 4000, 1\rangle\}$$

and Low be defined on a set of percentages of normal oxygen content,

$$Low = \{\langle 0, 1\rangle, \langle 25, 0.75\rangle, \langle 50, 0.5\rangle, \langle 75, 0.25\rangle, \langle 100, 1\rangle\}$$

As a shorthand notation, we write the rule as a logical proposition High $\Rightarrow$ *Low, where it is understood that the proposition concerns altitude on the left side and oxygen on the right side. We construct the relation* $\boldsymbol{R}^{(2)}$*, connecting High and Low, using Mamdani inference* (min)*:*

$$\boldsymbol{R}^{(2)} = \begin{array}{c|c|c|c|c|c|} & 1 & 0.75 & 0.5 & 0.25 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0 \\ \hline 0.5 & 0.5 & 0.5 & 0.5 & 0.25 & 0 \\ \hline 0.75 & 0.75 & 0.75 & 0.5 & 0.25 & 0 \\ \hline 1 & 1 & 0.75 & 0.5 & 0.25 & 0 \\ \hline \end{array}$$

The boxes and axis annotations make the construction of the table clearer: each element r_{xy} is the evaluation of $\min\left(\mu_{High}(x), \mu_{Low}(y)\right)$. *The numbers on the vertical axis correspond to μ_{High} and the numbers on the horizontal axis correspond to μ_{Low}.*

We interpret $\boldsymbol{R}^{(2)}$ as a lookup table: an input $\mu_{High}(x)$ having membership 0 *induces an output*

$$\mu_{Low}(y) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.15}$$

(first row), an input $\mu_{High}(x)$ having membership 0.25 *induces an output*

$$\mu_{Low}(y) = \begin{pmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0 \end{pmatrix} \tag{2.16}$$

(2nd row), and so on, until an input $\mu_{High}(x)$ having membership 1 *induces an output*

$$\mu_{Low}(y) = \begin{pmatrix} 1 & 0.75 & 0.5 & 0.25 & 0 \end{pmatrix} \tag{2.17}$$

(last row). The last instance shows that if the premise is true (truth-value 1*), the induced fuzzy set is Low, as desired. The matrix is thus a listing, or stack, of all possible outputs, depending on the degree of fulfillment of the input.*

This is exactly how a Mamdani controller functions, as we shall see in a later chapter. The or-projection in the compositional rule of inference is obsolete, because inputs to a controller are scalar; the or-projection applies to fuzzy input.

2.9 Summary

Fuzzy reasoning is based on fuzzy logic, which is in turn based on fuzzy set theory. The idea of a fuzzy set is basic and simple: an object is allowed to have a gradual membership of a set. The idea pervades all derived mathematical aspects of set theory. In fuzzy logic an assertion is allowed to be more or less true. A truth value in fuzzy logic is a real number in the interval [0, 1], rather than the set of two truth values {0, 1} of classical logic.

Classical logic can be fuzzified in many ways, but the central problem is to find a suitable definition for the connective ‘implication’. Fuzzy reasoning is based on the modus ponens rule of inference, which again rests on the definition of ‘implication’.

Not all laws in classical logic are valid in fuzzy logic, and therefore a fuzzy system is a trade-off between mathematical rigour and engineering requirements. A backward approach is recommended:

1. first decide which laws (tautologies) are required;
2. define ‘and’, ‘or’, ‘not’, ‘implication’, and ‘equivalence’;
3. check by means of their truth-tables whether the laws in step 1 hold; and
4. if not, go to 2.

All the derivations are based on the truth-domain $\{0, u, 0.5, v, 1\}$ with $0 < u < 0.5$ and $0.5 < v < 1$, such that v is the negation of u. If it turns out in a particular application that this is an inadequate representation of the continuous truth-domain [0, 1], care should be taken to check the results.

Fuzzy logic has seen many applications, one among them being fuzzy control. There is more to be said about fuzzy logic, and fuzzy control applies just a subset of the arsenal of operations and definitions.

2.10 Notes and References

The notes below contain recommendations for beginners as well as for more advanced reading.

Fuzzy logic For an introduction to fuzzy logic and its applications, see the book by Zimmermann (1993). It gives an overview of the mathematics, without being overly mathematical. For an advanced study of pure fuzzy logic, the book by Nguyen and Walker (2000) includes algebras, relations, possibility theory, and fuzzy integrals. The original papers by Zadeh are still relevant and accessible. An article in IEEE Computer is a good starting point (Zadeh 1988), before reading the two central papers (Zadeh 1973, 1975). The first original paper is historically interesting (Zadeh 1965). The array-based approach to logic is founded on the work by Franksen (1979) and developed for commercial use by Møller (1986, 1998).

Applications There are now many collections describing applications of fuzzy logic. Two remarkable books by Constantin von Altrock describe case studies related to control and business (von Altrock 1995, 1996). A collection of contributions, edited by Zimmermann, gives an overview of the applications to engineering, medicine, management, and psychology, among others (Zimmermann 1999). An important collection of articles regarding image processing and pattern recognition is Bezdek and Pal published by IEEE (1992). Regarding automatic control, see the overview article by Lee (1990).

Tools The Fuzzy Logic Toolbox is a Matlab toolbox for membership functions, connectives, inference systems, adaptation of the membership functions, Simulink support, and C code generation. The toolbox includes a tutorial, which is another excellent starting point for learners. Together with the book by Jang *et al.* (1997) students of fuzzy logic and fuzzy control are well equipped. There are many software tools; since software products develop quickly, the World Wide Web is the best reference for the same.

3

Fuzzy Control

Roughly speaking, fuzzy control is 'control with rules'. A fuzzy controller can include empirical rules, and that is especially useful in operator-controlled plants. Consider the typical fuzzy controller:

$$\begin{array}{ll} 1.\ \text{If } \mathit{error} \text{ is Neg and } \mathit{change\ in\ error} \text{ is Neg then } \mathit{control} \text{ is NB} & \\ 2.\ \text{If } \mathit{error} \text{ is Neg and } \mathit{change\ in\ error} \text{ is Zero then } \mathit{control} \text{ is NM} & (3.1) \\ \dots & \end{array}$$

The rules are in the familiar if–then format, with the premise on the *if*-side and the conclusion on the *then*-side. The premise value 'Neg' is a *linguistic term* short for the word 'negative', the conclusion value 'NB' stands for 'negative big' and 'NM' for 'negative medium'. The collection of rules is a *rule base*. A computer can execute the rules and compute a control action depending on the measured inputs *error* and *change in error*.

The inclusion of fuzzy rules in a controller raises more design questions than usual. The objective here is to identify and explain those design choices.

In a rule-based controller the control strategy is in a more or less natural language. A rule-based controller is intelligible and maintainable for a non-specialist. An equivalent controller could be implemented using conventional techniques – it is just more convenient to isolate the control strategy in a rule base when operators control the plant.

In the *direct control* scheme in Figure 3.1 the fuzzy controller is in the forward path of a feedback control system. The plant output y is compared with a reference *Ref*, and if there is a deviation $e = Ref - y$, the controller takes action according to the control strategy embedded in the rule base. In the figure, the arrows can be hyper-arrows containing several signals at a time for multi-loop control.

There are at least four main sources for finding control rules (Takagi and Sugeno in Lee 1990):

- *Expert experience and control engineering knowledge.* One classical example is a handbook for cement kiln operators implemented in the *FLS controller,* by the cement company FL Smidth (Holmblad and Østergaard 1982, 1995). The most common approach is to question experts or operators with a carefully organized questionnaire.

Foundations of Fuzzy Control Jan Jantzen
© 2007 John Wiley & Sons, Ltd. ISBN: 0-470-02963-3

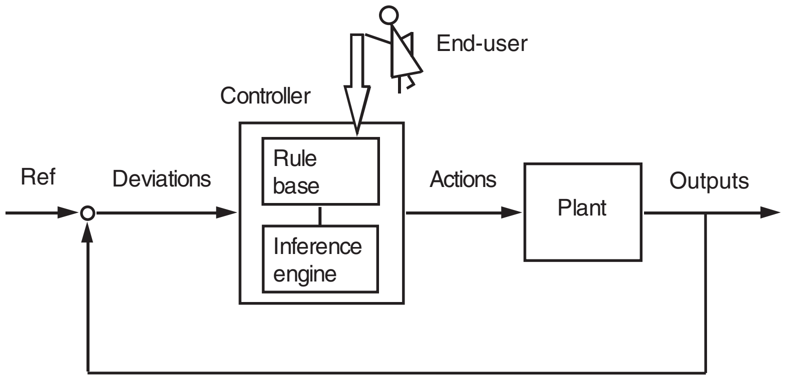


Figure 3.1: Direct control.

- *Based on the operator's control actions.* Observing an operator's control actions or examining a logbook may reveal fuzzy *if–then* rules of input–output relationships.
- *Based on a fuzzy model of the plant.* A linguistic rule base may be viewed as an inverse model of the controlled plant. Thus the fuzzy control rules can perhaps be obtained by inverting a fuzzy model of the plant, if fuzzy models of the open and closed-loop systems are available (Braae and Rutherford in Lee 1990). This method is restricted to low order systems. Another approach is *fuzzy identification* (Tong; Takagi and Sugeno; Sugeno – all in Lee 1990, Pedrycz 1993).
- *Based on learning.* The self-organizing controller is an example of a controller that finds the rules itself. It is an example of the more general control scheme: model reference adaptive control.

The chapter describes the basic components and functions of fuzzy controllers.

3.1 Controller Components

In the block diagram Figure 3.2, the controller is between a pre-processing block and a post-processing block. The following explains the diagram block by block.

Pre-processing block

Let us assume the inputs are *crisp* measurements from measuring equipment, rather than linguistic. A pre-processor, the first block in Figure 3.2, conditions the measurements before they enter the controller. Examples of pre-processing are

- quantization in connection with sampling or rounding to integers;
- normalization or scaling onto a particular, standard range;
- filtering in order to remove noise;
- averaging to obtain long-term or short-term tendencies;

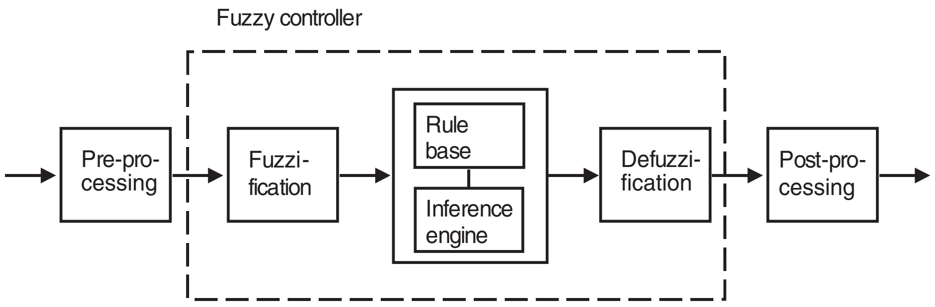


Figure 3.2: Building blocks of a fuzzy controller.

- a combination of several measurements to obtain key indicators; and
- differentiation and integration, or their approximations in discrete time.

A *quantizer* converts an inbound measurement in order to fit it to a discrete universe. Assume, for instance, that the variable *error* is 4.6, but the universe is $\mathcal{U} = \langle -5, -4, \ldots, 0, \ldots, 4, 5 \rangle$. The quantizer thus rounds to 5, the nearest level. Quantization is a means to reduce data, but if the quantization is too coarse the controller may oscillate around the reference or even become unstable.

The FL Smidth (FLS) controller applies nonlinear *scaling* (Figure 3.3). The operator supplies a value for a typical small measurement, a typical normal measurement, and a typical large measurement according to experience (Holmblad and Østergaard 1982). The combined effect of the scaling and the membership functions is a distortion of the membership functions. In fact, over the years, only the break points have been adjusted, while the primary sets have remained unchanged.

A dynamic controller has additional time-related inputs: derivatives, integrals, or previous values of measurements backwards in time. The pre-processor forms these.

Fuzzification block

The first block inside the controller is *fuzzification*, which is a lookup in the membership functions to derive the membership grades. The fuzzification block thus evaluates the input

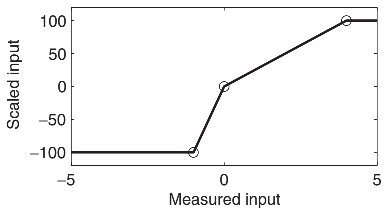


Figure 3.3: Example of nonlinear scaling of an input measurement to a standard universe [−100, 100]. Circles indicate a typical small, a typical normal, and a typical large measurement acquired from a skilled operator. (figscal.m)

measurements according to the premises of the rules. Each premise produces a membership grade expressing the degree of fulfilment of the premise.

Rule base block

A rule allows for several variables both in the premise and the conclusion. A controller can therefore be multi-input–multi-output (MIMO) or single-input–single-output (SISO). The typical SISO controller regulates a control signal according to an error signal. A controller may actually apply the *error*, the *change in error*, and the *integral error*, but we will still call it SISO control, because the inputs are based on a single feedback loop. This section assumes that the control objective is to regulate a plant output around a prescribed *setpoint* (*reference*) using a SISO controller.

A linguistic controller contains rules in the *if–then* format, but they can appear in other formats. Matlab's Fuzzy Logic Toolbox presents the rules to the end-user in a format similar to the one below:

1. If *error* is Neg and *change in error* is Neg then *control* is NB
2. If *error* is Neg and *change in error* is Zero then *control* is NM
3. If *error* is Neg and *change in error* is Pos then *control* is Zero
4. If *error* is Zero and *change in error* is Neg then *control* is NM
5. If *error* is Zero and *change in error* is Zero then *control* is Zero (3.2)
6. If *error* is Zero and *change in error* is Pos then *control* is PM
7. If *error* is Pos and *change in error* is Neg then *control* is Zero
8. If *error* is Pos and *change in error* is Zero then *control* is PM
9. If *error* is Pos and *change in error* is Pos then *control* is PB

The rules are an example for the sake of illustration, but we shall use it throughout the book. The names Zero, Pos, Neg are labels of fuzzy sets as well as NB, NM, PB and PM (negative big, negative medium, positive big, and positive medium respectively). The designer assigns the names. The example has two inputs, *error* and *change in error*. The latter is based on the time derivative of the former. These are the inputs to the controller. The conclusion side of each rule prescribes a value for the variable *control*, the output of the controller. The same set of rules is presented here in a *relational* format.

Error	Change in error	Control
Neg	Neg	NB
Neg	Zero	NM
Neg	Pos	Zero
Zero	Neg	NM
Zero	Zero	Zero
Zero	Pos	PM
Pos	Neg	Zero
Pos	Zero	PM
Pos	Pos	PB

(3.3)

The top row is a heading. It is understood that the two leftmost columns constitute the premises, the rightmost the conclusions, and each row represents one rule. This format is more compact, and it provides an overview of the rule base quickly. The relational format is certainly suited for storing in a relational database. The relational format implicitly assumes that the premise variables are connected by a connective – logical 'and' or logical 'or'. The same connective applies to all rules, not a mixture of connectives. Incidentally, a fuzzy rule with an 'or' combination of terms can be converted into an equivalent 'and' combination of terms using the laws of logic (foremost DeMorgan's laws).

A third, even more compact format is the *tabular* format,

		Change in error		
		Neg	Zero	Pos
	Neg	NB	NM	Zero
Error	Zero	NM	Zero	PM
	Pos	Zero	PM	PB

(3.4)

The premise variables *error* and *change in error* are laid out along the axes, and the conclusions are inside the table. Symmetries can be discovered readily, and an empty cell is an indication of a missing rule; thus the format is useful for checking completeness. When the premise variables are *error* and *change in error*, that format is also called a *linguistic phase plane*. In case the number of premise variables is $n > 2$, the table grows to an n-dimensional array.

A nested arrangement can accommodate several conclusions. A rule with several conclusions can alternatively be broken down into several rules, each having one conclusion.

Lastly, a *graphical format* displays the fuzzy membership curves (see Example 3.1.2). This graphical user interface illustrates the inference mechanism better than the other formats, but uses more space on a computer monitor.

The most prominent connective is the 'and' connective, often implemented as multiplication instead of minimum. The examples, so far, only contained 'and' operations, but a rule such as 'If *error* is very Neg and not Zero or *change in error* is Zero then . . .' is also possible.

The connectives 'and' and 'or' are always defined in pairs. For example, loosely written,

$$\begin{array}{ll}
\mathcal{A} \wedge \mathcal{B} \equiv \min(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)) & \text{minimum} \\
\mathcal{A} \vee \mathcal{B} \equiv \max(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)) & \text{maximum} \\
\qquad\qquad\text{or} & \\
\mathcal{A} \wedge \mathcal{B} \equiv \mu_{\mathcal{A}}(x) * \mu_{\mathcal{B}}(x) & \text{algebraic product} \\
\mathcal{A} \vee \mathcal{B} \equiv \mu_{\mathcal{A}}(x) + \mu_{\mathcal{B}}(x) - \mu_{\mathcal{A}}(x) * \mu_{\mathcal{B}}(x) & \text{probabilistic sum}
\end{array}$$

There are other, more complex, definitions (e.g. Zimmermann 1993).

Before designing the membership functions it is necessary to consider the universes for the premises and conclusions. Take, for example, the rule

If *error* is Neg and *change in error* is Pos then *control* is Zero

The membership functions for Neg and Pos must be defined for all acceptable measurements of *error* and *change in error,* nevertheless, a *standard universe* may be convenient.

Premise membership functions can be continuous or discrete. A continuous membership function is a function defined on a continuous universe. A discrete membership function is a vector with a finite number of elements. The latter case requires specification of the range of the universe and the value at each discrete sampling point. The choice between fine and coarse resolution is a trade-off between accuracy, computational speed, and memory space. The quantizer takes time to execute, and if this time is too precious, continuous membership functions will make the quantizer obsolete.

Example 3.1.1 *Standard universes*
Many authors and several commercial controllers use standard universes.

- *The FLS controller uses the real number interval* $[-1, 1]$.
- *Authors of the early papers on fuzzy control used the short integer range* $[-6, 6]$, *because computer memory was scarce at that time.*
- *Another possibility is the interval* $[-100, 100]$ *corresponding to the percentage of full range of a measurement.*
- *Yet another possibility is the integer range* $[0, 4095]$ *arising from a 12 bit conversion of an analogue signal to digital representation.*
- *A variant is the integer range* $[-2047, 2048]$, *where the interval is shifted in order to accommodate negative numbers.*

The choice of data type may govern the choice of universe. For example, the voltage range $[-5, 5]$ *volts could be represented as an integer range* $[-50, 50]$, *or as a floating point range* $[-5.0, 5.0]$; *a signed byte data type has an allowable integer range* $[-128, 127]$.

Scaling is a means to expand the range of operation of a variable. If a controller input mostly operates within a small interval, increasing the scaling factor increases the range of operation within the universe.

The designer is inevitably faced with the problem of how to design the term sets, for example, the *family* of terms Neg, Zero, and Pos. There are two specific questions to consider: (1) How does one determine the shape of the sets? and (2) How many sets are necessary and sufficient?

According to fuzzy set theory the choice of shape and width is subjective, thus a solution is to ask the plant operators to draw their personal preferences for the membership curves; but operators likely find it difficult to settle on particular curves. A few rules of thumb apply, however:

- A term set should be sufficiently wide to allow for noise in the measurement.
- A certain amount of overlap is desirable; otherwise the controller may run into poorly defined states, where it does not return a well-defined output.
- If there is a gap between two neighbouring sets, no rules fire for values in the gap. Consequently the controller is undefined in that gap.
- The necessary and sufficient number of sets in a family depends on the width of the sets, and vice versa.

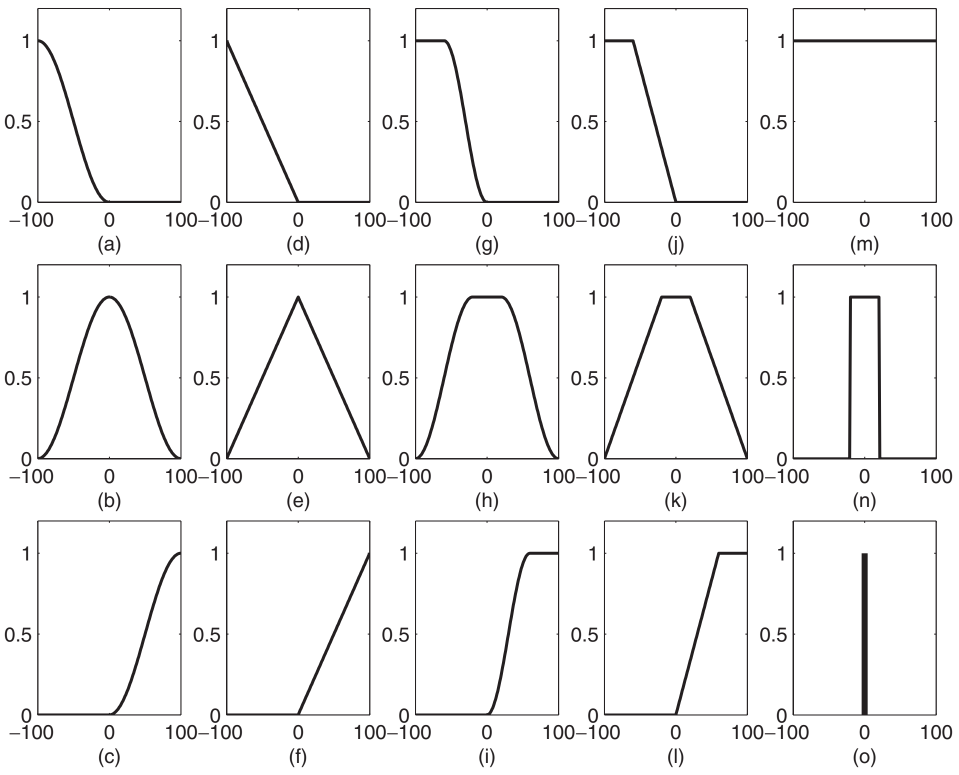


Figure 3.4: Examples of primary sets. Columnwise: families of smooth triangular (a–c), triangular (d–f), smooth trapezoidal (g–i), trapezoidal functions (j–l). The last column is the set *anything* (m), a crisp set (n), and a singleton set (o). (figsets.m)

Membership functions can be flat on the top, piece-wise linear and triangular, trapezoidal, or ramps with horizontal shoulders. Figure 3.4 shows some typical shapes of membership functions.

A constant in the conclusion is theoretically a *singleton conclusion*. For example, in the rule base

1. If *error* is Pos then *control* is 10 volts
2. If *error* is Zero then *control* is 0 volts
3. If *error* is Neg then *control* is -10 volts

the control action is a constant. There are at least three advantages to this: (1) the computations are simpler; (2) it is possible to drive the control signal to its extreme values; and (3) it is more intuitive. The constant can be represented as a fuzzy singleton $\langle x, \mu_{\mathcal{A}}(x)\rangle$ placed in position x. For example, 10 *volts* would be equivalent to the fuzzy membership function $\langle 0, 0, 0, 0, 1\rangle$ defined on the universe $\langle -10, -5, 0, 5, 10\rangle$ *volts*.

Example 3.1.2 *Membership functions*
Fuzzy controllers use a variety of membership functions. A common example is the Gaussian *curve based on the exponential function,*

$$\mu_{Gauss}(x) = \exp\left[\frac{-(x - x_0)^2}{2\sigma^2}\right]$$

This is a standard Gaussian curve with a maximum value of 1*; x is the independent variable on the universe, x_0 is the position of the peak relative to the universe, and σ is the standard deviation.*

The bell *membership function does not use the exponential,*

$$\mu_{Bell}(x) = \left[1 + \left(\frac{x - x_0}{\sigma}\right)^{2a}\right]^{-1}$$

The extra parameter a, usually positive, affects the width of the membership function and the slope of the sides. The FLS controller uses the equation

$$\mu_{FLS}(x) = 1 - \exp\left[-\left(\frac{\sigma}{x_0 - x}\right)^{a}\right]$$

It is also possible to use other functions, for example, the sigmoid *known from neural networks, or the cosine-based trapezoids from the previous chapter.*

Inference engine block

Figure 3.5 is a graphical construction of the inference, where each of the nine rows represents one rule. Consider, for instance, the first row: if the *error* is negative (row 1, column 1) and the *change in error* is negative (row 1, column 2) then the control action should be negative big (row 1, column 3). The chart corresponds to the rule base (3.2). Since the controller combines the *error* and the *change in error*, the controller is a fuzzy version of a proportional-derivative (PD) controller.

The instances of the *error* and the *change in error* are indicated by the vertical lines through the first and second columns of the chart. For each rule, the inference engine looks up the membership value where the vertical line intersects a membership function.

The *firing strength* α_k of a rule k is the degree of fulfilment of the rule premise. Rule k causes a fuzzy membership value $\mu_{\mathcal{A},k}(error)$ corresponding to the *error* measurement, and a membership value $\mu_{\mathcal{B},k}(change\ in\ error)$ corresponding to the *change in error* measurement. Their *aggregation* is the combination,

$$\alpha_k = \mu_{\mathcal{A},k}(error) \wedge \mu_{\mathcal{B},k}(change\ in\ error)$$

The $\wedge$-operation is the 'and' connective combining the two propositions in Equation (3.2); in general, it can be a combination of many propositions connected by $\wedge$ or $\vee$.

The *activation* of a rule is the derivation of a conclusion depending on the firing strength. Only a portion of each singleton is activated, and min or $*$ (multiplication) is applied as the *activation operator*. The result is the same, when the conclusions are singletons $\langle S_k, 1\rangle$ $(k = 1, 2, \ldots, 9)$, but in general, $*$ scales the membership curves, thus preserving the

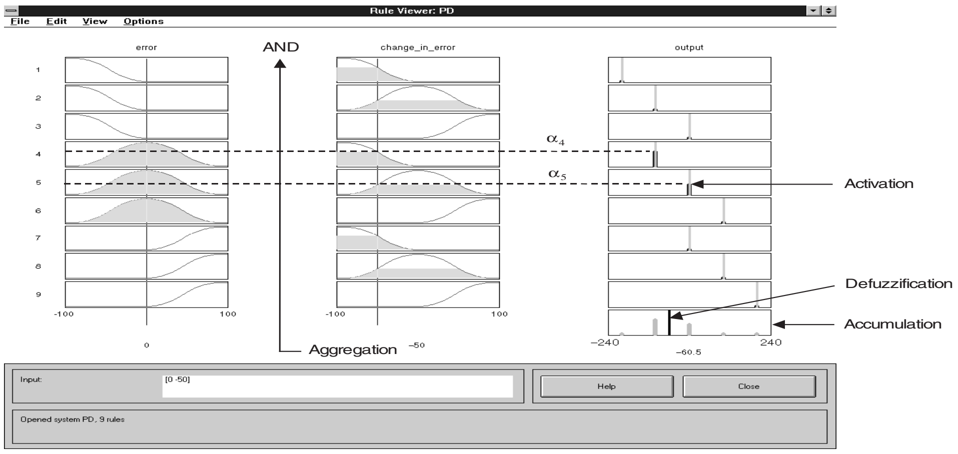


Figure 3.5: Graphical construction of the control signal in a fuzzy PD controller (generated in the Matlab Fuzzy Logic Toolbox)

initial shape, while min clips them. Both methods work well in practice, although the multiplication results in a slightly smoother control signal. In Figure 3.5, only rules 4 and 5 contribute.

A rule k can be weighted a priori by a weighting factor $\omega_k \in [0, 1]$, which is its *degree of confidence*. In that case the firing strength is modified to

$$\alpha_k^* = \omega_k * \alpha_k$$

The degree of confidence ω_k is determined by the designer, or an optimization program.

All activated conclusions are *accumulated*, using the set union operation, to the final graph on the bottom right (Figure 3.5). Usually max-accumulation is applied, but alternatively sum-accumulation can be applied, if it makes sense to count overlapping areas more than once. In the figure, $*$-activation followed by max-accumulation results in the membership function

$$\mu_c(S_k) = \left\langle 0, \alpha_4^* * s_4, \alpha_5^* * s_5, 0, 0 \right\rangle$$

The conclusions may contain several control actions. An example of a one-input-two-output rule is ‘If $error$ is $\mathcal{A}$ then u_1 is $\mathcal{B}$ and u_2 is $\mathcal{C}$’. The inference engine executes two conclusions in parallel by applying the firing strength to both conclusion sets $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$. In practice, one would implement this situation as two rules rather than one: ‘If $error$ is $\mathcal{A}$ then u_1 is $\mathcal{B}$’ and ‘If $error$ is $\mathcal{A}$ then u_2 is $\mathcal{C}$’.

Defuzzification block

The resulting fuzzy set μ_c (Figure 3.5, bottom right) must be converted to a single number in order to form a control signal to the plant. This is *defuzzification*. In the figure the defuzzified control signal is the x-coordinate marked by a dashed, vertical line. Several defuzzification methods exist.

The crisp control value u_{COG} is the abscissa of the *centre of gravity* of the fuzzy set. For discrete sets, its name is *centre of gravity for singletons,* COGS,

$$u_{COGS} = \frac{\sum_i \mu_c(x_i)\, x_i}{\sum_i \mu_c(x_i)} \tag{3.5}$$

where x_i is a point in the universe $\mathcal{U}$ of the conclusion $(i = 1, 2, \ldots)$, and $\mu_c(x_i)$ its membership of the resulting conclusion set. The expression is the membership weighted average of the elements of the set. For continuous sets, replace summations by integrals and call it COG. The method is much used, although its computational complexity is relatively high.

With singleton conclusions (Figure 3.5) and sum-accumulation, the resulting defuzzified value is

$$u = \frac{\sum_k \alpha_k^* S_k}{\sum_k \alpha_k^*} \tag{3.6}$$

Here S_k is the position of the singleton in rule k in $\mathcal{U}$, and α_k^* is the firing strength of rule k. It has the advantage that u is differentiable with respect to the singletons S_k, a useful property for optimization algorithms.

The *bisector of area* method, BOA, finds the abscissa x of the vertical line that partitions the area under the membership function into two areas of equal size. For discrete sets, u_{BOA} is the abscissa x_j that minimizes

$$\left| \sum_{i=1}^{j} \mu_c(x_i) - \sum_{i=j+1}^{i_{\max}} \mu_c(x_i) \right|, \qquad 1 < j < i_{\max} \tag{3.7}$$

Here $i_{\max}$ is the index of the largest abscissa $x_{i_{\max}} \in \mathcal{U}$. Its computational complexity is relatively high. There may be several solutions x_j.

An intuitive approach is to choose the point of the universe with the highest membership. Several such points may exist, and it is common practice to take the *mean of maxima* (MOM),

$$u_{\mathrm{MOM}} = \frac{\sum\limits_{i \in \mathcal{I}} x_i}{|\mathcal{I}|}, \quad \mathcal{I} = \left\{ i \mid \mu_c(x_i) = \mu_{\max} \right\}$$

where $\mathcal{I}$ is the (crisp) set of indices i where $\mu_c(x_i)$ reaches its maximum $\mu_{\max}$, and $|\mathcal{I}|$ is its cardinality (the number of members). This method disregards the shape of the fuzzy set, but the computational complexity is relatively good.

Another possibility is to choose the position in the universe of the *leftmost maximum* (LM),

$$u_{LM} = x_{\min(\mathcal{I})}$$

or the position in the universe of the *rightmost maximum* (RM)

$$u_{RM} = x_{\max(\mathcal{I})}$$

A robot, for example, must choose left or right to avoid an obstacle in front of it; thus the defuzzifier must choose one (LM) or the other (RM), not something in between. These defuzzification methods are indifferent to the shape of the fuzzy set, but the computational complexity is relatively small.

Post-processing block

If the inferred control value is defined on a standard universe, it must be scaled to *engineering units,* for instance: volts, meters, or tons per hour. An example is the scaling from the standard universe $[-1, 1]$ to the physical units $[-10, 10]$ volts. The post-processing block contains an output gain that can be tuned.

3.2 Rule-Based Controllers

Three distinct variants of controller have evolved historically: the *Mamdani*, the *FLS*, and the *Sugeno* controllers. They use the same general inference scheme, but they differ with respect to activation method and conclusion membership functions.

A SISO rule base will pinpoint the essential differences,

$$\text{If } error \text{ is Neg then } control \text{ is Neg}$$
$$\text{If } error \text{ is Zero then } control \text{ is Zero}$$
$$\text{If } error \text{ is Pos then } control \text{ is Pos}$$

The input is *error*, proportional to the deviation from the setpoint. The $control$ action has the same sign as *error*, linguistically. Thus the rule base expresses a fuzzy proportionality between *error* and *control*, and the controller is a fuzzy version of a proportional (P-) controller. The following paragraphs explain the characteristics of the controller variants by means of graphical construction.

The Mamdani controller

Figure 3.6 illustrates the Mamdani controller. Each of the three rows refers to one rule. A particular instance of an *error* measurement ($error = -50$) is indicated by a vertical dashed line intersecting all three rules. Firing strengths are indicated by horizontal dashed lines.

The Mamdani controller applies activation function min, resulting in a clipping of the conclusion sets. The accumulated conclusion on the far right of the figure thus contains sharp breakpoints. Defuzzification method COG results in the control signal $u = -25.7$ in the figure.

The inference mechanism is motivated by intuitive clarity: it is evident how gradual fulfilment of a rule contributes to the accumulated conclusion.

Example 3.2.1 *Mamdani inference*
Take the inference in Figure 3.6; how is it implemented? Behind the scenes the premise membership functions are continuous, while the conclusion functions are discrete, divided into 201 *integer points in* $[-100, 100]$. *But for brevity, let us just use five points here. Assume the conclusion universe* $\mathcal{U}$ *is defined by the Matlab vector*

```
u = [-100  -50  0  50  100]
```

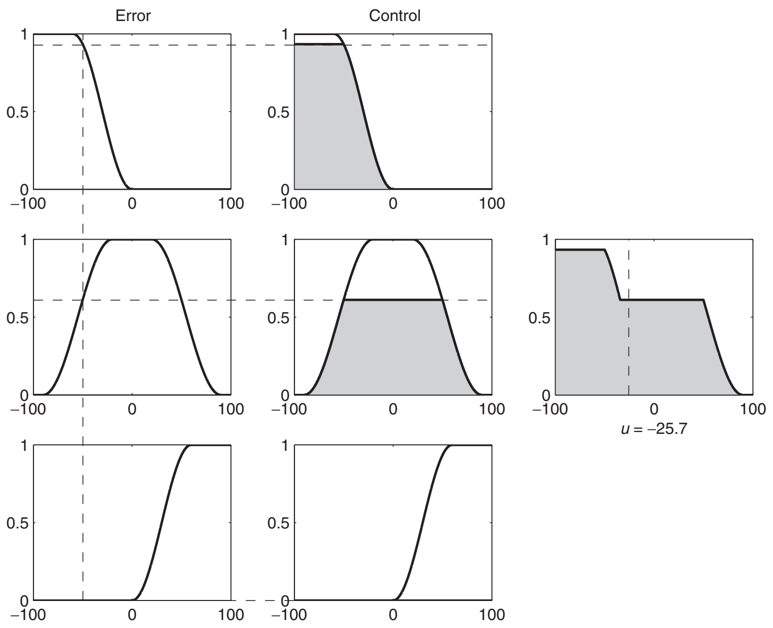


Figure 3.6: Mamdani inference. (figrbase.m)

A smooth trapezoid function $\mu_{STrapezoid}(x; a, b, c, d)$ *defines the membership functions with*

$$\mu_{Neg} = \mu_{STrapezoid}(x; -100, -100, -60, 0)$$

$$\mu_{Zero} = \mu_{STrapezoid}(x; -90, -20, 20, 90)$$

$$\mu_{Pos} = \mu_{STrapezoid}(x; -100, 0, 60, 100)$$

where a is the left footpoint, b is the left shoulderpoint, c is the right shoulderpoint, and d is the right footpoint. Inserting `u` *for* x *the conclusion term set is represented by three vectors*

```
 Neg = [1  0.93  0.05  0  0]
Zero = [0  0.61  1  0.61  0]
 Pos = [0  0  0.05  0.93  1]
```

Here we inserted the whole vector `u` *in place of the running point* `x`*; the result is thus a vector for each set. In the figure* error $= -50$, *the unit is a percentage of the full range. Thus the firing strength of the first rule is* $\alpha_1 =$ `Neg(2)` $= 0.93$. *Using* min *as the activation function, the conclusion is in Matlab* `min(0.93, Neg)`, *which yields*

```
0.93  0.93  0.05  0  0
```

The firing strength of the second rule is $\alpha_2 =$ `Zero(2)` $= 0.61$, *and the firing strength of the third rule is* $\alpha_3 =$ `Pos(2)` $= 0$. *Activate the two remaining rules, and stack all three* *contributions on top of each other,*

```
/0.93  0.93  0.05     0  0\
|   0  0.61  0.61  0.61  0|
\   0     0     0     0  0/
```

Accumulation using max *down each column yields the vector*

```
0.93  0.93  0.61  0.61  0
```

The COGS *yields*

$$\begin{aligned} u_{COGS} &= \frac{\sum_i \mu_c(x_i)\, x_i}{\sum_i \mu_c(x_i)} \\ &= \frac{0.93 * (-100) + 0.93 * (-50) + 0.61 * 0 + 0.61 * 50 + 0 * 100}{0.93 + 0.93 + 0.61 + 0.61 + 0} \\ &= -35.4 \end{aligned}$$

which is the defuzzified control signal (before post-processing). The number differs from the number in the figure, because of the lower resolution of five element vectors rather than 201.

The FLS controller

Figure 3.7 illustrates the FLS controller. The membership functions are the same as earlier, and the error measurement is the same.

But the FLS controller applies the activation function ‘product’, causing a scaling of the conclusion sets. The defuzzification method is again COGS for the sake of comparison, although the real FLS controller applies the BOA method. The defuzzified control signal is $u = -29.7$, which is less negative than the Mamdani control signal.

The $*$-activation is motivated by a wish to preserve the shape of the conclusion sets.

The Sugeno controller

We saw that conclusions can be singletons, but they can also be linear combinations of the inputs, or even a complex function of the inputs (Takagi and Sugeno 1985). The general *Sugeno* rule structure is

$$\text{If } f(e_1 \text{ is } \mathcal{A}_1, e_2 \text{ is } \mathcal{A}_2, \ldots, e_k \text{ is } \mathcal{A}_k) \text{ then } y = g(e_1, e_2, \ldots, e_k)$$

Here f is a logical function that connects the sentences forming the premise, y is the conclusion, and g is a function of the inputs. A simple example is

$$\text{If error is Zero and change in error is Zero then control } y = c$$

where c is a constant, or in other words, a singleton $\langle x_i, \mu_y(x_i)\rangle = \langle c, 1\rangle$. This is a *zero*-order conclusion. A slightly more complex rule is

$$\text{If } E \text{ is } \mathcal{A} \text{ and } CE \text{ is } \mathcal{B} \text{ then } u = a * E + b * CE + c$$

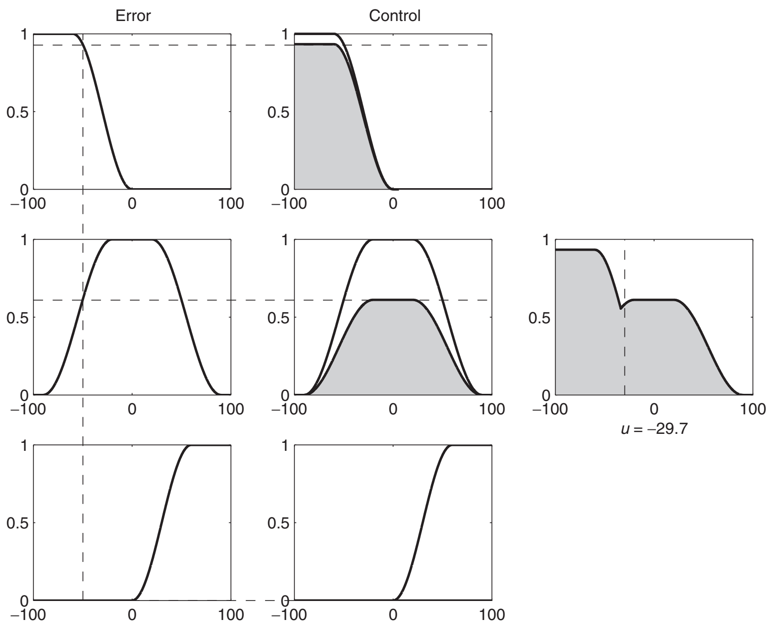


Figure 3.7: FLS inference. (figrbase.m)

where a, b, and c are all constants, and (E, CE) are abbreviations for *error* and *change in error* respectively. This is a *first*-order conclusion. Inference with several rules proceeds as before, but each control action is linearly dependent on the inputs. The control action from each rule is a dependent singleton, and the defuzzified control signal is the weighted average of the contributions from each rule (sum-accumulation and COGS).

The controller in the example can be extended to interpolate between linear PD controllers, each controller dominated by one rule, with a weighting depending on the overlap of the premise membership functions. This is useful in a nonlinear control system, where each controller operates in a subspace of the operating envelope. Higher-order conclusions $y = g(e_1, e_2, \ldots, e_k)$ are also possible.

Example 3.2.2 *Rule-based interpolation (after Takagi and Sugeno 1985)*
Suppose we have two rules

1. If error is Large then output is Line 1

2. If error is Small then output is Line 2

Line 1 is defined as $0.2 * \mathit{error} + 90$ *and line 2 is defined as* $0.6 * \mathit{error} + 20$*. The rules interpolate between the two lines in the interval where the membership functions overlap (Figure 3.8). Outside of that interval the conclusion is a linear function of the* error.

The Sugeno controller is illustrated in Figure 3.9. The premise membership functions are the same as before, but the Sugeno controller applies singleton conclusions. The accumulation operation is $+$, and the defuzzification method is COGS; the overall operation

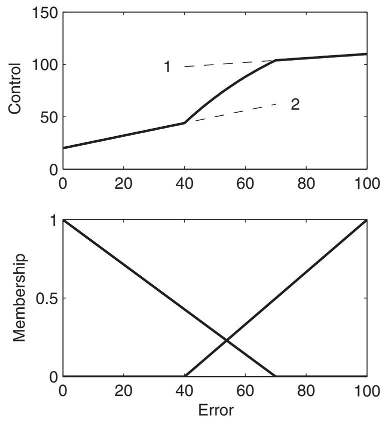


Figure 3.8: Interpolation between two lines (a), in the interval of overlap of two membership functions (b). (figsug2.m)

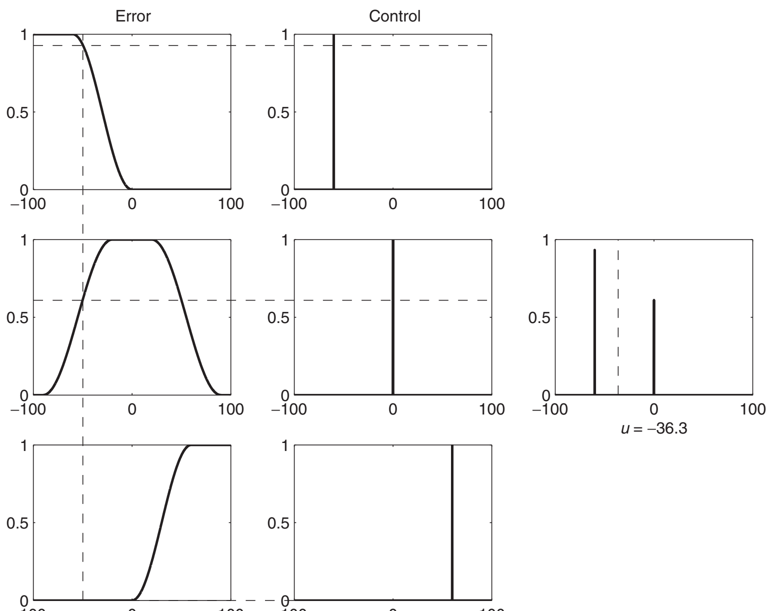


Figure 3.9: Sugeno inference with singleton output. (figrbase.m)

is a firing strength weighted average of the singleton conclusions. The defuzzified control signal is $u = -36.3$, which is more negative than the two previous controllers.

The singleton conclusions are motivated by simplicity and differentiability for optimization algorithms.

3.3 Table-Based Controller

With discrete premise universes, it is possible to infer all possible control actions offline, *before* putting the controller into operation. In a *table-based controller* the relation between all combinations of the premise universe points (Cartesian product) and their corresponding control actions are arranged in a table. With two controller inputs and one control action, the table is a two-dimensional *lookup table*. With three inputs the table becomes a three-dimensional array. The pre-calculation improves execution speed, as the runtime inference is reduced to a table lookup that is normally faster. Below is a small example of a lookup table corresponding to the rulebase (3.2) with the membership functions in Figure 3.5,

		Change in error				
		−100	−50	0	50	100
	100	0	40	100	160	200
	50	−40	0	61	121	160
Error	0	−100	−61	0	61	100
	−50	−160	−121	−61	0	40
	−100	−200	−160	−100	−40	0

(3.8)

A typical application for the table-based controller is to embed it in a processing unit, in a car, for instance, where the table is downloaded to a controller that performs the table look-up.

With two inputs and one output the input–output mapping is a surface, the *control surface*. Figure 3.10 is a mesh plot of the relationship between *error* and *change in error* on the premise side, and *control* action u on the conclusion side, resulting from a rule base with nine rules (Figure 3.5). The horizontal plateaus are due to maxima of the premise sets. The plateau in the centre implies a low sensitivity towards changes in either *error* or *change in error* near the steady state. This is an advantage if noise sensitivity must be low when the plant is near the reference. On the other hand, if the plant is open-loop unstable near the reference, it will be necessary to have a larger gain around the centre.

A negative value of *error* implies the plant output y is above the reference *Ref.* A positive value of *error* implies the plant output y is below the reference *Ref.* A negative

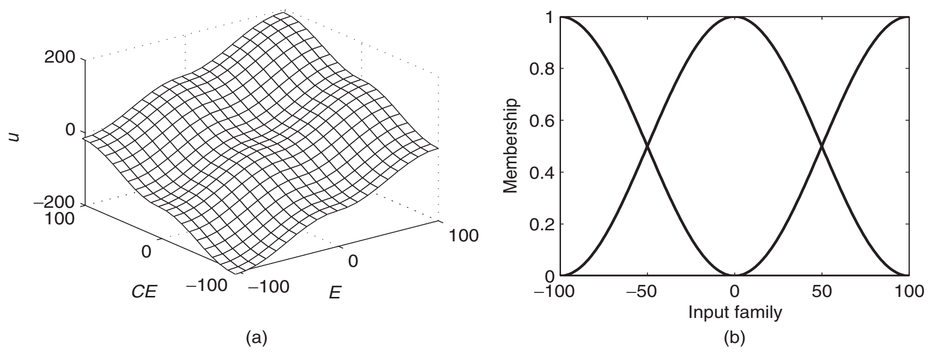


Figure 3.10: Control surface corresponding to the rule base in Figure 3.5. (figsurfs.m)

value of *change in error* implies an increasing plant output y (for constant *Ref*), while a positive value implies a decreasing plant output y.

The table in the preceding text contains several regions of interest. The cell in the centre of the table has *error* equal to zero, that is, the plant is on the reference. Furthermore, the *change in error* is zero here, that is, the plant is steady. Thus the centre cell is the stable point where the plant has settled on the reference in steady state. The diagonal is zero; all these are the favourable states, where the plant is either stable on the reference or approaching the reference. Should the plant move away a little from the zero diagonal, because of noise or a disturbance, the table values are small, and the controller will make small corrections to get it back. Should the plant be far off the reference and, furthermore, heading away from it, we are in the upper left and lower right corners: here the controller calls for drastic changes. Generally, the numerical values on the two sides of the zero diagonal may be any values, portraying an asymmetric control strategy.

During a response with overshoot, after a positive step in the reference, a plot of the point (*error*, *change in error*) will follow a trajectory in the table that spirals clockwise from the lower left corner of the table towards the centre. It is similar to a *phase plane* trajectory, the plot of a variable against its own derivative. A skilled designer, or an optimization algorithm, may adjust the numbers during a tuning session to obtain a particular response.

If the resolution in the table is too coarse it will cause *limit cycles*, oscillations about the reference. The table allows the *error* to drift away from the centre cell until it jumps into a neighbouring cell with a non-zero control action. This can be solved with *bilinear interpolation* between the cells instead of rounding to the nearest cell.

Example 3.3.1 *Bilinear interpolation*

Considering a two-dimensional table, a computed error E may fall between two neighbouring discrete values in the universe, E_i and E_{i+1}, such that $E_i < E < E_{i+1}$. The computed change in error CE may fall between two neighbouring discrete values in its universe, CE_j and CE_{j+1}, such that $CE_j < CE < CE_{j+1}$. The resulting control signal is found by interpolating linearly in the E axis direction between the first pair,

$$u_1 = g\,(E;\, F(i, j), F(i+1, j))$$

and the second pair,

$$u_2 = g\,(E;\, F(i, j+1), F(i+1, j+1))$$

and then in the CE-axis direction,

$$u = g\,(CE;\, u_1, u_2)$$

The function g is linear interpolation, and F_{ij} is the fuzzy lookup table $(i, j = 1, 2, \ldots)$.

A three-input controller implies a three-dimensional lookup table. Assuming a resolution of, say, 21 points in each universe, the table holds $21^3 = 9261$ elements. The choice of 21 points is from a standard universe $[-100, 100]$ with steps of 10, but the choice of resolution is arbitrary. It would be a tremendous task to fill these in manually, but it is manageable with rules.

A three-dimensional table can be reshaped into a two-dimensional relational representation. Rearrange the table into four columns: one for each of the three inputs (x, y, z) and one for the control action u, see, for example, Table 3.1. The table lookup is now a question of finding the correct row, and picking the corresponding u value.

Table 3.1: Relational representation of a three-dimensional lookup table.

x	y	z	u
−100	−100	−100	−100
−100	−100	−67	−89
−100	−100	0	−67
−100	−100	67	−44
−100	−100	100	−33
−100	−67	−100	−89
−100	−67	−67	−78
−100	−67	0	−56
...	...	...	...
100	100	100	100

3.4 Linear Controller

There are three sources of nonlinearity in a fuzzy controller. First, the rule base. The position, shape, and number of membership functions on the premise side, as well as nonlinear input scaling, cause nonlinear characteristics. Even the rules themselves can express a nonlinear control strategy. Second, the inference engine. If the connectives $\wedge$ and $\vee$ are implemented as min and max respectively, they are nonlinear. The same applies to min-activation and max-accumulation. And third, the defuzzification method. Several defuzzification methods are nonlinear.

It is possible to construct a rule base, however, with a linear input–output characteristic (Siler and Ying 1989, Mizumoto 1992, Qiao and Mizumoto 1996):

- The premise universes must be large enough for the inputs to stay within the limits, in other words, to avoid saturation. Each premise family should contain a number of terms, with an overlap such that the sum of membership values for any particular input instance is 1. This is achieved with duplicates of symmetric, triangular sets that cross their neighbour sets at the membership value $\mu = 0.5$. Their peaks will thus be equidistant. Any input instance can thus be a member of at most two sets simultaneously, and the membership of each is a piece-wise linear function of the input.

- The number of terms in each family determines the number of rules, since the rule base must consist of the $\wedge$-combination of all terms to ensure completeness. With singleton conclusion sets $\langle s_i, 1\rangle$, s_i must be the sum of the peak positions of the premise sets. Take, for example, the first of the nine rules in (3.2),

  If *error* is Neg and *change in error* is Neg, then *control* is NB.

  Assuming the peak of Neg is in -100, then the constant NB must equal $-100 + (-100) = -200$, if the rule base is to act like a summation.

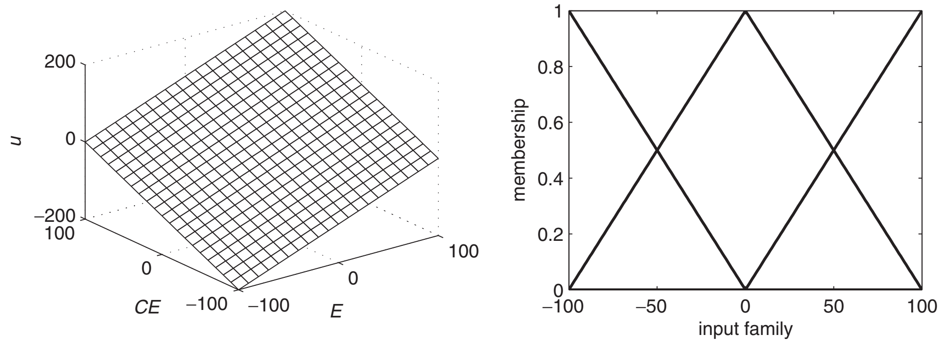


Figure 3.11: Linear control surface. The controller acts like a summation: $u = E + CE$. (figsurfs.m)

- We must choose multiplication for the connective $\wedge$. Using sum-accumulation and COGS the denominator vanishes, because all firing strengths add up to 1.

It is an advantage to use conclusion singletons; then the rule base becomes equivalent to a plain *summation* of the inputs. The following list summarizes five design choices for achieving a fuzzy rule base equivalent to a summation.

Conditions In order to achieve a fuzzy controller $u = F(E, CE)$ equivalent to the summation $u = E + CE$, the following conditions must be fulfilled:

1. Use triangular premise sets that cross at $\mu = 0.5$.
2. Build a rule base containing all possible $\wedge-$ combinations of the premise terms.
3. Use multiplication $(*)$ for the $\wedge-$ connective.
4. Use conclusion singletons, positioned at the sum of the peak positions of the premise sets.
5. Use sum-accumulation and *COGS* defuzzification.

With these design choices the control surface is a diagonal plane (Figure 3.11). Such a fuzzy controller is two controllers in one: (1) it has the design of a fuzzy controller, and (2) it is equivalent to a summation. It has a transfer function, and the usual methods for tuning and calculating stability of the closed-loop system apply.

3.5 Analytical Simplification of the Inference*

Given the conditions in the form of design choices mentioned earlier, the inferred output has a simple analytical expression, even for the nonlinear case. We shall use the shorthand

*Can be skipped in a first reading.

notation

> N_E for $\mu_{Neg}(E)$ corresponding to 'error is Neg',
>
> Z_E for $\mu_{Neg}(E)$ corresponding to 'error is Zero',
>
> P_E for $\mu_{Neg}(E)$ corresponding to 'error is Pos',
>
> N_{CE} for $\mu_{Neg}(CE)$ corresponding to 'change in error is Neg',
>
> Z_{CE} for $\mu_{Neg}(CE)$ corresponding to 'change in error is Zero'
>
> P_{CE} for $\mu_{Neg}(CE)$ corresponding to 'change in error is Pos', and
>
> S_i for the singleton of rule i.

Four rules

Take the rule base with four rules,

> If *error* is Neg and *change in error* is Neg then *control* is NB
>
> If *error* is Neg and *change in error* is Pos then *control* is Zero
>
> If *error* is Pos and *change in error* is Neg then *control* is Zero
>
> If *error* is Pos and *change in error* is Pos then *control* is PB

corresponding to rules 1, 3, 7, and 9 in (3.2). The rule base is constructed in accordance with design choice 2. The controller output is by design choice 5 and Equation (3.6),

$$u = \frac{\sum_k \alpha_k^* S_k}{\sum_k \alpha_k^*}$$

where S_k is the position of the conclusion singleton of rule k, and α_k^* is the firing strength of rule k. The expression is the activation weighted average of the conclusion singletons. For example, the first rule ($k = 1$) is activated to the degree

$$\alpha_1^* = N_E * N_{CE}$$

by design choice 3. Singleton $S_1 = s_{NB}$, and the contribution from the first rule to the numerator is

$$\alpha_1^* * s_{NB}$$

by design choices 4 and 5. Similar results can be derived for the remaining rules. We can in fact write the inferred controller output directly.

The numerator is,

$$\begin{aligned}\sum_{k=1}^{4} \alpha_k^* S_k = {} & N_E * N_{CE} * s_{NB} + N_E * P_{CE} * s_{Zero} \\ & + P_E * N_{CE} * s_{Zero} + P_E * P_{CE} * s_{PB}\end{aligned}$$

Singleton s_{Zero} is 0 by design choice 4, therefore two terms vanish. Furthermore, $s_{NB} = -s_{PB}$, and by design choice 1, $N(x) = 1 - P(x)$ where x is either E or CE. The numerator can therefore be reduced,

$$\begin{aligned}\sum_{k=1}^{4} \alpha_k^* S_k &= N_E * N_{CE} * s_{NB} + P_E * P_{CE} * s_{PB} \\ &= (1 - P_E)(1 - P_{CE}) * (-s_{PB}) + P_E * P_{CE} * s_{PB} \\ &= (P_E + P_{CE} - 1) * s_{PB}\end{aligned}$$

The denominator is

$$\sum_{k=1}^{4} \alpha_k^* = N_E * N_{CE} + N_E * P_{CE} + P_E * N_{CE} + P_E * P_{CE}$$

Or,

$$\begin{aligned}\sum_{k=1}^{4} \alpha_k^* &= (1 - P_E) * (1 - P_{CE}) + (1 - P_E) * P_{CE} \\ &\quad + P_E * (1 - P_{CE}) + P_E * P_{CE} \\ &= 1\end{aligned}$$

In summary, the controller output can be written

$$u = (P_E + P_{CE} - 1) * s_{PB}$$

Singleton $s_{PB} = 200$ when using standard universes, and the final expression is clearly linear when P_E and P_{CE} are linear. The expression is valid even for nonlinear P_{CE} and P_E, that is, nonlinear fuzzy membership functions μ_{Pos} and μ_{Neg}, as long as $\mu_{Pos}(x) + \mu_{Neg}(x) = 1$.

Nine rules

For the rule base with nine rules (3.2) we can derive a similar result, only slightly more complex.

Again we note that the rule base is constructed in accordance with design choice 2. The numerator is, in this case,

$$\begin{aligned}\sum_{k=1}^{9} \alpha_k^* S_k &= N_E * N_{CE} * s_{NB} + N_E * Z_{CE} * s_{NM} + N_E * P_{CE} * s_{Zero} \\ &\quad + Z_E * N_{CE} * s_{NM} + Z_E * Z_{CE} * s_{Zero} + Z_E * P_{CE} * s_{PM} \\ &\quad + P_E * N_{CE} * s_{Zero} + P_E * Z_{CE} * s_{PM} + P_E * P_{CE} * s_{PB}\end{aligned}$$

Singleton s_{Zero} is 0 by design choice 4, therefore three terms vanish. Furthermore, $s_{NB} = -s_{PB}$, $s_{NM} = -0.5 * s_{PB}$, and $s_{PM} = 0.5 * s_{PB}$. By design choice 1, we have $Z(x) = 1 -$

$(P(x)+N(x))$, where x is either E or CE. The numerator reduces to,

$$\begin{aligned}\sum_{k=1}^{9}\alpha_k^* S_k &= N_E * N_{CE} * s_{NB} + N_E * Z_{CE} * s_{NM}\\ &\quad + Z_E * N_{CE} * s_{NM} + Z_E * P_{CE} * s_{PM}\\ &\quad + P_E * Z_{CE} * s_{PM} + P_E * P_{CE} * s_{PB}\\ &= N_E * N_{CE} * (-s_{PB})\\ &\quad + N_E * (1-(P_{CE}+N_{CE})) * (-\tfrac{1}{2} * s_{PB})\\ &\quad + (1-(P_E+N_E)) * N_{CE} * (-\tfrac{1}{2} * s_{PB})\\ &\quad + (1-(P_E+N_E)) * P_{CE} * \tfrac{1}{2} * s_{PB}\\ &\quad + P_E * (1-(P_{CE}+N_{CE})) * \tfrac{1}{2} * s_{PB}\\ &\quad + P_E * P_{CE} * s_{PB}\\ &= \tfrac{1}{2}\,(P_E - N_E + P_{CE} - N_{CE})\, s_{BP}\end{aligned}$$

The denominator is

$$\begin{aligned}\sum_{k=1}^{9}\alpha_k^* &= N_E * N_{CE} + N_E * Z_{CE} + N_E * P_{CE}\\ &\quad + Z_E * N_{CE} + Z_E * Z_{CE} + Z_E * P_{CE}\\ &\quad + P_E * N_{CE} + P_E * Z_{CE} + P_E * P_{CE}\\ &= N_E * N_{CE} + N_E * (1-(P_{CE}+N_{CE})) + N_E * P_{CE}\\ &\quad + (1-(P_E+N_E)) * N_{CE}\\ &\quad + (1-(P_E+N_E)) * (1-(P_{CE}+N_{CE}))\\ &\quad + (1-(P_E+N_E)) * P_{CE}\\ &\quad + P_E * N_{CE} + P_E * (1-(P_{CE}+N_{CE})) + P_E * P_{CE}\\ &= 1\end{aligned}$$

In summary, the controller output can be written

$$u = \tfrac{1}{2}\,(P_E - N_E + P_{CE} - N_{CE})\, s_{PB}$$

Singleton $s_{PB} = 200$ when using standard universes, and the final expression is clearly linear when P_{CE}, P_E, N_{CE}, and N_E are linear. The expression is valid even for nonlinear P_{CE}, P_E, N_{CE}, and N_E, that is, nonlinear fuzzy membership functions μ_{Pos}, μ_{Zero} and μ_{Neg}, as long as $\mu_{Pos}(x) + \mu_{Zero}(x) + \mu_{Neg}(x) = 1$.

3.6 Summary

In a fuzzy controller the measurement data passes through a pre-processing block, a controller, and a post-processing block. Pre-processing consists of a linear or nonlinear scaling, as well as a quantization when using discrete membership functions (vectors). When using continuous membership functions, the membership of each input measurement is looked up in a function.

With all other choices being equal, it is recommended to apply continuous premise membership functions, but discrete conclusion membership functions, preferably singletons.

When designing the rule base, the designer must consider the number of term sets, their shape, and their overlap. The rules themselves must be established, unless more advanced means such as adaptation are available. There is a choice between multiplication and minimum in the activation function. There is also a choice regarding defuzzification. The post-processing consists in a scaling of the control action to engineering units. The following is a checklist of design choices to make:

Rule base–related choices. Number of inputs and outputs, rules, universes, continuous or discrete, the number of membership functions, their overlap and width, singleton conclusions.

Inference engine–related choices. Connectives, modifiers, activation operation, aggregation operation, and accumulation operation.

Defuzzification method. COG, COGS, BOA, MOM, LM, and RM.

Pre- and post-processing. Scaling, gain factors, quantization, and sampling time.

Some of these items must always be considered, others may not play a role in a particular design.

The control surface provides insight, because it is a picture of the input–output characteristics of the controller; it is the central object that governs the behaviour of the controller. Changing the membership functions causes the control surface to change shape. A linear control surface settles several design choices and opens up for a fuzzy proportional-integral-derivative (PID) type of control, as the next chapter shows.

3.7 Notes and References

Theory The overview by Lee (1990) is still a good starting point to gain insight into the inner workings of fuzzy controllers. The most efficient way to learn is to study the Fuzzy Logic Toolbox for Matlab, especially the Fuzzy Inference System (Mathworks 2006). The book by Driankov *et al.* (1996) is explicitly aimed at the control engineering community, in particular, engineers in industry and university students, with the intention of covering just the relevant parts of the theory and focusing on principles rather than particular applications or tools. Another central reference is the book by Passino and Yurkovich (1998), which provides case studies and a wide coverage of the control area, including identification, adaptive control, and supervisory control. Another wide coverage of the control area is the book by Wang (1997). Fuzzy model identification is treated thoroughly by Babuska (1998), who also offers a related Matlab toolbox for download[1]. Another central reference for modelling is Jang *et al.* (1997), which combines neural networks and machine learning in the so-called soft

[1]http://www.dcsc.tudelft.nl/~babuska/klbook.htm

computing approach. Model-based fuzzy control with gain scheduling and sliding mode control is treated by Palm *et al.* (1997). Fuzzy control has been merged with the learning capabilities of artificial neural networks into neuro-fuzzy control (e.g. Nauck *et al.* 1997, Lin and Lee 1996) or intelligent control systems (Gupta and Sinha 1996), and with further techniques such as genetic algorithms into soft computing (Jang *et al.* 1997). For an overview of theoretically oriented work, see the collection edited by Farinwata *et al.* (2000). For future perspectives see the article by Sala *et al.* (2005).

Applications The book by Constantin von Altrock (1995) describes case studies related to appliances (air conditioning, heating, washing machine, clothes dryer), the automotive industry (brakes, engine, transmission, skidding, air conditioning), process control (decanter, incineration, ethylene production, cooling, waste water, food processing), and other applications (battery charger, optical disk drive, camcorder, climate control, elevator, camera, anaesthesia, aircraft landing). Many related reports are easily accessible from the Fuzzy Application Library on the World Wide Web site[2] of the software product fuzzyTECH. The collection of intelligent control systems (Gupta and Sinha 1996) mentioned here presents applications within robot control, adaptive control, knowledge-based systems, robust control, expert systems, and discrete event systems. There is an international standard for Programmable Controllers defining programming methodology, environment, and functional characteristics (IEC 2000).

[2]http://www.fuzzytech.com

4

Linear Fuzzy PID Control

A fuzzy PID controller is a fuzzified proportional-integral-derivative (PID) controller. It acts on the same input signals, but the control strategy is formulated as fuzzy rules.

If a control engineer changes the rules, or the tuning gains to be discussed later, it is difficult to predict the effect on rise time, overshoot, and settling time of a closed-loop step response, because the controller is generally nonlinear and its structure is complex.

In contrast, a PID controller is a simple, linear combination of three signals: the P-action proportional to the error e, the I-action proportional to the integral of the error $\int e\mathrm{d}t$, and the D-action proportional to the time derivative of the error $\mathrm{d}e/\mathrm{d}t$, or $\dot{e}$ for short. This chapter introduces a systematic tuning procedure for fuzzy PID type controllers.

Assume that the control objective (Figure 4.1) is to control the controlled output $y = x + n$, which includes measurement noise n, around a reference input *Ref*, either after a change in the reference, a change in the load l, or due to the noise n. Assume for simplicity that the plant is monotonous such that the plant output x increases when the plant input $u + l$ increases. The controller must increase the control signal u when the controlled output y is below the reference and decrease u when the controlled output y is above the reference. This is *negative* feedback since the control signal moves in the *opposite* direction of the plant output.

It is natural to choose the error $e = Ref - y$ as an input to a fuzzy controller, as in PID control, and let the controller act on the magnitude and the sign of e. It follows that the integral of the error and the derivative of the error are useful signals to act on as well. A simple fuzzy control strategy with only four rules, based on error and its derivative, is

1. If *error* is Neg and *change in error* is Neg then *control* is NB
2. If *error* is Neg and *change in error* is Pos then *control* is Zero (4.1)
3. If *error* is Pos and *change in error* is Neg then *control* is Zero
4. If *error* is Pos and *change in error* is Pos then *control* is PB

The premise variable *error* is proportional to e, *change in error* is proportional to $\mathrm{d}e/\mathrm{d}t = \dot{e}$, and the conclusion variable *control* is proportional to the control signal. We shall distinguish

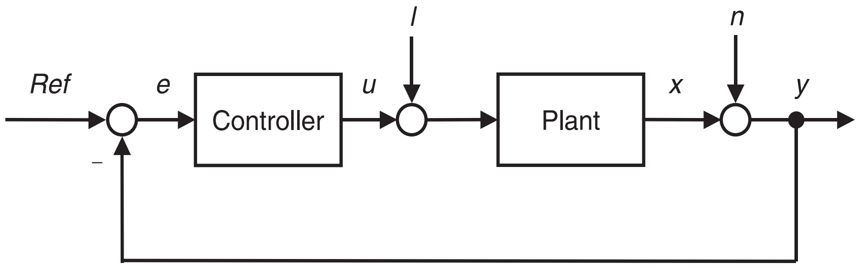


Figure 4.1: Feedback loop with load l and noise n.

between *error* and e, and *change in error* and de/dt; when possible we reserve the terms *error* and *change in error* for rule bases and e and de/dt for the signals derived from $e = Ref - y$; the difference in each case is a gain factor. The names Pos and Neg are labels of fuzzy sets, and likewise NB (negative big), Zero (control signal is zero), and PB (positive big).

There are methods for tuning PID controllers, for example: hand-tuning, Ziegler–Nichols tuning, optimal design, pole placement design, and auto-tuning (Åström and Hägglund 1995). There is much to gain, if these methods are carried forward to fuzzy controllers.

Fuzzy PID controllers are similar to PID controllers under certain assumptions about the shape of the membership functions and the inference method (Siler and Ying 1989, Mizumoto 1992, Qiao and Mizumoto 1996, Tso and Fung 1997). A design procedure for fuzzy controllers of the PID type, based on PID tuning, is the following:

Procedure Design fuzzy PID

1. Build and tune a conventional PID controller first.
2. Replace it with an equivalent linear fuzzy controller.
3. Make the fuzzy controller nonlinear.
4. Fine-tune it.

The idea is to start the controller design with a crisp PID controller, stabilize the closed-loop system, and tune it to a satisfactory performance. With a linear controller, and given a linear model of the plant, it is even possible at this stage to carry out stability calculations, for instance: gain margins, eigenvalues, and Nyquist plots. From the solid foundation of linear control theory, it is safer to move to fuzzy control, rather than starting from scratch. Such a procedure has a scope limited by PID control:

Scope The procedure is relevant whenever PID control is possible, or already implemented.

Our starting point is the ideal continuous PID controller,

$$u = K_p \left(e + \frac{1}{T_i} \int e(t)\,\mathrm{d}t + T_d \frac{\mathrm{d}e}{\mathrm{d}t} \right). \tag{4.2}$$

The control signal u is a linear combination of the error e, its integral and its derivative. The parameter K_p is the *proportional gain*, T_i is the *integral time*, and T_d the *derivative time*.

In digital controllers, the equation must be approximated. Replacing the derivative term by a backward difference and the integral by a sum using rectangular integration, and given a constant – preferably small – sampling time T_s, the simplest approximation is,

$$u(n) = K_p \left(e(n) + \frac{1}{T_i} \sum_{j=1}^{n} e(j) T_s + T_d \frac{e(n) - e(n-1)}{T_s} \right) \tag{4.3}$$

Index n refers to the time instant. By *tuning* we shall mean the activity of adjusting the parameters K_p, T_i, and T_d in order to achieve a good closed-loop performance.

4.1 Fuzzy P Controller

In discrete time, a proportional controller is defined by

$$u(n) = K_p e(n) \tag{4.4}$$

It is derived from the PID controller in Equation (4.3) with the I-action set to zero ($1/T_i = 0$) and the D-action set to zero ($T_d = 0$). The *fuzzy proportional* (FP) controller in the block diagram in Figure 4.2 accordingly acts on the error e, and its control signal is U.

Signals are represented by lower case symbols before gains and upper case symbols after gains. Thus the notation E represents the term *error,* and $E = GE * e$ (the symbol $*$ is multiplication), and u represents *control,* where $GU * u = U$.

The FP controller has two tuning gains GE and GU, where the crisp proportional controller has just one, K_p. The control signal $U(n)$, at the time instant n is generally a nonlinear function of the input $e(n)$,

$$U(n) = f(GE * e(n)) * GU \tag{4.5}$$

The function f denotes the rule base mapping. It is generally nonlinear, as mentioned; but with a favourable choice of design, a linear approximation is

$$f(GE * e(n)) \approx GE * e(n) \tag{4.6}$$

Insertion into Equation (4.5) yields the control signal

$$U(n) = GE * e(n) * GU = GE * GU * e(n) \tag{4.7}$$

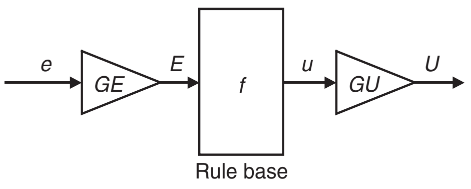


Figure 4.2: Fuzzy proportional controller, FP.

Comparing with Equation (4.4) the product of the gain factors for the linear controller corresponds to the proportional gain,

$$GE * GU = K_p \tag{4.8}$$

The linear approximation is exact if, firstly, we choose the same universe for premise sets and conclusion sets, for example, percentages of full scale [−100, 100]. Secondly, the rule base

$$\begin{aligned}&1.\ \text{If } E(n) \text{ is Pos then } u(n) \text{ is } 100\\ &2.\ \text{If } E(n) \text{ is Zero then } u(n) \text{ is } 0\\ &3.\ \text{If } E(n) \text{ is Neg then } u(n) \text{ is } -100\end{aligned} \tag{4.9}$$

with Pos, Zero, and Neg implemented appropriately – that is, according to the conditions in the previous chapter – provides a linear input–output mapping. The controller is thus equivalent to a crisp P controller.

Given a target proportional gain K_p – from a tuned, crisp P controller – Equation (4.8) determines one fuzzy gain factor when the other is chosen. The equation has one degree of freedom, since the fuzzy P controller has one more gain factor to adjust than the crisp P controller. This can be used to exploit the full range of the premise universe. For example, assume that the maximal reference step is 1, whereby the maximal $e(n)$ is 1, and assume the universe for $E(n)$ is [−100, 100], then GE should be close to 100. When GE is chosen, Equation (4.8) determines GU.

4.2 Fuzzy PD Controller

Because of the plant dynamics, it will take some time before a change in the control signal is noticeable in the plant output, and the proportional controller will be equally late in correcting for an error. Derivative action helps to predict the future error, and the PD controller uses the derivative action to improve closed-loop stability. The discrete time PD controller is,

$$u(n) = K_p\left(e(n) + T_d\frac{e(n) - e(n-1)}{T_s}\right) \tag{4.10}$$

by Equation (4.3) with the I-action set to zero $(1/T_i = 0)$.

The second term in the parenthesis is proportional to an estimate of the error, T_d seconds ahead of the time instant n, where the estimate is obtained by linear extrapolation of the straight line connecting $e(n-1)$ and $e(n)$.

With $T_d = 0$ the controller is purely proportional, but when T_d is gradually increased, it will dampen possible oscillations. If T_d is increased too much the step response of the closed-loop system becomes *overdamped,* and it will start to oscillate again.

Input to the *fuzzy proportional-derivative* (FPD) controller in Figure 4.3 is $e(n)$ and $\dot{e}(n)$, where

$$\dot{e}(n) \approx \frac{e(n) - e(n-1)}{T_s} \tag{4.11}$$

The backward difference is a simple discrete approximation to the differential quotient, and more accurate digital implementations are available (e.g. Åström and Hägglund 1995,

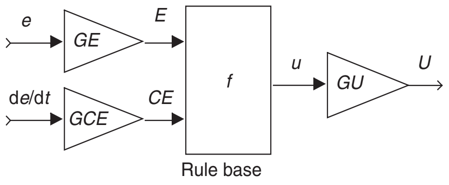


Figure 4.3: Fuzzy PD controller, FPD.

p. 93). The notation CE represents the term *change in error,* and $CE = GCE * \dot{e}$. Notice that Equation (4.11) deviates from the straight difference $e(n) - e(n-1)$ used in the early days of fuzzy control.

The control signal $U(n)$, at the time instant n, is a nonlinear function of *error* and *change in error*,

$$U(n) = f(GE * e(n), GCE * \dot{e}(n)) * GU \tag{4.12}$$

Again the function f is the rule base mapping, only this time it is a surface depending on two variables. It is usually nonlinear, but with a favourable choice of design, a linear approximation is

$$f(GE * e(n), GCE * \dot{e}(n)) \approx GE * e(n) + GCE * \dot{e}(n) \tag{4.13}$$

Insertion into Equation (4.12) yields the control action for the linear controller,

$$U(n) = (GE * e(n) + GCE * \dot{e}(n)) * GU \tag{4.14}$$

$$= GE * GU * (e(n) + \frac{GCE}{GE}\dot{e}(n)) \tag{4.15}$$

Comparing Equations (4.10) and (4.15), the gains are related as follows:

$$GE * GU = K_p \tag{4.16}$$

$$\frac{GCE}{GE} = T_d \tag{4.17}$$

The linear approximation is exact when the fuzzy control surface is a plane acting like a summation; compare the conditions in the previous chapter. Thus the rule base (4.1), with Pos and Neg implemented appropriately, provides a linear input–output mapping. The conclusion universe must be defined as the sum of the premise universes. Assume, for instance, that the premise universes are both $[-100, 100]$, and we choose singleton conclusions $NB = -200$ and $PB = 200$, then the control surface will be the plane $u(n) = E(n) + CE(n)$. By that choice, the controller is equivalent to a crisp PD controller, and we can exploit Equations (4.16) and (4.17).

The fuzzy PD controller may be applied when proportional control is inadequate. The derivative term reduces overshoot, but it may be sensitive to noise as well as abrupt changes of the reference causing *derivative kick* in Equation (4.11).

4.3 Fuzzy PD+I Controller

If the closed-loop system exhibits a sustained error in steady state, integral action is necessary. The integral action will increase (decrease) the control signal if there is a positive (negative) error, even for small magnitudes of the error. Thus, a controller with integral action will always return to the reference in steady state.

A *fuzzy PID controller* acts on three inputs: *error*, *integral error*, and *change in error*. A rule base with three premise inputs can be a problem. With three premise inputs, and, for example, three linguistic terms for each input, the complete rule base consists of $3^3 = 27$ rules, making it cumbersome to maintain. Furthermore, it is difficult to settle on rules concerning the integral action, because the initial and final values of the integral depends on the load l. The integral action in the crisp PID controller serves its purpose, however, and a simple design is to combine crisp integral action and a fuzzy PD rule base in the *fuzzy PD+I* (FPD+I) controller (see Figure 4.4).

The integral error $IE = GIE \int e\mathrm{d}t$ is proportional to the accumulation of all previous error measurements in discrete time, with

$$\int e\mathrm{d}t \approx \sum_{j=1}^{n} e(j)T_s$$

Rectangular integration is a simple approximation to the integral, and more accurate approximations exist (e.g. Åström and Hägglund 1995, p 93). The control signal $U(n)$ after the gain GU, at the time instant n, is a nonlinear function of *error*, *change in error*, and *integral error*,

$$U(n) = \left[f(GE * e(n), GCE * \dot{e}(n)) + GIE \sum_{j=1}^{n} e(j)T_s \right] * GU \tag{4.18}$$

The function f is again the control surface of a PD rule base. The mapping is usually nonlinear, but with a favourable choice of design, a linear approximation is Equation (4.13). Insertion into Equation (4.18) yields the control action,

$$\begin{aligned} U(n) &\approx \left[GE * e(n) + GCE * \dot{e}(n) + GIE \sum_{j=1}^{n} e(j)T_s \right] * GU \\ &= GE * GU * \left[e(n) + \frac{GCE}{GE} * \dot{e}(n) + \frac{GIE}{GE} \sum_{j=1}^{n} e(j)T_s \right] \end{aligned} \tag{4.19}$$

In the last line we have assumed that GE is non-zero. Comparing Equations (4.3) and (4.19) the gains are related as follows:

$$GE * GU = K_p \tag{4.20}$$

$$\frac{GCE}{GE} = T_d \tag{4.21}$$

$$\frac{GIE}{GE} = \frac{1}{T_i} \tag{4.22}$$

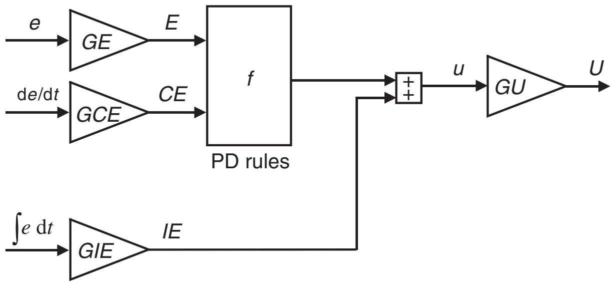


Figure 4.4: Fuzzy PID controller, FPD+I.

The FPD+I controller provides all the benefits of PID control, but also the disadvantages regarding derivative kick. The integral error removes any steady state error, but can also cause *integrator windup*.

4.4 Fuzzy Incremental Controller

An incremental controller adds a *change* in control signal Δu to the current control signal,

$$\begin{aligned} u(n) &= u(n-1) + \Delta u(n) T_s \Rightarrow \\ \Delta u(n) &= K_p \left(\frac{e(n) - e(n-1)}{T_s} + \frac{1}{T_i} e(n) \right) \end{aligned}$$

using Equation (4.3) with $T_d = 0$. The controller output is an increment to the current control signal.

The *fuzzy incremental* (FInc) controller in Figure 4.5 is of almost the same configuration as the FPD controller except for the added integrator. The conclusion in the rule base is now called *change in output* (*cu*), and the gain on the output is, accordingly, *GCU*. The control signal $U(n)$ at time instant n is the sum of all previous increments,

$$\begin{aligned} U(n) &= \sum_{j=1}^{n} (cu(j) * GCU * T_s) \\ &= \sum_{j=1}^{n} (f(GE * e(j), GCE * \dot{e}(j)) * GCU * T_s) \end{aligned} \tag{4.23}$$

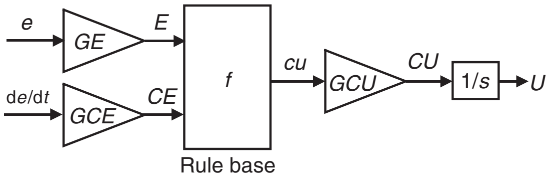


Figure 4.5: Incremental fuzzy controller, FInc.

Notice again that this definition deviates from the historical fuzzy controllers, where the sampling period T_s was left out. The function f is again the control surface of a PD rule base. The mapping is usually nonlinear, but with the usual favourable choice of design, Equation (4.13) is a linear approximation. Insertion into Equation (4.23) yields the control action,

$$\begin{aligned}
U(n) &\approx \sum_{j=1}^{n}(GE * e(j) + GCE * \dot{e}(j)) * GCU * T_s \\
&= GCU * \sum_{j=1}^{n}\left[GE * e(j) + GCE * \frac{e(j) - e(j-1)}{T_s}\right] * T_s \\
&= GCU * \left[GE * \sum_{j=1}^{n} e(j) * T_s + GCE * \sum_{j=1}^{n}(e(j) - e(j-1))\right] \\
&= GCE * GCU * \left[\frac{GE}{GCE}\sum_{j=1}^{n}(e(j) * T_s) + e(n)\right]
\end{aligned} \tag{4.24}$$

By comparing Equations (4.3) and (4.24) it is clear that the linear controller is a crisp PI controller ($T_d = 0$), and the gains are related as follows:

$$GCE * GCU = K_p$$
$$\frac{GE}{GCE} = \frac{1}{T_i}$$

Notice that the proportional gain K_p now depends on *GCE*. The gain $1/T_i$ is determined by the ratio between the two fuzzy input gains, and is the inverse of the derivative gain T_d in FPD control; the gains *GE* and *GCE* change roles in FPD and FInc controllers.

It is an advantage that the controller output is driven directly from an integrator, because (1) simply limiting the integrator prevents integrator windup, and (2) the integrator cancels noise to an extent that smooths the control signal.

To summarize, Table 4.1 shows for each of the four controller types the relationships between the PID tuning parameters and fuzzy gain factors valid for fuzzy linear controllers acting like a summation.

Table 4.1: Relationships between linear fuzzy and PID gains.

Controller	K_p	$1/T_i$	T_d
FP	$GE * GU$		
FInc	$GCE * GCU$	GE/GCE	
FPD	$GE * GU$		GCE/GE
FPD+I	$GE * GU$	GIE/GE	GCE/GE

4.5 Tuning

Several tuning aspects may be illustrated by static considerations (Åström and Hägglund 1995). For purely proportional control, consider the feedback loop in Figure 4.1, where the controller has the proportional gain K_p and the plant has the gain K in steady state. The plant output x is related to the reference *Ref*, the load l, and the measurement noise n by the equation

$$x = \frac{K_p K}{1 + K_p K}(Ref - n) + \frac{K}{1 + K_p K} l$$

If n and l are zero, then K_p should be high in order to ensure that the plant output x is close to the reference *Ref*. Furthermore, if the load l is non-zero, a high value will make the system less sensitive to changes in the load. But if the noise n is non-zero, K_p should be moderate – otherwise the system will be too sensitive to noise. If the plant dynamics are considered, the closed-loop system will normally be unstable if K_p is high. Obviously the tuning of K_p is a balance between the control objectives: stability, noise sensitivity, reference following, and load regulation.

Ziegler–Nichols tuning

A PID controller may be tuned using the *Ziegler–Nichols frequency response method* (Ziegler and Nichols in Åström and Hägglund 1995).

Procedure Ziegler–Nichols frequency response method

1. Increase the proportional gain until the system oscillates; the resulting gain is the ultimate gain K_u.
2. Read the time between peaks T_u at this setting.
3. Use Table 4.2 to achieve approximate values for the controller gains.

The sample period may be related to the derivative gain T_d. Åström and Wittenmark (1984) suggest that the sample period should be between 1/10 and 1/2 of T_d. Combining with the Ziegler–Nichols rules, T_s should be approximately equal to 1–5 % of the ultimate period T_u. Another rule of thumb is that T_s should be chosen to be slightly smaller than the dominating time constant in the plant, for instance, between 1/10 and 1/5 of that time constant.

Table 4.2: The Ziegler–Nichols rules (frequency response method).

Controller	K_p	T_i	T_d
P	$0.5K_u$		
PI	$0.45K_u$	$T_u/1.2$	
PID	$0.6K_u$	$T_u/2$	$T_u/8$

Ziegler and Nichols also give another method called the *reaction curve* or *step response* method (see, e.g., Åström & Hägglund 1995). That method uses the open-loop step response to find the gains, and this is an advantage if oscillations in the closed-loop system should be avoided for safety reasons.

Example 4.5.1 *Ziegler–Nichols frequency response method*
Assume the plant in Figure 4.1 has the transfer function

$$G(s) = \frac{1}{(s+1)^3}$$

Insert a PID controller with differential and integral action removed by setting $T_d = 0$ *and* $1/T_i = 0$. *Gradually increase the proportional gain until the closed-loop system reaches a stable oscillation (Figure 4.6). This gain is* $K_u = 8$ *and the ultimate period, read from the plot, is 4 peaks in about 15 seconds, or* $T_u = 15/4$.

There is a unit load on the system, therefore, the controller must employ integral action. The third row in Table 4.2 implies $K_p = 0.6 * K_u = 4.8$, $T_i = T_u/2 = 15/8$, *and* $T_d = T_u/8 = 15/32$.

Figure 4.7 shows the closed-loop response after a step in the reference at time equal to zero, and a step in the load at time equal to 20 seconds. The initial overshoot is fairly large and the load response is slightly overdamped.

Ziegler and Nichols aimed at a response to a load change with a *decay ratio* of one quarter. Decay ratio is the ratio between two consecutive peaks of the error after a step change in reference or load; thus in a quarter-decay response the second overshoot is 25 % of the first – a compromise between a fast response and a small overshoot. The relationships of Table 4.2 therefore do not fit all situations.

The results are poor for systems with a time lag much greater than the dominating time constant. In general, the rules often result in rather poor damping. The table generally works better for PID control than for PI control, and it does not give guidance for PD control. A related, more accurate, but also slightly more cumbersome, method is *Kappa-Tau tuning* based on the dimensionless parameters: the relative gain Kappa (κ) and the relative deadtime Tau (τ) (Åström and Hägglund 1995, p. 217, Åström *et al.* 1992).

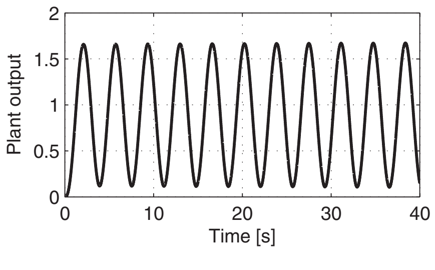


Figure 4.6: Ziegler–Nichols oscillation of plant $1/(1+s)^3$. (figzn.m)

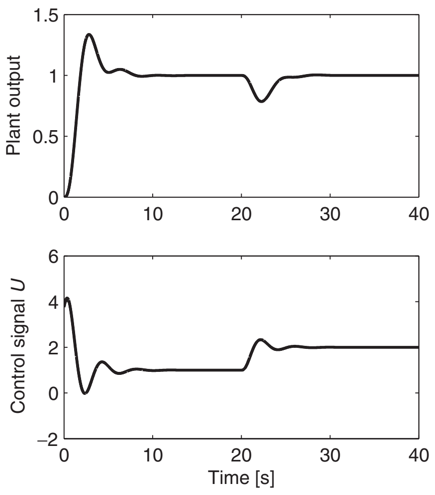


Figure 4.7: PID control of the plant $1/(1+s)^3$. A reference step at 0 seconds is followed by a load step at 20 seconds. Ziegler-Nichols settings: $K_p = 4.8$, $T_i = 15/8$, and $T_d = 15/32$. (figzn.m)

Hand-tuning

Hand-tuning is based on rules of thumb used by experienced process engineers (see Table 4.3). The tuning is a compromise between fast reaction and stability. There are exceptions to the rules in the table; if the plant contains an integrator, an increase in K_p often results in more stable control. The following is a hand-tuning procedure adapted from Smith (1979).

Procedure Hand-tuning

1. Remove all integral and derivative action by setting $T_d = 0$ and $1/T_i = 0$.
2. Tune the proportional gain K_p to give the desired response, ignoring any final value offset from the setpoint.
3. Increase the proportional gain further and adjust the derivative gain T_d to dampen the overshoot.

Table 4.3: Rules of thumb for tuning PID controllers.

Action	Rise time	Overshoot	Stability
Increase K_p	Faster	Increases	Decreases
Increase T_d	Slower	Decreases	Increases
Increase $1/T_i$	Faster	Increases	Decreases

4. Adjust the integral gain $1/T_i$ to remove any final value offset.
5. Repeat from step 3 until the proportional gain K_p is as large as possible.

The procedure adjusts the derivative gain before the integral gain, but in practice the sequence may be reversed. A process engineer can use the procedure right away, online, and develop a feel for how the closed-loop system behaves. A disadvantage is that it may take a long time to develop this feel, and it is difficult to sense whether the final settings are optimal.

Example 4.5.2 *Load requires integrator*
In a water rig the control objective is to adjust the water level in a tank after a step in the reference. The rig consists of a feed pump and a tank, and the tank has an outlet.

(a) Assume first that the outlet is closed. For this problem a P controller is sufficient. As soon as the water reaches the setpoint level, the error becomes zero, and the controller will stop the feed pump.

(b) If there is overshoot, for example, if the feed pump reacts sluggishly, the controller should slow down the pump well before the water reaches the setpoint. A PD controller is then appropriate.

(c) Assume now that the outlet is open. The controller must try and reach the setpoint and keep pumping to compensate for the water running out of the outlet. A sustained control signal in steady state is necessary to balance the outflow. Thus, integral action is necessary and a PI or PID controller will be appropriate.

The second step in the design procedure is to replace the PID controller with an equivalent linear fuzzy controller, in accordance with the requirements for linearity from the previous chapter, and gain factors in accordance with Table 4.1. The closed-loop system should show exactly the same step response; the implementation is correct when the fuzzy rule base can be taken out and replaced with a pure summation block.

Table 4.4 summarizes advantages and disadvantages of all four fuzzy controllers. The fuzzy P controller may be used as a starting point. To improve the settling time and reduce overshoot, fuzzy PD is the choice. If there is a steady state error, a FInc controller or a fuzzy PD+I is the choice.

Table 4.4: Quick reference to controller characteristics.

Controller	Advantage	Disadvantage
FP	Simple	Maybe too simple
FPD	Less overshoot	Noise sensitive, derivative kick
FInc	Removes steady state error, smooths control signal	Slow
FPD+I	All in one	Windup, derivative kick

To emphasize, the controllers with f replaced by a summation are linear approximations to the corresponding fuzzy configurations; the relations hold for the *approximations* only. Also, for fixed universe controllers, the conclusion universe must be the sum of the premise universes. With premise universes $[-100, 100]$, for instance, the conclusion universe of an FPD controller should be $[-200, 200]$.

Example 4.5.3 *Gain transfer in other implementations*
What if we come across controller implementations other than the above? How are the gains related then?

(a) In a particular fuzzy PD controller $\dot{e}$ is implemented as a straight difference $\Delta e = e(n) - e(n-1)$. Comparing Equations (4.3) and (4.14) implies $(GCE/GE) * \Delta e = T_d * \Delta e/T_s$, and this implies in turn $T_d = (GCE/GE) * T_s$. Similarly with the FPD+I controller. Then the last column in Table 4.1 should be multiplied by T_s. As a consequence an increase in the sampling period will increase the differential time.

(b) In a particular FInc controller $\dot{e} = \Delta e/T_s$, but $U(n) = \sum u_i * GCU$ (without the multiplication by T_s). Then (4.24) must be modified to

$$U(n) = \frac{GCE}{T_s} * GCU * \left[\frac{GE}{GCE} \sum_{i=1}^{n} e(i) * T_s + e(n) \right]$$

Comparison with (4.3) yields $K_p = GCE * GCU/T_s$, and the integral time is unchanged. As a consequence an increasing sampling period implies a decreasing proportional gain.

(c) A particular fuzzy PD has the premise and conclusion universes $[-100, 100]$. *The linear controller is equivalent to the usual linear approximation, but with half the output gain. Thus Table 4.1 must be used with $GU/2$ instead of GU. The general rule is to use the table with GU/r (or GCU/r) where r is the number of inputs when the conclusion universe equals the premise universes.*

(d) The PID controller can be given on the so-called parallel form,

$$u = K_p e + K_i \int e(t) * \mathrm{d}t + K_d \frac{\mathrm{d}e}{\mathrm{d}t}$$

where the control signal is a linear combination of three terms. The proportional gain K_p has been multiplied through in the parenthesis, such that $K_i = K_p * 1/T_i$ and $K_d = K_p * T_d$. By inspection, the gains of the linear FPD+I controller are related in the following manner: $K_p = GE * GU$, $K_i = GIE * GU$, and $K_d = GE * GU$.

4.6 Scaling

Saturation of the input signals in the premise universes upsets the linearity of the fuzzy controller. Take the third-order plant $1/(s+1)^3$ from Figure 4.7. A suitable value of the gain on *error* is $GE = 100$. But if the gain is increased to $GE = 400$, and all other gains adjusted in accordance with Table 4.1, whereby the proportional gain, differential time, and integral time remain the same, the controller saturates in the premise universes.

Scaling is a means to avoid saturation, but the relationships in Table 4.1 must still be observed in order to preserve the tuning. Consider, for example, the FPD controller in

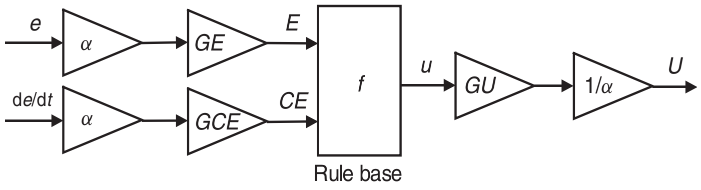


Figure 4.8: Scaling of gains in an FPD controller by means of a scaling factor α.

Equation (4.14). We may scale the input gains by a factor α without altering the tuning, because

$$\begin{aligned}&(GE * e(n) + GCE * \dot{e}(n)) * GU \\ &= (\alpha * GE * e(n) + \alpha * GCE * \dot{e}(n)) * GU * \frac{1}{\alpha}\end{aligned} \tag{4.25}$$

That is, we multiply by α on the premise side, and cancel out by $1/\alpha$ on the conclusion side. This is *not* valid for nonlinear controllers, only their linear approximations. Figure 4.8 illustrates the scaling in a block diagram. Given a linear rule base, the values of K_p and T_d are independent of α. If saturation occurs in the premise universes, α must be decreased in magnitude until signals E and CE do not saturate during a step response. The phase plot clearly shows the effect.

Scaling is introduced analogously in the other fuzzy PID controllers.

4.7 Simulation Study: Higher-Order Process

When the plant is of order higher than two, the PID controller starts to experience difficulties, and better responses can be achieved with more complex controllers. Consider therefore the third-order plant

$$G(s) = \frac{1}{(s+1)^3}$$

The plant is given a unit reference step at time $t = 0$, and a unit step on the load at time $t = 20$ seconds. We shall apply the design procedure from the introductory section of this chapter.

Step 1. Build and tune a conventional PID controller

The plant is identical to the plant in the Ziegler–Nichols example earlier. Therefore we will use the results from the Ziegler–Nichols frequency response method, that is, $K_p = 4.8$, $T_i = 15/8$, and $T_d = 15/32$. An earlier figure (Figure 4.7) shows the response. There is a load on the system, therefore integral action is required to remove steady state offset, and the third row of the Ziegler–Nichols table (Table 4.2) was used.

The response could possibly be improved by hand-tuning, but we shall settle on the Ziegler–Nichols settings for now.

Since we are in the linear domain, with a linear plant and a linear PID controller, we can plot the frequency response of the closed-loop system. The frequency response characterizes

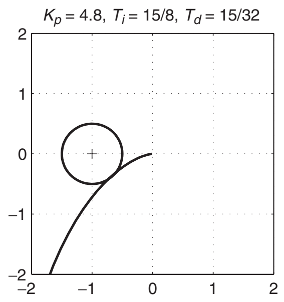


Figure 4.9: Partial Nyquist plot. The plant $G(s) = 1/(s + 1)^3$ is controlled by a PID controller with parameters $K_p = 4.8$, $T_i = 15/8$, and $T_d = 15/32$. The frequency ω increases from 0.1 to 20 towards the origin of the coordinate system. (fignyqs.m)

the dynamics by the way sine waves propagate through the system. A Nyquist plot provides a complete description of the system for the chosen input frequencies. In principle, it is determined by sending sinusoids of varying angular frequencies ω through the system, and then analyzing the frequency phase shift $\phi(\omega)$ and amplitude gain $a(\omega)$ by comparing the input with the response. It is equivalent to Bode plots of $\phi(\omega)$ and $a(\omega)$, except the information is merged into one plot instead of two with ω as an independent parameter: It is a polar plot of the complex transfer function $G_N(i\omega)$, with $a(\omega) = |G_N(i\omega)|$ the length of the locus vector and $\phi(\omega) = \quad G_N(i\omega)$ the angle.

Figure 4.9 is a Nyquist plot of the transfer function $G_N(i\omega) = G_c(i\omega)\,G(i\omega)$, where $G(i\omega)$ is the plant and $G_c(i\omega)$ the ideal controller transfer function,

$$G_c(i\omega) = Kp\left(1 + \frac{1}{T_i}\frac{1}{i\omega} + T_d i\omega\right)$$

The controller parameters are the Ziegler–Nichols parameters found previously. The additional circle around the *critical point* $(-1 + i0)$, marked by a '+' sign, has diameter 0.5. In order to achieve a good dampening of the closed-loop response, the Nyquist curve should pass by the critical point in a safe distance, which is equivalent to having closed-loop poles in a safe distance from the imaginary axis of a pole plot. A reasonable distance is in the range 0.5 to 0.77 (corresponding to the sensitivity range 2 to 1.3, Åström and Hägglund 1995, p 125). Thus the circle marks the lower limit of the range. The plot shows that with the Ziegler–Nichols settings the Nyquist curve just about touches the circle at a tangent.

The three parameters of the PID controller affect the Nyquist curve in specific manners, and by varying them a designer can in principle shape the curve. Increasing the proportional gain K_p will map a point $G_N(i\omega)$ to an image farther away from the origin, in the direction of its radius vector. Increasing the integral gain $1/T_i$ will map the same point to an image, found by adding a vector in the direction $-iG_N(i\omega)$ orthogonally towards the right seen from the origin. Increasing the derivative gain T_d will map the same point to an image, found by adding a vector in the direction $iG_N(i\omega)$ orthogonally towards the left seen from the origin.

Figure 4.10 is a *tuning map* showing the consequences of changing the Ziegler–Nichols settings by ±20 %. Since the Ziegler–Nichols settings only give approximate values, the

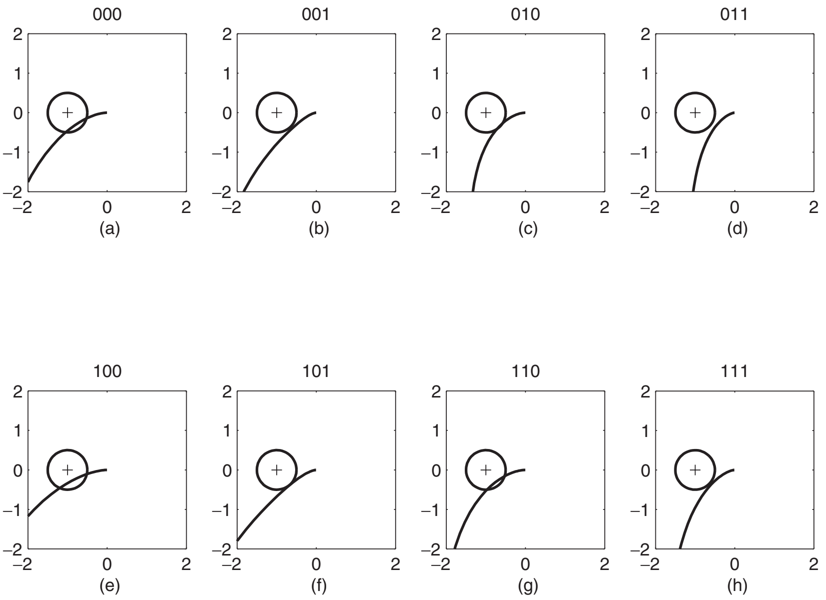


Figure 4.10: Tuning map for PID control of $G(s) = 1/(s+1)^3$. Each plot is a Nyquist curve of the transfer function round the closed loop. The three parameters K_p, T_i, and T_d are changed successively by +20 %, indicated by a 1, or −20 %, indicated by a 0, relative to the Ziegler–Nichols settings. For instance, the indicator 001 (above b) shows that K_p and T_i are decreased while T_d is increased. (fignyqs.m)

tuning map provides empirical knowledge about how changes affect the behaviour of the closed-loop system. It provides a visual overview, which can otherwise be difficult to achieve.

The figure shows that some changes are unfortunate (subfigures a, c, e, g), while one change is particularly favourable (subfigure d): decreasing K_p while increasing T_i and T_d. To interpret, it is advantageous to slow down the controller (K_p), reduce the integral action (increase T_i), and increase the damping (T_d). A test will show that this indeed results in a better response, reducing the initial overshoot to roughly one-third, and at the same time improving the creeping appearance of the load response.

Step 2. Replace it with an equivalent linear fuzzy controller

We will apply fuzzy PD+I control, since it is the only configuration of those previously mentioned that can be made equivalent to crisp PID control. The fuzzy PD sub-controller must be implemented as a Sugeno type controller, in order to achieve linearity. We shall follow the checklist given in the summary of the previous chapter.

Rule base–related choices The number of inputs and outputs are given by the PID implementation, and we shall use the same inputs. The rule base must provide for a linear control surface. The simplest rule base with two inputs consists of the four rules

(4.1). The premise universes are arbitrarily chosen to be the generic percentages of full scale, $[-100, 100]$. They are continuous, since we will apply continuous membership functions on the premise side. The input families are already hinted at in the rule base: on the premise side we shall use membership functions Neg and Pos, and on the conclusion side we shall use NB, Zero (control signal is zero), and PB. In order to achieve linearity and a rule base equivalent to a pure summation, Neg and Pos must be triangular and overlap by 50% stretching over the full length of the universe. Furthermore, the conclusion universe must be $[-200, 200]$, since there are two inputs, and the membership functions must be singletons at $NB = -200$, $PB = 200$.

Inference engine–related choices The two premise variables *error* and *change in error* must be combined with the connective 'and'. Furthermore 'and' must be implemented as multiplication in order to achieve linearity. For activation we apply multiplication and for accumulation we apply sum. Hedges are not relevant for this case.

Defuzzification method For defuzzification we apply centre of gravity for singletons (COGS). Since we are implementing a Sugeno type controller, the combined activation, accumulation, and defuzzification operation simplifies to weighted average, with the activation strengths weighting the singleton positions.

Pre- and post-processing For keeping the design general, we shall implement the α*-scaling* in Equation (4.25), but keep $\alpha = 1$ as long as we are in the linear domain. Quantization is not relevant, because we are employing continuous premise membership functions and it will not be necessary to implement a table-based controller. Regarding the choice of gain factors, we are guided by the PID settings. We choose $GE = 100$ since the error universe is $[-100, 100]$, and according to the plot in Figure 4.7, the maximal error is 1. By Equation (4.20) GU is now fixed by the relation

$$GU = K_p/GE = 4.8/100$$

The gain GCE is then determined by Equation (4.21),

$$GCE = GE * T_d = 100 * 15/32$$

The last gain is, by Equation (4.22),

$$GIE = GE * 1/T_i = 100 * 8/15$$

The sample time T_s is chosen at 0.05 seconds, since T_d is near 0.5 and one-tenth of that should be appropriate.

The step response with the linear FPD+I (Figure 4.11, left column) is exactly identical to that of the PID controller (Figure 4.7). The right-hand column of Figure 4.11 shows the phase trajectory of the control signal mapped onto the control surface. It shows that $|E_{\max}| \leq 100$ and $|CE_{\max}| \leq 55$ – thus there was no saturation in the universes.

The two final steps of the design procedure will be skipped here, because they concern the nonlinear aspects of fuzzy control.

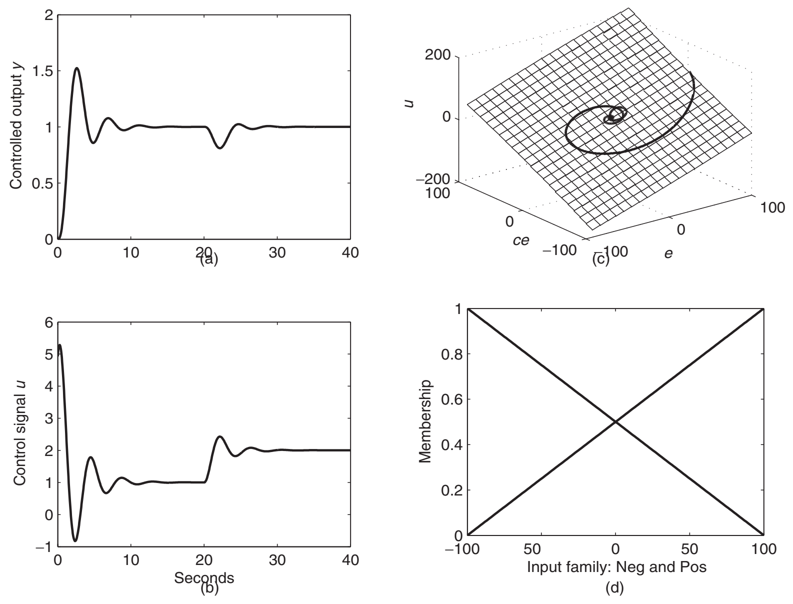


Figure 4.11: Fuzzy FPD+I control of the plant $1/(s+1)^3$. The left column shows the response (a) and the control signal (b). The right column shows the control surface is linear as well as the trajectory of the control signal (c). The bottom plot shows (d) the input family Neg and Pos. (figfpdi4.m)

4.8 Practical Considerations*

The previous sections assume the ideal controller given by Equation (4.2), but in practice the equation is modified. Differentiation in a digital controller, for example, can be implemented in a number of ways to avoid problems with noise. Modifications pertaining to PID control can also be transferred to fuzzy controllers.

Setpoint weighting

The ideal PID controller in Equation (4.3) is sensitive to abrupt changes in the reference. For example, a unit step in the reference causes the proportional action to jump by the amount K_p and the derivative action to jump to a large magnitude, when the sampling time is small. *Setpoint weighting* modifies the error signal used in the proportional action e_p and the derivative action e_d such that the effect of a sudden change in the reference signal will be attenuated. The modified controller based on e_p and e_d is (Åström and Hägglund 1995),

$$u(n) = K_p \left(e_p(n) + \frac{1}{T_i} \sum_{j=1}^{n} e(j) T_s + T_d \frac{e_d(n) - e_d(n-1)}{T_s} \right) \tag{4.26}$$

*Can be skipped in a first reading.

The error signal in the proportional action is

$$e_p(n) = b * Ref(n) - y(n) \tag{4.27}$$

and the error signal in the derivative action is

$$e_d(n) = c * Ref(n) - y(n) \tag{4.28}$$

The error signal in the integral action remains unmodified. For constant *Ref* the closed-loop response to load changes will be independent of the values of the parameters b and c. The response to changes in the reference signal, however, will depend on b and c. For $b = 0$ the proportional action reacts to changes in the controller output only, which generally reduces the overshoot. For $c = 0$ the derivative action reacts to changes in the controlled output only, thus completely avoiding differentiation of a discontinuous jump in the reference signal. With $b = c = 1$ we achieve the original configuration.

The configuration of the equivalent linear fuzzy controller is unaffected, as long as it is understood that e is replaced by e_p from Equation (4.27) and $\dot{e}$ by e_d from Equation (4.28).

Filtered derivative

In the presence of high frequency noise, the derivative action causes unwanted spikes in the control signal. A low-pass filter in combination with the pure derivative in the ideal controller in Equation (4.2) attenuates the spikes. The derivative action is modified to (Åström and Hägglund 1995),

$$u_d = -\frac{sK_pT_d}{1 + s\frac{T_d}{N}} y \tag{4.29}$$

where s is the Laplace operator for differentiation, and $\left(-sK_pT_dy\right)$ is the ideal D-action. It is multiplied by a first order transfer function $1/(1 + sT_d/N)$, which is a low-pass filter with the cut-off frequency N/T_d. High frequency noise ($s \to \infty$) is amplified at most by a factor K_pN, and in steady state ($s = 0$) the D-action vanishes. For $N \to \infty$ the expression in Equation (4.29) tends towards the ideal derivative action. Typical values in practice are $8 \le N \le 20$.

Replacing s with backward differences leads to a digital implementation. Note first that the factor K_pT_d in the numerator of Equation (4.29) is the gain on the derivative of the ideal differential, which is equivalent to $GCE * GU$ in the linear FPD controller. The remainder is

$$z = -\frac{s}{1 + s\frac{T_d}{N}} y$$

A discrete time approximation is

$$\begin{aligned}
z(n) + \frac{T_d}{N}\frac{z(n) - z(n-1)}{T_s} &= -\frac{y(n) - y(n-1)}{T_s} \Leftrightarrow \\
z(n)(T_s + \frac{T_d}{N}) &= \frac{T_d}{N} z(n-1) - (y(n) - y(n-1)) \Leftrightarrow \\
z(n) &= \frac{T_d}{NT_s + T_d} z(n-1) - \frac{y(n) - y(n-1)}{T_s + \frac{T_d}{N}}
\end{aligned}$$

Thus $z(n)$ is the signal to feed into the FPD controller in place of $\dot{e}(n)$, with T_d replaced by GCE/GE. It is a linear combination of the previous value $z(n-1)$ and the change in controlled output $y(n) - y(n-1)$. Again it is seen that for $N \to \infty$, $z(n)$ tends towards the unfiltered difference quotient $-(y(n) - y(n-1))/T_s$ arising from the combination of Equations (4.26) and (4.28).

Anti-windup

The integrator in the I-action can sum up to a large magnitude, much larger than called for; it *winds up*. Integrator windup occurs when the actuator after the controller along the signal path has limits, such as a maximum opening of a valve. The actuator remains at its limit corresponding to u_{lim}, while the integrator keeps integrating causing a control signal $u(n) > u_{lim}$. When the error changes sign, the integrator starts to wind down, but the actuator stays at its limit until $u(n)$ passes u_{lim} as the control signal returns to the operating range of the actuator. Windup can cause a large overshoot, or an oscillation, where the actuator bounces from one extreme to the other, the so-called *chattering*.

One remedy (Rundqwist 1991) is to apply *conditional integration*, where the integrator is switched off at a prescribed condition, for example,

- to stop integrating when the control error is large, or $|e| > e_0$;
- to stop integrating when the controller saturates, or $u(n) > u_{lim}$;
- to stop integrating as before *and* the control error has the same sign as the control signal, $\text{sgn}(u(n)) = \text{sgn}(e(n))$;
- to limit the integrator I, such that $|I(n)| \leq I_0$; or
- to stop integrating and assign a prescribed value to the integrator when a condition is true.

Another approach is to use feedback from the control signal and the saturation value, such that their difference drives the controller. Finally, a last approach is to limit the controller input such that the control signal never saturates. This will often lead to a conservative, sluggish behaviour of the closed-loop system.

4.9 Summary

We have achieved a tuning procedure for fuzzy controllers. Referring to the FPD+I controller – because it covers both proportional, integral, and derivative action – the procedure is as follows.

Procedure Fuzzy controller design, especially steps 1 and 2

1. *Design a crisp PID controller*. Use a PID tuning method to find K_p, $1/T_i$, and T_d.
2. *Replace it with a linear fuzzy*. Use the FPD+I configuration, transfer K_p, $1/T_i$, and T_d to *GE*, *GCE*, *GIE*, and *GU* (*GCU*) using Table 4.1. Run the controller, and check for saturation in the premise universes. When it is removed, by means of α-*scaling*,

check that the closed-loop response is exactly the same with the fuzzy rule base replaced by a pure summation.

3. *Make it nonlinear*. See next chapter.

4. *Fine-tune it*. See next chapter.

The FPD+I controller has one degree of freedom, since it has one more gain factor than the crisp PID. This is used to exploit the full range of one premise universe. If, for example, the reference step is 1 and the universe for E is $[-100, 100]$, then fix GE at 100 in order to exploit the full range. The free variable should be GE or GCE, whichever signal has the largest magnitude after multiplication by its gain factor. A scaling factor α on the gains addresses the problem elegantly.

The performance depends on the control surface. With a linear control surface the fuzzy FPD+I controller can be made to perform exactly like a crisp PID controller.

Perhaps a nonlinear control table will perform better than a linear one, but it depends on the plant. Since an FPD+I controller contains a crisp PID controller as a special case, the fuzzy controller performs at least as well as the PID. Starting with a crisp PID controller and gradually making it fuzzy is safer than starting from scratch – and linear control theory applies. Steps 3 and 4 of the design procedure concern nonlinear control, which will be treated in the following chapter.

4.10 Notes and References

The link between the fuzzy and the PID gains has been sought by various authors (e.g. Siler and Ying 1989, Mizumoto 1992, Qiao and Mizumoto 1996). With the specially crafted linear fuzzy control surface in this chapter, the relationship is particularly transparent, opening up for a transfer of PID tuning methods.

The authoritative reference on PID control by Åström and Hägglund (1995) presents implementation algorithms and tuning methods, and develops the so-called kappa-tau tuning method. The methods are illustrated by simulation examples, one of them being the third-order plant used in this chapter. Recent tuning methods include Ziegler–Nichols, kappa-tau, pole placement, stability margins, D-partitioning, polynomial, Nyquist, genetic algorithms, adaptive interaction, cancellation, K-B parametrization, multiple integration, and frequency loop shaping; see the overview article by Cominos and Munro (2002). The article is part of a special issue on PID control (Isaksson and Hägglund 2002). For shaping the frequency response, we would ideally like the Nyquist curve to be the vertical line passing through -0.5 on the real axis, as that would mean unity closed-loop gain at all frequencies and an infinitely fast response (Gyöngy and Clarke 2006).

Historically, Ziegler and Nichols were the first to publish optimum settings found from open-loop tests and closed-loop tests (1942, 1943). Cohen and Coon recognized, however, that alternatives were necessary for certain types of plant (1953). For a historical account of the events and the instrumentation leading up to that point, see the article by Bennett (1993) or the summary in Åström and Hägglund (1995, p 117).

5

Nonlinear Fuzzy PID Control

The designer must brace himself for the next step into nonlinear control. The analytical methods lack generality in the nonlinear domain, and it is difficult to predict the behaviour of a nonlinear system.

The chapter focuses on phase plane analysis, known from nonlinear differential equation theory. The outcome is a map of a vector field and a trajectory plot, which together shed light on the global behaviour of a closed-loop system. The plots provide valuable hints to the design of the fuzzy controller, even for systems of higher order than two.

It is characteristic for a linear time-invariant system of the form

$$\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x} + \boldsymbol{B}\mathbf{u} \tag{5.1}$$

where $\mathbf{x}$ is the state variable vector, $\boldsymbol{A}$ is the system matrix, and $\boldsymbol{B}$ the input matrix, that (1) the response has an analytical solution, and (2) a sinusoidal input causes a sinusoidal output of the same frequency. Nonlinear systems do not possess similar characteristics; on the contrary, it is characteristic that the superposition principle does *not* hold (superposition is briefly: $y_1 + y_2 = G(u_1 + u_2)$, when $y = G(u)$ is a transfer function from input u to output y). It is further characteristic that the output depends not only on the frequency of the input, but also on amplitude. Therefore we cannot easily use Nyquist plots anymore, or at least several plots must be drawn: one for each chosen amplitude.

The overall plan for the following sections is to approach the final two steps of the design procedure: (3) insert a nonlinear rule base, and (4) fine-tune using hand-tuning. Basic knowledge of the state-space approach is a prerequisite.

5.1 Phase Plane Analysis

A *phase plot* is generally a plot of a variable and its time derivative plotted against each other. The fuzzy PD (FPD) controller operates with error and its time derivative, hence our interest in the phase plots. In fact, the control surface is a plot of change in error against error, and can therefore be regarded as a phase plot.

Phase plane analysis is a graphical method providing valuable visual overview. It has a drawback, however: it is limited to two-dimensional plots. Systems of order higher than two are normally excluded from phase plane analysis, but the chapter suggests an engineering shortcut in order to get *some* information, even for higher-order systems.

We can apply phase plane analysis without having to solve the differential equations analytically. Furthermore, it accommodates discontinuous (hard) nonlinearities.

Phase trajectory

Our starting point is a second-order autonomous system

$$\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x} \tag{5.2}$$

or written out,

$$\dot{x}_1 = a_{11}x_1 + a_{12}x_2 \tag{5.3}$$

$$\dot{x}_2 = a_{21}x_1 + a_{22}x_2 \tag{5.4}$$

where x_1 and x_2 are the state variables of the system, and $\boldsymbol{A}$ the system matrix. An *autonomous* system is a system without inputs, and the usual input term is missing.

Geometrically, the variables x_1 and x_2 span a two-dimensional space, called the *phase plane*. The solution in time $\mathbf{x}(t)$ to Equation (5.2), given an initial value $\mathbf{x}(0)$ of the state vector, is a curve in the phase plane with x_2 along the y-axis and x_1 along the x-axis. The curve is called a *phase trajectory*.

Equilibrium point

If the system settles down to an equilibrium state, as the result of some initial state, the equilibrium is characterized by zero motion in the phase plane. Such an *equilibrium point* (*singular point*) must satisfy the equation

$$\mathbf{0} = \boldsymbol{A}\mathbf{x}$$

If the $\boldsymbol{A}$ matrix is nonsingular (determinant is not zero), then the origin $\mathbf{x} = \mathbf{0}$ is an equilibrium point. Otherwise, there may be a collection of equilibrium points lying on a line through the origin. Nonlinear systems can have several equilibrium points, one, or none at all.

The ratio of Equation (5.4) to Equation (5.3) is the slope of the phase trajectory, since we have

$$\frac{\dot{x}_2}{\dot{x}_1} = \frac{\frac{dx_2}{dt}}{\frac{dx_1}{dt}} = \frac{dx_2}{dx_1}$$

The slope S at a given point (x_1, x_2) is therefore

$$S = \frac{dx_2}{dx_1} = \frac{a_{21}x_1 + a_{22}x_2}{a_{11}x_1 + a_{12}x_2} \tag{5.5}$$

Conversely, given a particular slope $S = S^*$, and solving the equation for x_2, results in an equation for a line along which the slope of all trajectories crossing it is the same. Such a curve is called an *isocline*, and a collection of isoclines can be used to graphically construct phase plane trajectories (see Slotine and Li 1991 for example).

Drawing isoclines is the classical route to take, but we shall deviate slightly and apply a view from differential geometry instead. Geometrically, the vector differential Equation (5.2) expresses the rate of change $\dot{\mathbf{x}}(t)$ at the time t of the state $\mathbf{x}(t)$ of the system. As $\mathbf{x}(t)$ is the locus vector of a point in the phase plane travelling along a trajectory, $\dot{\mathbf{x}}(t)$ is its *velocity vector*. The velocity vector is tangential to the phase trajectory. Physically, the velocity vector indicates the speed (its length) and the direction (its angle) along which the motion takes place.

The matrix $\boldsymbol{A}$ can be viewed as an operator that maps $\mathbf{x}$ to $\dot{\mathbf{x}}$. For almost all vectors in the plane, the operation is a composition of rotation and multiplication, such that $\dot{\mathbf{x}}$ is at an angle with $\mathbf{x}$ and it has a different length. For an eigenvector $\mathbf{v}$, however, $\boldsymbol{A}\mathbf{v} = \lambda\mathbf{v}$ where λ is the corresponding eigenvalue; thus the mapping is just a multiplication, the rotation is absent.

The geometrical view allows us to visually inspect the phase plane as a vector field.

Stability

Stability requires that the angle φ between the locus vector $\mathbf{x}$ and the velocity vector $\dot{\mathbf{x}}$ is larger than 90 degrees, such that the velocity has a component which points towards the origin (with the other component being orthogonal to the locus vector). Mathematically, this can be expressed as

$$\frac{\mathbf{x}^T\dot{\mathbf{x}}}{|\mathbf{x}|\,|\dot{\mathbf{x}}|} = \cos\varphi < 0 \tag{5.6}$$

Since $\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x}$, stability requires $\mathbf{x}^T\boldsymbol{A}\mathbf{x} < 0$, which is equivalent to the usual criterion that all the eigenvalues of $\boldsymbol{A}$ must have negative real parts.

Equations (5.3 and 5.4) enable us, for any particular instance $\left(x_1^*, x_2^*\right)$ of the state vector, to find the direction of movement $\left(\dot{x}_1^*, \dot{x}_2^*\right)$. Taking a number of discrete points $\left(x_1^*, x_2^*\right)$, distributed in the phase plane in an even manner, and plotting the velocity vector at each point, the designer will acquire an overview of the behaviour of the system in the region under consideration. Figure 5.1 shows six different types of behaviour around the equilibrium point of various $\boldsymbol{A}$ matrices:

Node If the eigenvalues of the $\boldsymbol{A}$ matrix are real and negative, the equilibrium is a *stable node*, because $\mathbf{x}(t)$ converges to $\mathbf{0}$ in an exponential decay. If both eigenvalues are positive, the equilibrium is an *unstable node*, because $\mathbf{x}(t)$ diverges exponentially. Since the eigenvalues are real, there are no oscillations in the motion; compare the plots of $x_1\,(t)$ in the time domain (Figure 5.1, first row).

Focus If the eigenvalues of the $\boldsymbol{A}$ matrix are complex conjugates, with negative real parts, the equilibrium is a *stable focus*. The state vector $\mathbf{x}(t)$ converges to $\mathbf{0}$, but in an oscillatory manner, whereby the trajectory in the phase plane encircles the origin one or more times, unlike the stable node. If both eigenvalues have positive real parts, the equilibrium is an *unstable focus*, because $\mathbf{x}(t)$ diverges in an oscillatory manner; compare the plots of $x_1\,(t)$ in the time domain (Figure 5.1, second row).

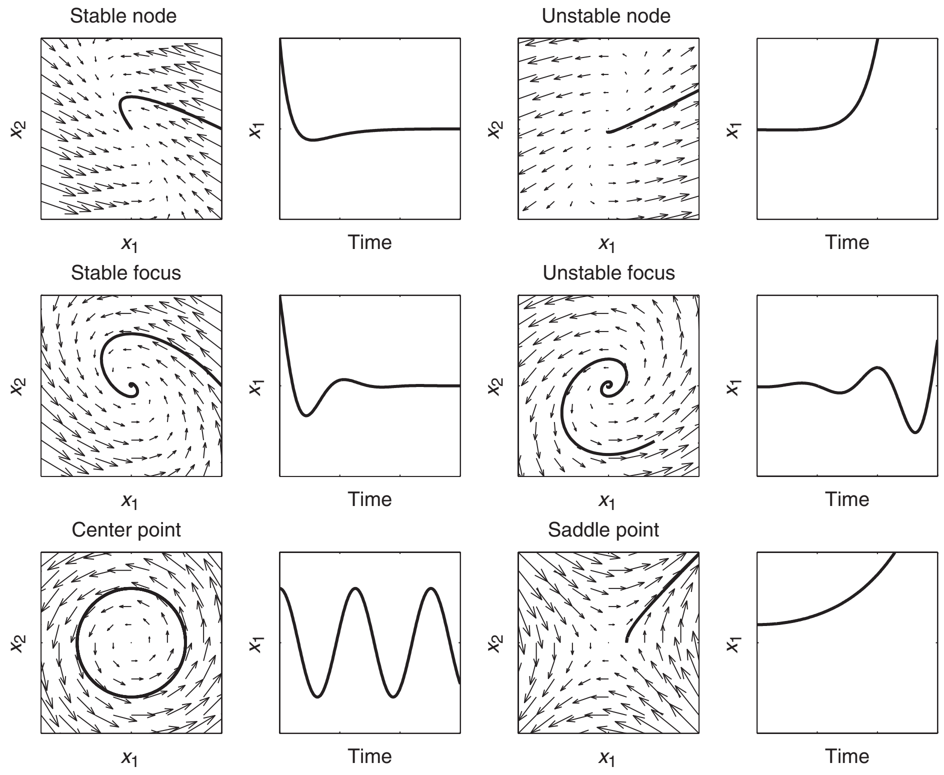


Figure 5.1: Equilibrium points. Read columnwise, columns 1 and 2 are stable or marginally stable, while columns 3 and 4 are unstable. The trajectory in an (x_1, x_2)-plane is equivalent to the time response on its right resulting from an individual initial state. (figequil.m)

Centre point If the eigenvalues of the $\boldsymbol{A}$ matrix are complex conjugates, with real parts equal to zero, the equilibrium is a *centre point (vortex)*, because $\mathbf{x}(t)$ encircles the equilibrium point, without converging or diverging. The plot of $x_1\,(t)$ in the time domain shows a sustained oscillation, a marginally stable system (Figure 5.1, third row, columns 1 and 2).

Saddle point If the eigenvalues of the $\boldsymbol{A}$ matrix are real, but one is negative and the other is positive, the equilibrium is a *saddle point*. Because of the unstable positive eigenvalue, almost all trajectories will diverge to infinity; see the plot of $x_1\,(t)$ in the time domain (Figure 5.1, third row, columns 3 and 4).

The eigenvalues completely determine the type of the equilibrium point, and the plots provide a visual clue to the behaviour around the equilibrium point.

Example 5.1.1 *Double integrator*

Consider a plant that can be modelled as a double integrator with the transfer function

$$\frac{y}{u} = \frac{1}{s^2} \Leftrightarrow \ddot{y} = u$$

Here u is the input, y is the output, and s is the Laplace operator for differentiation. It is a short step to arrive at a state-space model of the system. By the choice of state variables $x_1 = y$, and $x_2 = \dot{y}$, we rewrite into a set of equations

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= u\end{aligned} \tag{5.7}$$

Comparing with the state-space form, Equation (5.1), the matrices are,

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

By setting $\dot{x}_1 = \dot{x}_2 = 0$ in Equation (5.7), and $u = 0$ as well, because we are considering the autonomous system, we find that in equilibrium $x_2 = 0$, while x_1 can be anything. Thus the equilibrium is a line, the x_1-axis. Intuitively, any point on the x_1-axis has zero velocity and will stay there. Any point above or below the x_1-axis has a constant velocity that will move the point towards infinity.

The eigenvalues of the $\boldsymbol{A}$ matrix ($\lambda = 0$) are real, indicating an unstable node.

Even though the method is restricted to autonomous systems, it can be applied to closed-loop systems, especially if the reference is constant. If the local behaviour of a nonlinear system can be approximated by that of a linear system, the equilibrium types are the same.

Closed-loop system

Consider the system in Figure 5.2. Assume for now that both the plant and the controller are linear. Furthermore assume that the plant is of second order, the reference *Ref* is constant,

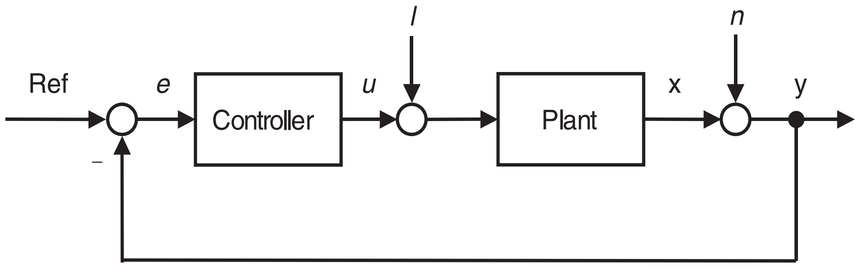


Figure 5.2: Feedback loop with load l and noise n.

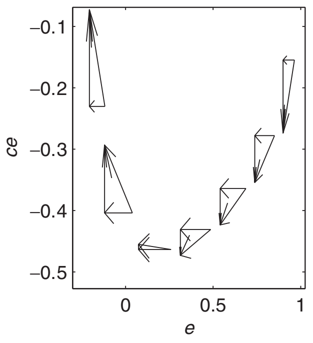


Figure 5.3: Velocity vectors along a trajectory. The velocity vector at each point is the sum of two vectors: the open-loop velocity plus the controller action. Compare Equation (5.10). (figs2load.m)

the plant has only a single input u, and there is no load or noise ($l = 0, n = 0$). The plant is then characterized by the state-space equations

$$\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x} + \mathbf{b}u \tag{5.8}$$
$$\mathbf{y} = \boldsymbol{C}\mathbf{x} + \mathbf{d}u$$

The controller delivers a control signal u, related to the error e by a transfer function G_c,

$$u = G_c e \tag{5.9}$$

Substituting Equation (5.9) into Equation (5.8), the closed-loop system equations appear,

$$\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x} + \mathbf{b}G_c e \tag{5.10}$$

Thus the velocity vector $\dot{\mathbf{x}}$ is a sum of two vector components: the open-loop velocity vector $\boldsymbol{A}\mathbf{x}$ plus a contribution from the controller $\mathbf{b}G_c e$. A plot of the two components shows the influence of the controller at various stages of the closed-loop response and thus provides a visual tuning aid (Figure 5.3).

Example 5.1.2 *Double integrator, trying proportional control*

For the double integrator in the previous example, we will investigate whether a proportional controller can stabilize the system. Assume therefore a P-controller $u = K_p e$, where K_p is the proportional gain and e the error,

$$e = Ref - x_1 \tag{5.11}$$

Inserting into Equation (5.8) yields the system equations

$$\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x} + \mathbf{b}u = \boldsymbol{A}\mathbf{x} + \mathbf{b}K_p e \tag{5.12}$$

with the same $\boldsymbol{A}$ matrix and $\mathbf{b}$ vector as in the open-loop system. Writing the equations out, the closed-loop system is,

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = K_p(Ref - x_1) = -K_p x_1 + K_p Ref$$

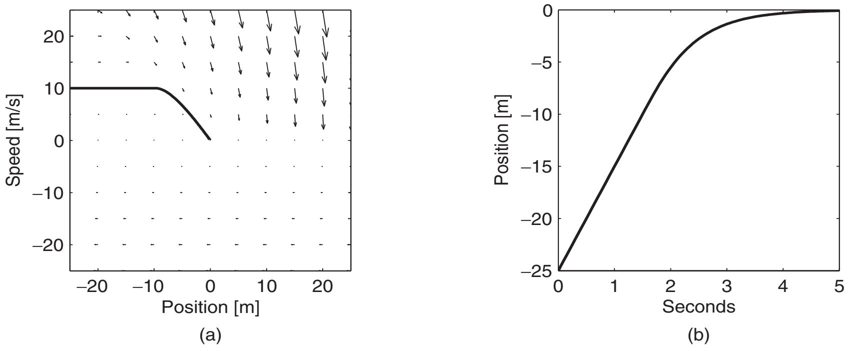


Figure 5.4: Stopping a car. The controller warps the vector field (a) such that the trajectory ends in the centre. Braking starts at a distance of 10 metres (b). (figcar.m)

The closed-loop system matrix and input vector are thus

$$\boldsymbol{A}_{cl} = \begin{bmatrix} 0 & 1 \\ -K_p & 0 \end{bmatrix}, \mathbf{b}_{cl} = \begin{bmatrix} 0 \\ K_p \end{bmatrix} \tag{5.13}$$

The eigenvalues of $\boldsymbol{A}_{cl}$ $\left(\lambda = \sqrt{-K_p}\right)$ *are either complex conjugates on the imaginary axis* $(K_p > 0)$*, indicating a marginally stable centre point in equilibrium, or the eigenvalues are real with one negative eigenvalue and one positive* $(K_p < 0)$*, indicating an unstable saddle point. We can conclude it is impossible to stabilize the system with proportional control.*

This explains the very sensitive solution to the problem of stopping a car using just a proportional controller (see Chapter 1).

Example 5.1.3 *Stopping a car, PD control*

Consider again the problem of stopping a car in front of a red stop light (see Chapter 1). We saw that a PD controller resulted in a good response; let us take a look at the motion in the phase plane.

Figure 5.4 shows that the controller distorts the vector field, especially in the upper half of the phase plane, while the rest of the plane remains unchanged because of the constraints on the control signal.

The example shows first of all that the phase plane provides valuable overview. Secondly, the phase plane method could also accommodate the nonlinear constraints on the control signal.

5.2 Fuzzy PD Control

The FPD controller lends itself to phase plane analysis. Its configuration is repeated in Figure 5.5; it has only two inputs, which makes it suitable for two-dimensional plotting. Furthermore, the FPD is important, because it is a component in two other controller configurations: FPD+I and FInc.

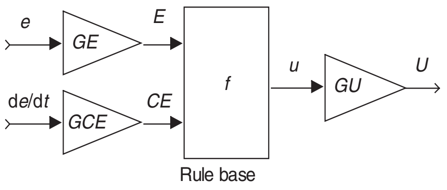


Figure 5.5: Fuzzy PD controller, FPD.

Table 5.1: Relationships between linear fuzzy and PID gains.

Controller	K_p	$1/T_i$	T_d
FP	$GE * GU$		
FInc	$GCE * GCU$	GE/GCE	
FPD	$GE * GU$		GCE/GE
FPD+I	$GE * GU$	GIE/GE	GCE/GE

Again, the starting point is the linear state-space model in Equation (5.8). Only this time, the controller is nonlinear in general, and the control signal U is the result of a nonlinear function F of the error and the change in error,

$$U = F(E, CE) * GU \tag{5.14}$$

Recall that there is a relationship between the PID tuning parameters and the fuzzy gains (Table 5.1, a copy of a previous table).

Substituting Equation (5.14) into Equation (5.8) the closed-loop system equations are,

$$\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x} + \mathbf{b}F(E, CE) * GU \tag{5.15}$$

We shall investigate how the controller component $\mathbf{b}F(E, CE) * GU$ affects the vector field of the phase plane.

Phase plane

With two inputs and one output the rule base mapping is the relationship between error and change in error on the premise side, and control action on the conclusion side, as portrayed in the control surface. To be precise, the control surface concerns the signals E and CE after the gains GE and GCE, and the rule base output u before the gain GU (Figure 5.5). Variables E and CE span a plane, which we regard as a phase plane between e and $\dot{e}$ having axes scaled by GE and GCE respectively. After a dynamic response, the sequence of errors and change in errors can be plotted, $CE(t)$ against $E(t)$, to form a phase trajectory. The (E, CE)-plane is bounded by the universes, and the trajectory will always stay within the boundary. In a table-based controller, the trajectory points to the cells in the control table visited by the controller.

To summarize, we have three different variants of a phase plane: (1) the original (x_1, x_2)-plane between two state variables, (2) the $(e, \dot{e})$-plane between error and its derivative, and (3) the (E, CE)-plane corresponding to a control table. All three are equivalent, in the sense that we can transform one into the other by means of a linear transformation based on the relationships: $e = Ref - x_1$, $E = GE * e$, and $CE = GCE * \dot{e}$.

Figure 5.6 shows a typical step response with overshoot. It has four consecutive stages marked by circles in the plots:

Stage 1: $E > 0, CE < 0$ Initially the error is large and positive, and the plant output is moving towards the reference. The error $E = GE * e = GE * (Ref - y)$ is positive (for $GE > 0$) as long as the plant output is below the reference. Furthermore, the change in error is negative (for $GCE > 0$), since $\dot{e} = -\dot{y}$, as long as the plant output is increasing. The situation corresponds to the fourth quadrant of the phase plane. The phase trajectory spirals in a clockwise direction.

Stage 2: $E < 0, CE < 0$ The plant output has overshot the reference and is still moving away from the reference. The error is negative, since the plant output is above the reference. Furthermore, the change in error is negative, since the plant output is still increasing. The situation corresponds to the third quadrant of the phase plane.

Stage 3: $E < 0, CE > 0$ The plant output is returning towards the reference. The error is negative, since the plant output is above the reference. Furthermore, the change in error is positive, since the plant output is now decreasing. The situation corresponds to the second quadrant of the phase plane.

Stage 4: $E > 0, CE > 0$ The plant output is moving away from the reference during an undershoot. The error is positive, and the plant output is below the reference. Furthermore, the change in error is positive, and the plant output is decreasing. The situation corresponds to the first quadrant of the phase plane.

Each stage corresponds to a quadrant in the phase plane, and the trajectory of the response can be affected or shaped to an extent by local rules in each quadrant. Take for example the nine rules fuzzy PD controller,

1. If *error* is Neg and *change in error* is Neg then *control* is NB
2. If *error* is Neg and *change in error* is Zero then *control* is NM
3. If *error* is Neg and *change in error* is Pos then *control* is Zero
4. If *error* is Zero and *change in error* is Neg then *control* is NM
5. If *error* is Zero and *change in error* is Zero then *control* is Zero (5.16)
6. If *error* is Zero and *change in error* is Pos then *control* is PM
7. If *error* is Pos and *change in error* is Neg then *control* is Zero
8. If *error* is Pos and *change in error* is Zero then *control* is PM
9. If *error* is Pos and *change in error* is Pos then *control* is PB

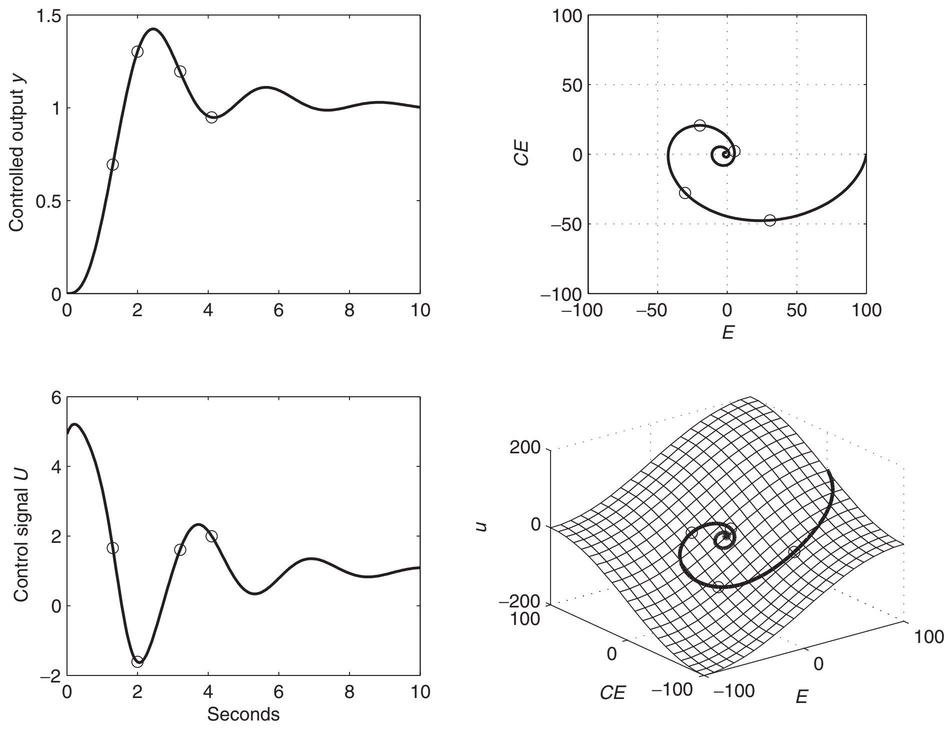


Figure 5.6: Phase plane regions. The four circles indicate four typical stages of a step response. (figregio.m)

Rule 1 concerns a negative error and a negative change in error, and this is a rule for stage 2 above. A designer can adjust the value of negative big (NB) in order to adjust the control (compare Figure 5.4) in the region of influence, which is roughly the third quadrant of the phase plane. Similarly for the other rules: rule 3 concerns stage 3 and quadrant 2; rule 7 concerns stage 1 and quadrant 4; rule 9 concerns stage 4 and quadrant 1; and rule 5 concerns the centre. The remaining rules concern the boundary zones between quadrants.

In Figure 5.6 the phase trajectory (upper right) is the projection of the trajectory on the control surface (lower right) onto the phase plane. By comparison, the diagonal in the northwest−southeast direction of the phase plane corresponds to zero control signal, and points above the diagonal correspond to positive control signals, while those below the diagonal correspond to negative control signals. The diagonal is thus a *switching line,* that is, the control signal changes sign when the trajectory crosses the line.

If the trajectory ends in the centre (origin), the plant is on the reference and not moving; that is an equilibrium. But a plant does not necessarily settle in the centre: a steady-state error results in an endpoint on the $CE = 0$ axis, but off the centre.

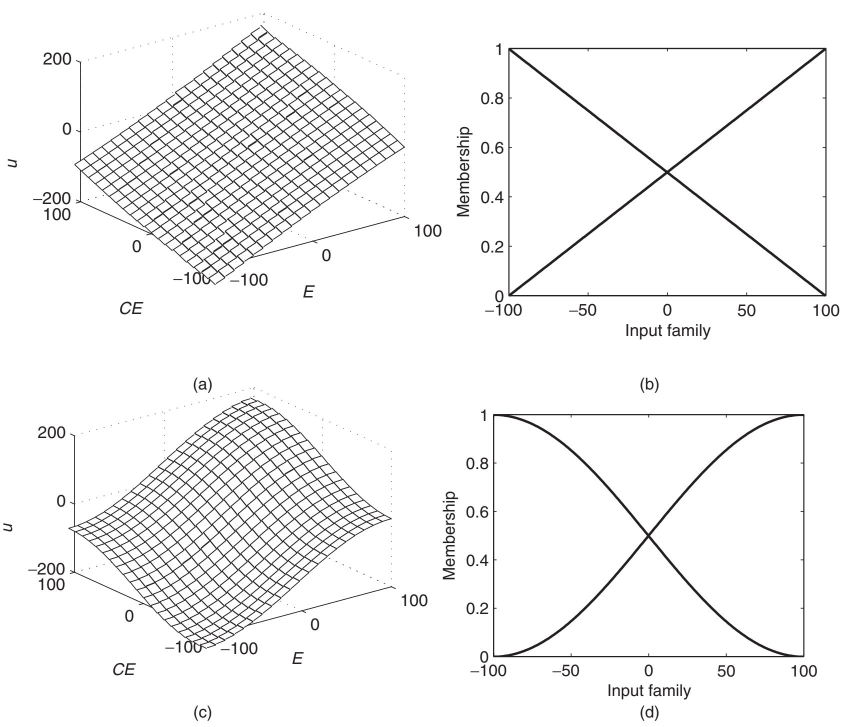


Figure 5.7: Linear surface (a) and saturation surface (c) with the input families that generated them (b and d respectively). (figsurfs.m)

Surface shaping

If an input family consists of the terms Pos, Zero, and Neg, then the complete combination of the two inputs consists of $3 \times 3 = 9$ rules. The shape of the sets affect the dynamics of the closed-loop system.

Figures 5.7 and 5.8 are examples of four surfaces. The three nonlinear surfaces are soft versions of the common nonlinearities saturation, deadzone, and quantizer, only in two dimensions:

Linear A *linear* surface (Figure 5.7, a and b) results from the rule base (5.16) using only rules 1, 3, 7, and 9, with triangular premise sets (Chapter 2) shown in the figure,

$$\mu_{Neg} = \mu_{Trapezoid}(x;\, -100, -100, -100, 100)$$

$$\mu_{Pos} = \mu_{Trapezoid}(x;\, -100, 100, 100, 100)$$

The surface in the figure is equivalent to the summation $E + CE$; compare the values on the axes. Since the surface is equivalent to a summation, the controller is similar to a crisp PD controller.

Saturation The *saturation* surface (Figure 5.7, c and d) is built using only rules 1, 3, 7, and 9, together with the premise sets shown in the figure,

$$\mu_{Neg} = \mu_{STrapezoid}(x; -100, -100, -100, 100)$$

$$`\mu_{Pos} = \mu_{STrapezoid}(x; -100, 100, 100, 100)$$

They are smooth trapezoids, built from segments of cosine functions (Chapter 2). Notice the absence of the central rule (number 5) with zero *error* and zero *change in error*. When the error is near zero, an increase will increase the control signal; but when the error reaches a certain level, a further increase causes little or no increase of the control signal. The same can be said for the change in error. That surface has a steeper slope – higher gain – near the centre of the table than the linear surface has. It has the same values pairwise in the four corners as the linear surface.

Deadzone The *deadzone* surface (Figure 5.8, a and b) is built from all nine rules. The figure shows the premise sets, defined as

$$\mu_{Neg} = 2\mu_{STrapezoid}(x; -200, -200, -200, 0)$$

$$\mu_{Zero} = 2\mu_{STrapezoid}(x; -200, 0, 0, 200)$$

$$\mu_{Pos} = 2\mu_{STrapezoid}(x; 0, 200, 200, 200)$$

They are smooth trapezoids, built from segments of cosine functions with half the frequency and twice the amplitude of the previous. That surface has a more gentle slope – lower gain – near the centre of the table. When the error is near zero, it affects the control signal little until the error exceeds a certain value. It also has the same values pairwise in the four corners as the linear surface.

Quantizer The *quantizer* surface (Figure 5.8, c and d) is a blend of the previous two surfaces. It is built using all nine rules, with nonlinear membership functions, defined as,

$$\mu_{Neg} = \mu_{STrapezoid}(x; -100, -100, -100, 0)$$

$$\mu_{Zero} = \mu_{STrapezoid}(x; -100, 0, 0, 100)$$

$$\mu_{Pos} = \mu_{STrapezoid}(x; 0, 100, 100, 100)$$

They are smooth trapezoids, built from segments of cosine functions. It has a flat plateau near the centre and other plateaus in several places. Even this surface has the same values as the other surfaces in the four corners.

The next example applies a nonlinear surface to the double integrator treated earlier.

Example 5.2.1 *Double integrator, FPD control*

Previous examples showed that the double integrator $1/s^2$ is open loop unstable, but possible to stabilize with PD control. Let us try with fuzzy PD control.

The open-loop system is Equation (5.7) and the closed-loop system will have a state-space model of the form of Equation (5.15). Figure 5.9 shows two simulations with crisp and fuzzy PD controllers respectively. The sampling time is $T_s = 0.05$ seconds everywhere. We followed the standard design procedure:

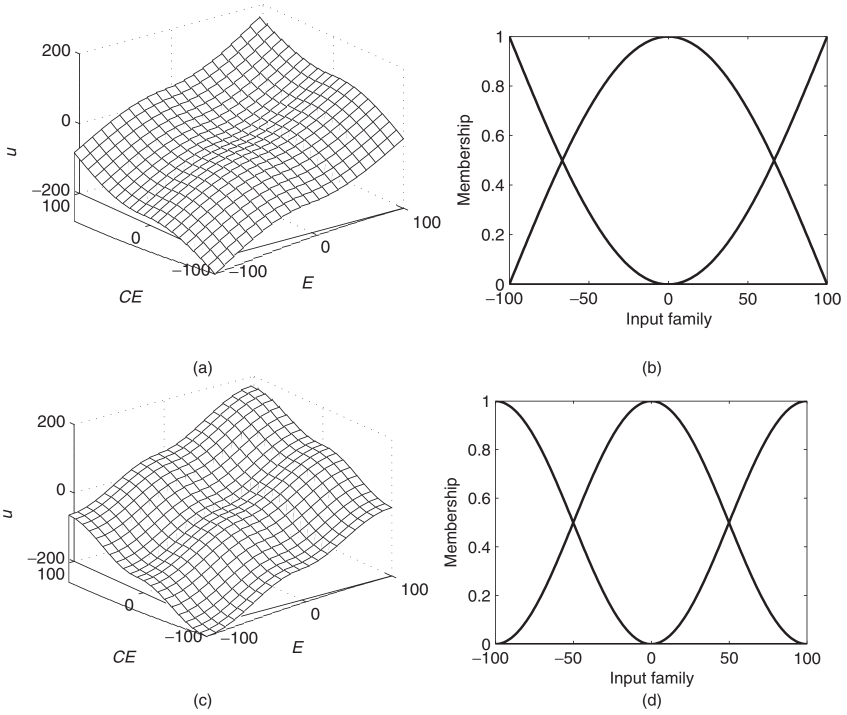


Figure 5.8: Deadzone surface (a) and quantizer surface (c) with the input families that generated them (b and d respectively). (figsurfs.m)

Step 1 *Crisp PD control. The closed-loop system equations are*

$$\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x} + \mathbf{b}u = \boldsymbol{A}\mathbf{x} + \mathbf{b}K_p(e + T_d\dot{e})$$

Hand-tuning resulted in the controller parameters

$$K_p = 0.5, T_d = 1$$

The response to a unit step in the reference is shown in Figure 5.9 (row 2, column 1). It is a loose setting, with 31% overshoot and some oscillation. The phase plane (row 1, column 1) shows that the controller has turned the vector field into a stable focus.

Step 2 *Replace with a linear FPD. The error is less than or equal to one, and to accommodate the whole range, the gain on the error is chosen to be* $GE = 100$*. The remaining gains are given by the gain relationships for a linear controller (Table 5.1),*

$$GU = K_p/GE = 0.5/100 = 0.005$$
$$GCE = GE * T_d = 100 * 1 = 100$$

With a linear control surface (Figure 5.7) the response was identical to that of the crisp PD controller.

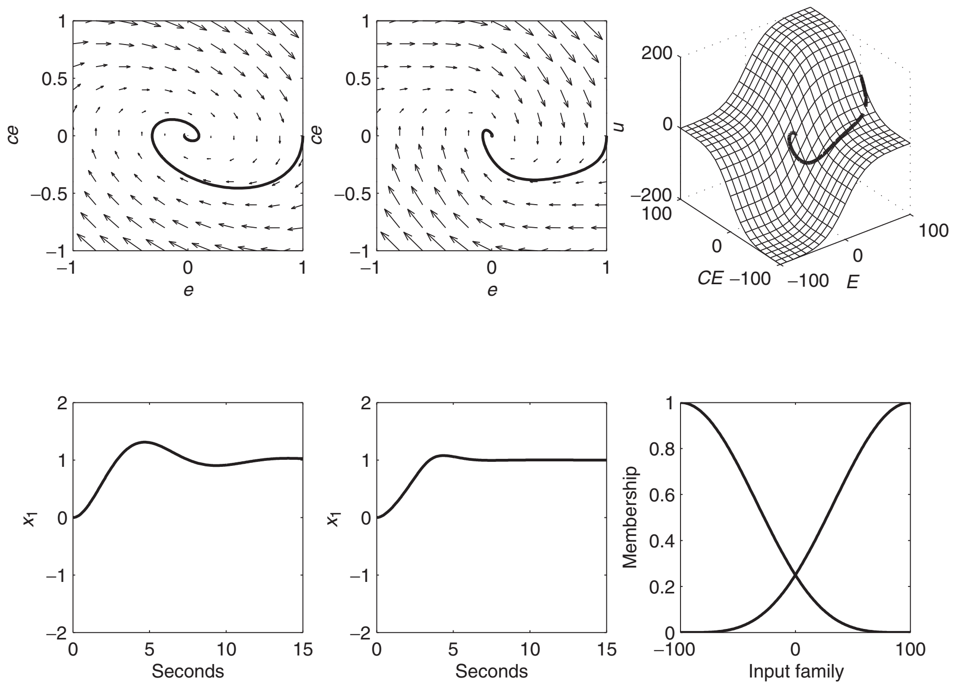


Figure 5.9: Step response of the plant $1/s^2$ with FPD control. Read columnwise, column 1 is crisp PD control, column 2 is FPD control, and column 3 its control surface with the generating membership functions underneath. The standard membership functions are squared. (figs2.m)

Step 3 *Make the FPD nonlinear. Inserting a saturation type surface diminished the overshoot, indicating that this was a good choice. In fact an even more pronounced shape was produced by squaring the membership functions, corresponding to the rule base*

If error is very Neg and change in error is very Neg then control is NB

If error is very Neg and change in error is very Pos then control is Zero

If error is very Pos and change in error is very Neg then control is Zero

If error is very Pos and change in error is very Pos then control is PB

The control surface is shown (row 1, column 3) together with the squared membership functions (row 2, column 3). The response in column 2 shows an overshoot of 8% and well dampened settling.

The fuzzy controller phase plot shows that the controller turns the velocity vectors slightly more inwards towards the centre, and the trajectory comes closer to the centre, compared to the linear controller.

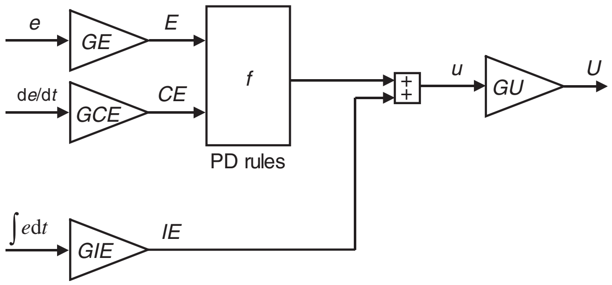


Figure 5.10: Fuzzy PID controller, FPD+I.

The example shows that the fuzzy controller can warp the vector field in a favourable direction. Two important questions arise: (1) is that *always* possible? and (2) what should a designer do to achieve that? These questions are still open.

5.3 Fuzzy PD+I Control

We would like to apply a similar analysis to FPD+I control in order to accommodate integral action. Figure 5.10, a copy of a previous figure, shows its configuration.

But the integrator creates problems by increasing the order of the closed loop system, which calls for three-dimensional plots, or multi-dimensional ones in the general case. That will be cumbersome. From an engineer's point of view, it may suffice to give up on completeness and make do with two-dimensional plots.

Again, the starting point is the linear state-space model, Equation (5.8). The controller is, in this case, a function of the integral error *IE*,

$$U = (F(E, CE) + IE) * GU \tag{5.17}$$

Insertion into Equation (5.8) the closed-loop system equations are,

$$\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x} + \mathbf{b}\,(F(E, CE) + IE) * GU \tag{5.18}$$

We can still in principle draw the vector field in the $(e, \dot{e})$-plane, but it depends on the integral action *IE* as well. Two strategies are possible, if the designer is willing to settle for a compromise:

Strategy 1 *Plot vector field.* Set $IE = k$ in Equation (5.18), where k is an arbitrary constant – such as 0, an estimate of the mean value of $IE(t)$ over time t, or an estimate of the final value of $IE(t)$. The plot of the two-dimensional vector field will thus be an approximation. However, for a particular time response, the phase trajectory and the value of $IE(t)$ is known at all sampling instances, and the committed error can be calculated and assessed. Geometrically, the plot is a section of a three-dimensional vector field parallel to the $(e, \dot{e})$-plane. The strategy entails a loss of accuracy, but it does provide an overview.

Strategy 2 *Plot trajectory components*. Refrain from drawing the whole vector field and instead, for each step of the time response, draw the component vectors (compare Figure 5.3). The plot will show the effect of the control vector locally along the trajectory. The strategy entails a loss of overview, but it does provide the full accuracy locally.

The next example applies both strategies to the same problem, in order to illustrate the strengths and weaknesses of either strategy.

Example 5.3.1 *Double integrator, FPD+I control*

As in the previous example, we are concerned with the double integrator $1/s^2$, only this time we will add a load $l = 0.5$ after 20 seconds, such that integral action becomes mandatory in order to remove the steady-state error. Let us try with fuzzy PD+I control.

For convenience, use the same PD parameters as in the previous example, and add integral action by hand-tuning. The final parameters are

$$Kp = 0.5, T_d = 1, T_i = 10$$

The equivalent fuzzy gain settings are (Table 5.1),

$$GE = 100, GCE = 100, GIE = 10, GU = 0.005$$

Now explore the two strategies outlined above.

Strategy 1 *Plot vector field. Figure 5.11 shows the response with the crisp PID controller (column 1) and the fuzzy PD+I controller (column 2) in the phase plane (upper) and the time domain (lower). The PID settings are somewhat loose, as in the previous example, and the crisp PID response is somewhat oscillatory. The phase plot shows two larger loops: one arising from the initial change of reference, the other arising from the step on the load at $t = 20$. The fuzzy controller shows an improved response with smaller overshoot and a smaller dip when the load kicks in. The phase plots show the relationship between e and $\dot{e}$, and the integral error is ignored by choosing $k = 0$ in the above description of strategy 1. The plots are therefore approximations to the actual vector field. In fact, the phase trajectories cross the direction of some arrows instead of following their direction, especially just after the load kicks in.*

Strategy 2 *Plot trajectory components. Figure 5.12 shows the trajectory in the $(e, \dot{e})$-plane. At every tenth sample point it shows how the true velocity is composed of the sum of the open-loop velocity and the controller component, which includes the integrator. All open loop velocities are horizontal, which is characteristic of the double integrator, because $\dot{e}$ is constant while e diverges to $\pm\infty$ with time, as a result of integrating a zero input signal. The controller component is always vertical, pointing either up or down, because the input affects acceleration only; compare the vector* **b** *in Equation (5.13). At the early stage of the response $(e \approx 1, \dot{e} \approx 0)$ the controller component points down while the open-loop component points left; it seems some improvement could be achieved by suppressing the control signal here, because the open-loop system by itself will move the locus point closer to the origin. At the initial stage of the load response, a similar situation occurs, with reverse directions.*

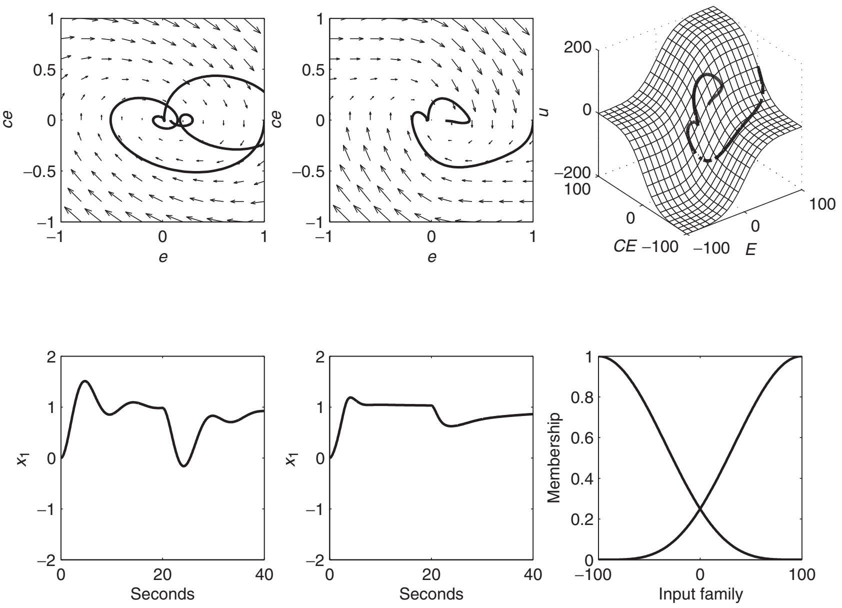


Figure 5.11: Step and load response of the plant $1/s^2$ with FPD+I control. (figs2load.m)

The vector field provides a global overview within the plane, but is not accurate in the detail. The trajectory components, however, are accurate, since the integral action is known precisely at each point for that particular response. They are local views, and therefore do not show the whole picture.

As an observation, the trajectory in the vector field ends in an equilibrium in the centre, but the trajectory is not always directed inwards. A sufficient *condition for stability is that the trajectory always approaches the equilibrium point, but the observation shows it is not a* necessary *condition. Both the crisp and the fuzzy responses pass through areas where the vector field temporarily points away from the equilibrium, or in other words, the angle between the locus vector and the velocity vector in the $(e, \dot{e})$-plane is sometimes less than* 90 *degrees – compare Equation (5.6) – nevertheless, the system is stable.*

The example shows there is a trade-off between accuracy and overview. The two types of plot provide one or the other as a hint to the designer on how to shape the control signal. In crisp PID control, changes to the parameters affect the response globally, whereas in fuzzy control changes to the rules affect the response locally.

5.4 Fine-tuning

The last step in the design procedure is to fine-tune the gains, now that the fuzzy controller is nonlinear. The final choice of gains is based on intuition and experience. However, a few rules of thumb can be derived from the linear approximations of PID control (Table 5.1):

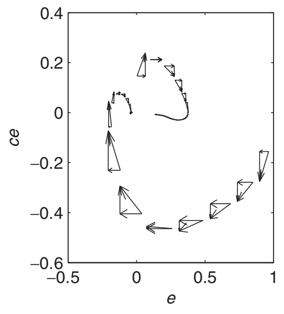


Figure 5.12: Phase plane trajectory. It is the same trajectory as in Figure 5.11, upper row, centre. (figs2load.m)

GE If the controller is to exploit the whole range of its universe, then the maximal $E = GE * e$ should equal the limit of the universe, that is

$$|e_{\max} * GE| = |Universe_{\max}|$$

With a unit reference step and the generic universe $[-100, 100]$ % of full range, the equation implies $GE = 100$. If GE is too high, all other settings being equal, the incremental controller will become less stable, because the integral gain is too high. In an FPD controller, GE affects the proportional gain and the derivative gain. One would like to have GE as large as possible to dampen noise and still have a fast response. In the FPD+I controller, one would also prefer GE as large as possible to promote the proportional gain at the expense of the integral gain and the derivative gain.

GCE Similarly if *change in error* is to exploit the whole range of its universe,

$$|\dot{e}_{\max} * GCE| = |Universe_{\max}|$$

In an FPD controller, a larger GCE means a larger derivative gain with no side effect on the proportional gain. To dampen noise to a minimum, one will therefore prefer GCE as small as possible. In the FInc controller an increase in GCE will decrease the integral gain and increase the proportional gain; thus one would like to keep GCE as large as possible to preserve stability. In the FPD+I controller, an increase in GCE will increase the derivative gain, so one would keep it as small as possible.

GCU/GU These affect the proportional gain, therefore one would like to have them as large as possible without creating too much overshoot. If too small, the system will be too slow, and if too large the system might become unstable.

A procedure for hand-tuning an FPD+I controller is the following (with trivial modifications this procedure covers the FPD and FInc controllers as well).

Procedure Hand-tuning FPD+I

1. Adjust *GE* (or *GCE*) according to step size and universe to exploit the range of the universe of *E* (or *CE*) fully.
2. Remove integral action and derivative action by setting $GIE = GCE = 0$. Tune *GU* to give the desired response, ignoring any final value offset.
3. Increase the proportional gain by means of *GU*, and adjust the derivative gain by means of *GCE* to dampen the overshoot.
4. Adjust the integral gain by means of *GIE* to remove any final value offset.
5. Repeat the whole procedure until *GU* is as large as possible.

Analytically, the stability will be similar to the stability of the system's linear approximation resulting from a Taylor series expansion near an equilibrium. In simulation, it is possible to experiment with different controller surfaces to find the stability margin and the sensitivity to dead times. The responses are amplitude dependent, however, and thereby dependent on the step size.

5.5 Higher-Order Systems

A question arises whether the method can be applied to systems of order higher than two, with some modification.

Consider a plant that can be modelled as a third-order system with the transfer function

$$\frac{y}{u} = \frac{1}{(s+1)^3} \tag{5.19}$$

Here u is the input, y is the output, and s is the Laplace operator for differentiation. We wish a state-space model of the system. Writing out Equation (5.19) we get,

$$\begin{aligned}(s+1)^3 y &= u \Leftrightarrow \\ \left(s^3 + 3s^2 + 3s + 1\right) y &= u \Leftrightarrow \\ \dddot{y} + 3\ddot{y} + 3\dot{y} + y &= u\end{aligned}$$

By the choice of state variables $x_1 = y$, $x_2 = \dot{y}$, and ignoring the usual third substitution $x_3 = \ddot{y}$ we rewrite into a set of equations

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \ddot{y} \\ &= \frac{1}{3}\left(-\dddot{y} - 3\dot{y} - y\right) + \frac{1}{3}u\end{aligned}$$

or

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{3}\left(-\dddot{y} - 3x_2 - x_1\right) + \frac{1}{3}u\end{aligned}$$

We now have not only two equations in x_1 and x_2, but also a $\dddot{y}$ appearing in the second equation. Nevertheless, the situation is similar to the previous situation arising from integral action, therefore we can follow the same strategies.

Following strategy 1, we set $\dddot{y} = k$, where k is an arbitrary constant. Now we can plot the vector field spanned by x_1 and x_2 or, alternatively, e and $\dot{e}$ since $e = Ref - x_1$ (for 'position' control) and $\dot{e} = -x_2$. The control signal u will be a function of e and $\dot{e}$. Possibly also $\int edt$, and in that case we make yet another approximation according to strategy 1.

Following strategy 2, we simulate a step response and thus the value of $\dddot{y}$ is known at all points of the phase plane trajectory. The control signal u is also known, and we can plot the trajectory and velocity components in the $(e, \dot{e})$-plane. That is 'only' a section of a multi-dimensional trajectory, but the movement in the $(e, \dot{e})$-plane is still an important facet of the closed-loop behaviour.

5.6 Practical Considerations

Up to this point, we have ignored a number of practical points for the sake of clarity. Thus we now turn to some suggestions and heuristics to consider in a practical implementation.

Limit cycles

A *limit cycle* is a periodic motion in a quasi-steady state of the system. It is a unique feature of nonlinear systems. It can occur, for example, if there is a deadzone such that the plant drifts away from the equilibrium until a certain point, where the controller takes action to move it back towards the equilibrium.

In the phase plane a limit cycle is defined as a closed and isolated curve. The trajectory must be both closed, indicating its periodic nature, and isolated indicating that nearby trajectories converge towards it or diverge from it.

Limit cycles can appear in fuzzy control systems. Take for example the double integrator controlled by an FPD controller in Figure 5.13. The control surface, to emphasize the phenomenon, has a flat plateau near the centre. This causes a standing oscillation in quasi-steady state, and it shows in the phase plane as a closed curve. The figure also shows the equivalent crisp PID control. By comparison, the fuzzy controller has warped the vector field near the centre such that the arrows run around the centre, rather than towards the centre.

Saturation in the universes

Saturation in the premise universes affects the dynamic response, and if it is unintended, it can be removed. For the sake of illustration Figure 5.14 shows what happens in the case of the double integrator $1/s^2$; with the scaling factor $\alpha = 2$ the error E saturates. The corresponding PID gains are the same as in Figure 5.11.

The error signal saturates early in the transient response as the control surface plot shows. The result is larger overshoot, slower response, and a sluggish load response.

Quantization

When the premise universes in a controller are discrete, it is possible to calculate all possible combinations of $E(n)$ with $CE(n)$ before putting the controller into operation; that is

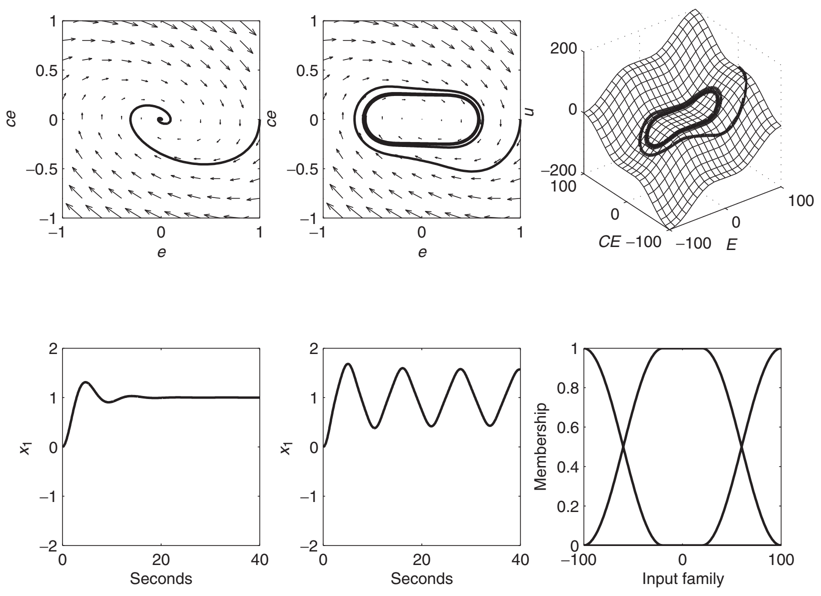


Figure 5.13: Limit cycle. The controller can cause a standing oscillation. (figs2limit.m)

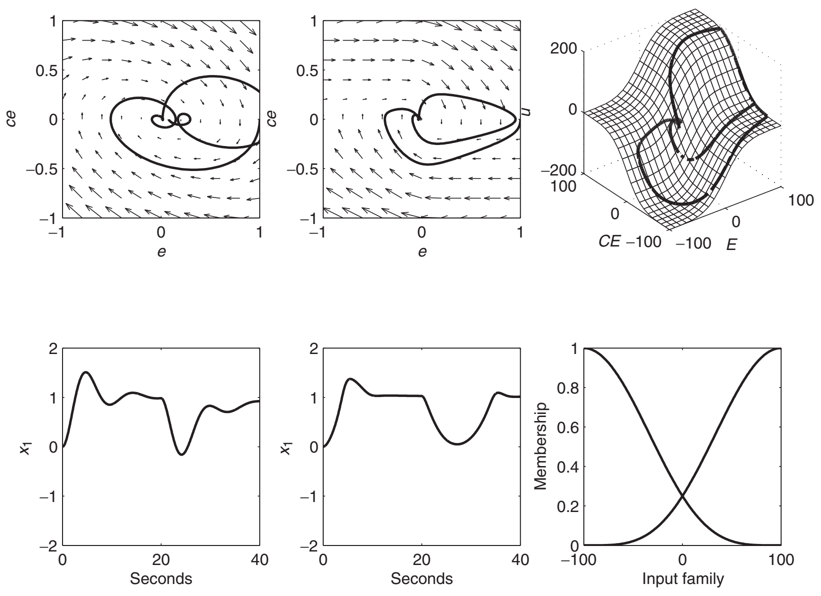


Figure 5.14: Saturation. FPD+I control of $1/s^2$, with $\alpha = 2$. (figs2load.m)

the principle underlying the table-based controller. The quantization in the table affects the performance.

With a quantum size of 10 in the premise universes the resolution is 5 % of full range. The result is a limit cycle, because the controller allows the plant to drift within a cell in the lookup table, until it shifts to another cell with a different control action. This is especially noticeable in steady state. The amplitude of the limit cycle is affected by the input gains.

There are three ways to reduce the limit cycle: (1) to increase the input gains, (2) to make the resolution finer, and (3) to use interpolation. The first option makes the controller more sensitive to small deviations from the set point, and may also cause saturation that changes the dynamics. A variant of the second option is to use a new table with a finer resolution when the plant approaches the set point. The third option removes the quantization effect altogether.

Noise

Measurement noise causes an oscillatory behaviour. If the noise frequency is high it will drive the phase plot to the edges of the *change in error* universe; beyond, the input universe will limit spikes. If the noise causes instability or disturbs the control, a filter may be necessary.

Another option is to use the fuzzy incremental controller, FInc. The integrator in the final block of the block diagram makes the control signal more smooth.

5.7 Summary

We set out to find a tuning procedure for fuzzy controllers, and the procedure we arrived at is given here. It refers to an FPD+I controller, because it is the most general controller, but it covers the other controllers as well with slight modifications.

Procedure Fuzzy controller design

1. *Design a crisp PID controller.* Use a PID tuning method to find K_p, $1/T_i$, and T_d.

2. *Replace it with a linear fuzzy controller.* Use the FPD+I configuration, transfer K_p, $1/T_i$, and T_d to *GE*, *GCE*, *GIE*, and *GU* (*GCU*) using Table 5.1. Run the controller, and check for saturation in the premise universes. When it is removed, by means of $\alpha-$scaling, check that the closed-loop response is exactly the same with the fuzzy rule base replaced by a pure summation.

3. *Make it nonlinear.* Follow strategy 1 to get an overview of the vector field in the phase plane. Adjust the rules to change the vector field locally. Then follow strategy 2 for finer adjustments to the rule base.

4. *Fine-tune it.* Use hand-tuning: use *GE* to improve the rise time, use *GCE* to dampen overshoot, and use *GIE* to remove any steady-state error.

The accuracy of the vector field in step 3 decreases with increasing order of the plant to be controlled; it is an approximation for coping with plants of order higher than two.

Another strategy would be to apply adaptation, see the following chapter, in the hopes that it will be faster and easier to tune the closed-loop system.

5.8 Notes and References

Aracil, García-Cerezo, and Ollero introduced in 1988 the idea of studying the vector field, with the plant and controller represented by vectors (Aracil *et al.* in Driankov *et al.* 1996). Being geometrical, the approach lends itself to graphical illustration. Furthermore, the robustness can be assessed by means of indices of stability (Aracil, Ollero and Garcia-Cerezo in Driankov *et al.* 1996).

The book by Slotine and Li (1991) contains an easy introduction to phase plane analysis. Atherton (1975) gives the topic a deeper treatment with many interesting interpretations. The traditional focus is on isoclines in order to draw the phase trajectories, while today this is easily done by computer.

An early attempt to apply the phase plane to estimate stability is based on the linguistic phase plane (Braae and Rutherford 1979a, 1979b). The phase plane has been applied in attempts to overcome time delays by shifting the phase plane (Li and Gatland 1995) or rotating it (Tanaka *et al.* 1991).

6

The Self-Organizing Controller

Mamdani and his PhD students developed the self-organizing controller (SOC) as an extension to the original fuzzy controller. They called it *self-organizing*, because it was able to adjust the control table of a fuzzy controller without human intervention. It was developed specifically to cope with time delays in a plant.

At that time, the distinction between the terms *self-organizing* and *adaptive* was unclear. Today, self-organization refers to a changing structure, such as the connections in a network. In daily conversation *to adapt* means to modify according to changing circumstances, for example, 'they adapted themselves to the warmer climate'. Today, the SOC is regarded as an adaptive controller.

An example of an adaptive controller is a ship's roll damper that adapts to changing sea waves. An adaptive controller is intuitively a controller that can modify its behaviour if the plant varies nonlinearly, or if the disturbances vary nonlinearly, or if the same controller is to be used under different conditions. Lacking a formal definition, a pragmatic definition is as follows:

Definition *Adaptive controller*. This is a controller with adjustable parameters and a mechanism for adjusting the parameters (Åström and Wittenmark 1995).

An adaptive controller has a distinct architecture consisting of two loops: (1) an inner control loop, which is the basic feedback loop; and (2) an outer loop, which adjusts the parameters of the controller (Figure 6.1).

6.1 Model Reference Adaptive Systems

In a *model reference adaptive system (MRAS),* a reference model specifies the desired output of the inner loop. There are some general assumptions concerning the blocks in Figure 6.1:

Plant We assume that we know the structure of the plant, although the parameters are unknown. For linear plants, this means knowing the number of zeros and the number of poles, but not the exact locations. For nonlinear plants, this means knowing the structure of the equations, but some parameters are unknown.

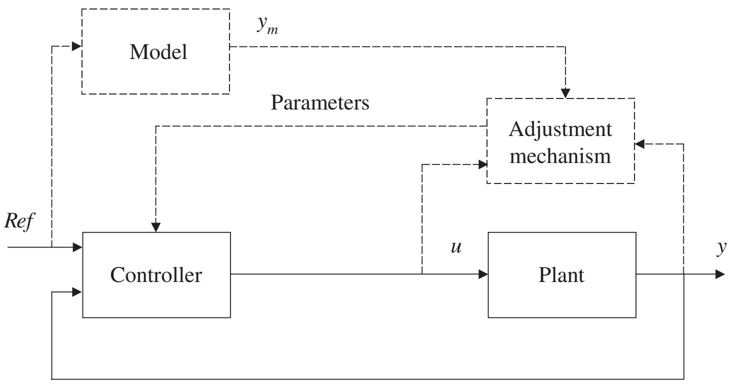


Figure 6.1: Model reference adaptive system, MRAS. The outer loop (dashed line) adjusts the controller parameters such that the error $\varepsilon = y - y_m$ becomes close to zero.

Model The reference model provides a *performance specification.* The choice of the reference model is part of the design. It should reflect the desired performance, such as rise time, settling time, and overshoot. This ideal behaviour should be achievable, which means there are some constraints on the structure of the reference model (order, type, relative degree) given the assumed structure of the plant. The model returns the *desired response* y_m to a reference signal *Ref.*

Controller The controller should have *perfect model-following* capacity, that is, the closed-loop transfer function of the inner loop should be identical to the model, when the plant is known. This imposes some constraints on the structure of the controller. When the plant parameters are unknown, the controller parameters will achieve perfect model-following asymptotically.

Adjustment mechanism The *adjustment mechanism* adjusts the parameters of the controller. The *model error* ε, which is the deviation of the plant response from the desired response, governs the adjustment of the parameters. A strategy, the *adaptation law*, adjusts the parameters such that the error ε – or a function of ε – is minimized.

It is a challenge to design the adjustment mechanism such that the inner-loop system remains stable. If the performance criterion can be optimized analytically, it may also be possible to analytically guarantee stability. Noise may cause problems, however, for the adaptation mechanism.

First-order systems

We shall use an example (Åström and Wittenmark 1995) with a first-order plant to illustrate how an adaptive control system can be designed, and to introduce the concepts in model reference adaptive control. The first-order system is chosen not only owing to its simplicity but also because we shall use the results later in connection with the fuzzy SOC.

The room temperature, the level of liquid in a tank being emptied, or the discharge of an electronic flash, are examples of systems that may be approximated by a first-order differential equation

$$\dot{y}(t) = -ay(t) + bu(t)$$

where $u(t)$ is the control signal, $y(t)$ is the measured output, and a and b are constant plant parameters that are unknown. The negative sign of a emphasizes that the plant is stable when a is positive.

Let a first-order reference model specify the desired performance of the inner-loop system,

$$\dot{y}_m(t) = -a_m y_m(t) + b_m Ref(t)$$

where a_m and b_m are constant parameters and $Ref(t)$ is a bounded reference signal. The parameter a_m must be positive in order to ensure that the model is stable, and b_m is chosen positive without loss of generality. Using Laplace notation, where s is the Laplace variable and ignoring initial conditions, the model is represented by

$$\begin{aligned} sy_m(s) &= -a_m y_m(s) + b_m Ref(s) \Leftrightarrow \\ (s + a_m)y_m(s) &= b_m Ref(s) \Leftrightarrow \\ y_m(s) &= \frac{b_m}{(s + a_m)} Ref(s) \end{aligned}$$

In other words, the model has the transfer function

$$G_m(s) = \frac{y_m(s)}{Ref(s)} = \frac{b_m}{(s + a_m)} \tag{6.1}$$

In order to simplify the notation, we shall omit the argument (s) in the Laplace domain and (t) in the time domain, except when we wish to distinguish parameters from signals.

The objective is to form a control law and an adaptation law such that the model error $\varepsilon = y - y_m$ converges to zero.

The MIT rule

One strategy is to define a positive objective function to be minimized according to the MIT rule (invented at the Massachusetts Institute of Technology, MIT). Define an objective function

$$J(\theta) = \frac{1}{2}\varepsilon^2$$

that depends on the adjustable parameter θ. This objective function is always non-negative, and minimizing $J(\theta)$ will also minimize ε. The MIT rule suggests to change the parameter θ in the direction of the negative gradient of J, that is,

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = -\gamma \frac{\partial J}{\partial \theta} = -\gamma \varepsilon \frac{\partial \varepsilon}{\partial \theta}$$

The *adaptation gain* γ is a parameter that the designer chooses; it determines the step length during the iterative search for the minimum. The *sensitivity derivative* $\partial\varepsilon/\partial\theta$ is the sensitivity of the error to changes in the parameter θ; if the sensitivity is large, then the parameter change will be large – all other parameters being equal. The parameter change also depends directly on the magnitude of the error ε. Assuming that the parameter θ changes much slower than the other dynamics, derivatives can be calculated assuming θ as a constant. For example $\mathrm{d}(\theta u)/\mathrm{d}t = (\mathrm{d}\theta/\mathrm{d}t)u + \theta(\mathrm{d}u/\mathrm{d}t)$ using the product rule for differentiation, but if θ is assumed as a constant, an approximation is $\theta(\mathrm{d}u/\mathrm{d}t)$. In general, approximation is necessary to evaluate the sensitivity derivative.

The objective function is arbitrary. For example, the alternative function $J(\theta) = |\varepsilon|$ results in the parameter rate of change $\mathrm{d}\theta/\mathrm{d}t = -\gamma(\partial\varepsilon/\partial\theta)sign(\varepsilon)$. An even simpler adaptation law is $\mathrm{d}\theta/\mathrm{d}t = -\gamma sign(\partial\varepsilon/\partial\theta)sign(\varepsilon)$ which avoids the evaluation of the sensitivity derivative.

Choice of control law

To continue the design, choose the control law

$$u(t) = \theta_1 Ref(t) - \theta_2 y(t) \tag{6.2}$$

This choice of structure allows for perfect model-following. The closed-loop dynamics are

$$\begin{aligned}\dot{y}(t) &= -ay(t) + bu(t)\\ &= -ay(t) + b(\theta_1 Ref(t) - \theta_2 y(t))\\ &= (-a - b\theta_2)y(t) + b\theta_1 Ref(t)\end{aligned} \tag{6.3}$$

Indeed, if the plant parameters were known, the parameter values

$$\begin{aligned}\theta_1 &= \frac{b_m}{b}\\ \theta_2 &= \frac{a_m - a}{b}\end{aligned}$$

would make the dynamics of the inner loop and the model identical, and provide a zero model-following error. Since a and b are unknown, the controller must achieve this objective adaptively.

Choice of adaptation law

To apply the MIT rule, introduce the model error

$$\varepsilon = y - y_m$$

The sensitivity derivative is

$$\frac{\partial\varepsilon}{\partial\theta} = \frac{\partial(y - y_m)}{\partial\theta} = \frac{\partial y}{\partial\theta}$$

since the model output y_m does not depend on the controller parameter θ. It follows from Equation (6.3) that the closed-loop plant output y is determined by

$$y(s) = \frac{b\theta_1}{s + a + b\theta_2} Ref(s) \tag{6.4}$$

Now differentiate with respect to the controller parameters:

$$\frac{\partial y}{\partial \theta_1} = \frac{b}{s + a + b\theta_2} Ref(s)$$

$$\frac{\partial y}{\partial \theta_2} = -\frac{b^2\theta_1}{(s + a + b\theta_2)^2} Ref(s) = -\frac{b}{s + a + b\theta_2} y(s)$$

The switch from signal $Ref(s)$ to signal $y(s)$ in the last equation used Equation (6.4). But the equations still contain the plant parameters a and b, which are unknown; we would like to dispose of these parameters. An approximation is therefore required. We observe that the denominator $s + a + b\theta_2$ is the closed-loop denominator, which will be the same as the model denominator under perfect model-following. Therefore, we apply the approximation

$$s + a + b\theta_2 \approx s + a_m$$

The approximation will be reasonable when the parameters are close to their correct values. We thus achieve the following adaptation laws for adjusting the parameters:

$$\frac{\mathrm{d}\theta_1}{\mathrm{d}t} = -\gamma\varepsilon \frac{\partial \varepsilon}{\partial \theta_1} = -\gamma\varepsilon \left[\frac{b}{s + a + b\theta_2} Ref(s) \right] \approx -\gamma\varepsilon \left[\frac{b}{s + a_m} Ref(s) \right]$$

$$\frac{\mathrm{d}\theta_2}{\mathrm{d}t} = -\gamma\varepsilon \frac{\partial \varepsilon}{\partial \theta_2} = -\gamma\varepsilon \left[-\frac{b}{s + a + b\theta_2} y(s) \right] \approx \gamma\varepsilon \left[\frac{b}{s + a_m} y(s) \right]$$

The final step is to dispose of the unknown parameter b. Since it is constant, it can be absorbed in the adaptation gain, but we have to know the sign of b in order to use the correct sign for the adaptation gain. This is a mild condition in practice, since it can be determined experimentally: if a positive step on the reference increases $y(t)$, then the sign is positive, and if it decreases, the sign is negative. By the substitution $\gamma b = \gamma' sign(b) a_m$, we achieve the adaptation laws

$$\frac{\mathrm{d}\theta_1}{\mathrm{d}t} \approx -\gamma' sign(b)\varepsilon \left[\frac{a_m}{s + a_m} Ref(s) \right]$$

$$\frac{\mathrm{d}\theta_2}{\mathrm{d}t} \approx \gamma' sign(b)\varepsilon \left[\frac{a_m}{s + a_m} y(s) \right]$$

The expressions in square brackets are filtered versions of the signals $Ref(s)$ and $y(s)$, such that the steady-state gain of the filter is 1. Thus the change in θ_1 depends on *Ref*, which is the signal that θ_1 multiplies, and the change in θ_2 depends on y, which is the signal that θ_2 multiplies; this is a consequence of the sensitivity derivative.

Convergence

The behaviour of the system is now illustrated in simulation. The parameters are chosen to be $a = 1$, $b = 0.5$, $a_m = b_m = 2$, $\gamma = 1$. The input signal is a square wave with amplitude 1, and Figure 6.2 shows the response. The control is quite good as early as the second or third step in the reference. The plant output (row 1 in the figure, solid line) follows the model (dashed) with increasing accuracy. The control signal (row 2) becomes more and

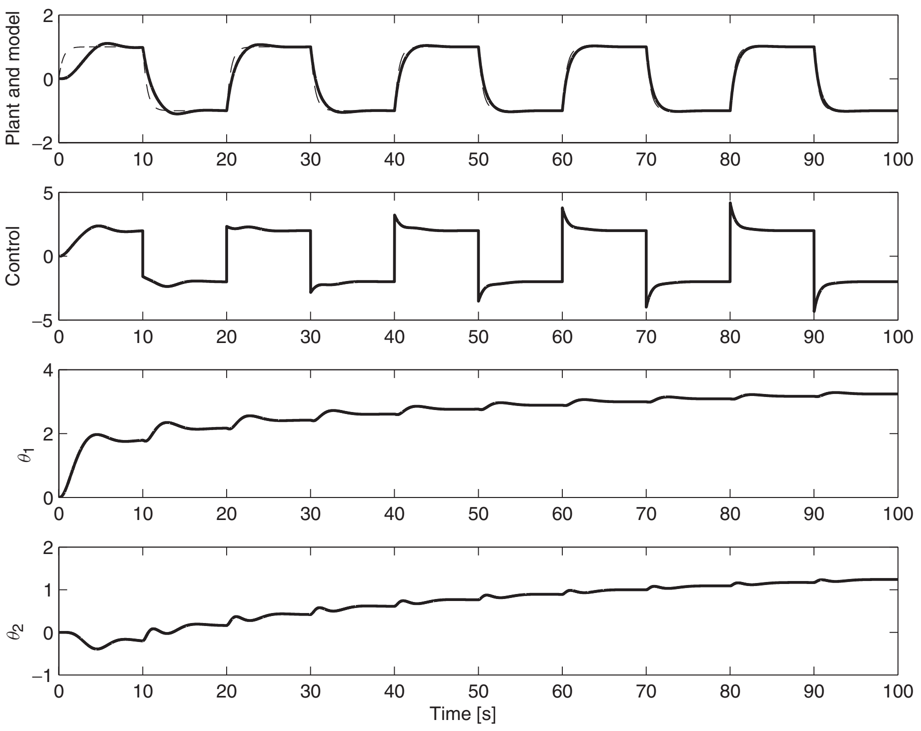


Figure 6.2: First-order MRAS with the parameters $a_m = b_m = 2$, $a = 1$, $b = 0.5$, $\gamma = 1$. The system is stable, and the plant output and the system output converges towards the model (top, dashed). (figadap1.m)

more articulated as the parameters θ_1 and θ_2 increase (rows 3 and 4). They are far from the correct values $\theta_1 = 4$ and $\theta_2 = 2$, but they will converge given a longer simulation period.

The system is stable and it converges towards perfect model-following, which can be shown by means of Lyapunov theory (see, for example, Åström and Wittenmark 1995, or Slotine and Li 1991).

In general, the model error can converge even though the parameters do not converge to their correct values; the controller may work for a subset of frequencies if only the ratio of θ_1 to θ_2 is correct. The parameters change more when the control signal switches, which illustrates the requirement for an input signal sufficiently rich in frequencies, so-called persistently exciting. It is particularly apparent in the fuzzy SOC, which makes local changes in the state space; the system must, in principle, visit the whole state space in order to complete the adaptation.

6.2 The Original SOC

The SOC has a hierarchical structure in which the inner loop is a table-based controller and the outer loop is the adjustment mechanism (Figure 6.3). The idea behind the adaptation

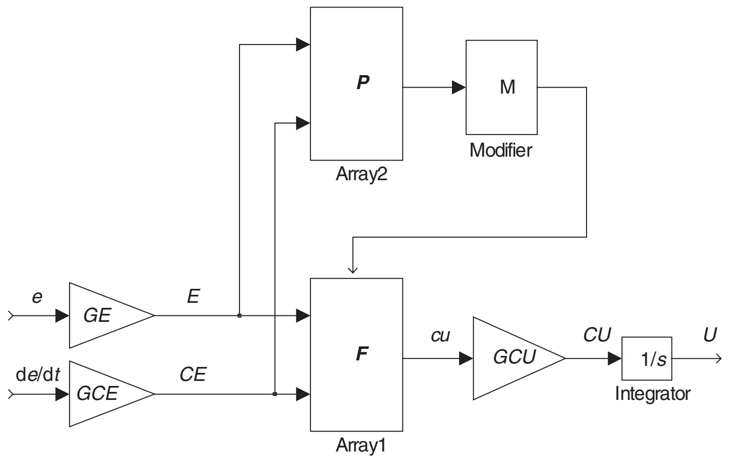


Figure 6.3: Self-organizing controller, SOC. The outer loop adjusts the controller lookup table ***F***according to the performance measure in ***P***.

is to let the adjustment mechanism update the values in the control table ***F*** on the basis of the current performance of the controller. If the performance is undesirable, the cell in ***F*** that is believed to be responsible receives a penalty, such that next time that cell is visited, the performance specification will be closer to or even equal to zero. Explanations based on the MRAS are as follows:

Plant The developers of SOC made no assumptions about the structure of the plant, but they required that the step response be monotonous, in other words, a minimum-phase system. The plant can be nonlinear, and may contain a large deadtime. The dominant time constant and any deadtime must be known approximately, for example, as a result of a step response experiment.

Model A *performance specification* p governs the magnitude of each change to ***F***. The performance specification depends on the current values of error and the change in error. The performance specifications are preset numbers organized in a *performance table* ***P*** the size of ***F***. Table ***P*** expresses what is desirable, or undesirable rather, in a transient response. Table ***P*** can be built using linguistic rules, but is often built by hand; see the two examples in Figures 6.4 and 6.5. The same performance table ***P*** may be used with a different plant, without prior knowledge of the plant, since it just expresses the *desired* transient response.

Controller The inner loop is an incremental, digital controller. The change in output CU_n at the current time n is added to the control signal U_{n-1} from the previous time instant, modelled as a summation in Figure 6.3. The two inputs to the controller are the error e and its derivative $\dot{e}$. The signals are multiplied by tuning gains, GE and GCE respectively, before entering the rule base block ***F***. In the original SOC, ***F*** is a lookup table, possibly generated from a linguistic rule base. The table lookup value, called *change in output, cu*, is multiplied by the output gain GCU and digitally integrated to become the control signal U. The integrator block can be omitted, however; then the table value is usually called u (not cu), multiplied by a gain GU (not GCU), and used directly as the control signal U (not CU).

Adjustment mechanism The outer loop monitors *error* E and *change in error* CE, and it modifies table $\boldsymbol{F}$ through a *modifier* algorithm **M**. The controller can start from scratch with an $\boldsymbol{F}$-table of zeros; but it will converge faster towards a steady table, if $\boldsymbol{F}$ is primed with sensible numbers to begin with.

The SOC was developed in the 1970s on the basis of sound engineering intuition in an attempt at simplicity. Compared to an MRAS, it lacks the principle of minimizing an objective function, and analytical treatment is difficult. On the other hand, the assumptions are milder.

Adaptation law

The SOC adapts the system in accordance with the desired response. At the sampling instant n,

1. it records the deviation from the desired state and
2. it corrects table $\boldsymbol{F}$ accordingly.

The performance table $\boldsymbol{P}$ evaluates the current state and returns a performance specification $\boldsymbol{P}(i_n, j_n)$. Index i_n corresponds to E_n, such that $E_n = \mathcal{U}_e(i_n)$, where $\mathcal{U}_e$ is the input universe. Index j_n corresponds to CE_n, such that $CE_n = \mathcal{U}_{ce}(j_n)$, where $\mathcal{U}_{ce}$ is the other input universe.

Figures 6.4 and 6.5 are examples of early performance tables. Intuitively, a zero performance specification $\boldsymbol{P}(i_n, j_n) = 0$ implies that the state (E_n, CE_n) is satisfactory. If the performance specification is non-zero that state is unsatisfactory, to a degree indicated by the magnitude of the number p_n. When p_n is non-zero, the modifier **M** assumes the control signal must be adjusted by the amount p_n. The current control signal cannot be held responsible, however, because it takes some time before a control action shows up in the plant output.

The simple strategy is to go back a number of samplings d in time to correct an earlier control signal. The modifier must therefore know the time lag in the plant. The integer d is comparable to the plant time lag; here d is called the *delay-in-penalty* (in the literature it is called delay-in-*reward,* but that seems slightly misleading).

It is required that an increase in plant output calls for an adjustment of the control signal *always* in the same direction, whether it be an increase or a decrease. The modifier thus assumes that the plant output depends monotonously on the input.

The precise adjustment mechanism is simply

$$u_{n-d} = u_{n-d} + p_n$$

In terms of the tables $\boldsymbol{F}$ and $\boldsymbol{P}$, the adjustment rule is

$$\boldsymbol{F}(i, j)_{n-d} = \boldsymbol{F}(i, j)_{n-d} + \boldsymbol{P}(i, j)_n$$

	CE												
E	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6
−6	−6	−6	−6	−6	−6	−6	−6	0	0	0	0	0	0
−5	−6	−6	−6	−6	−6	−6	−6	−3	−2	−2	0	0	0
−4	−6	−6	−6	−6	−6	−6	−6	−5	−4	−2	0	0	0
−3	−6	−5	−5	−4	−4	−4	−4	−3	−2	0	0	0	0
−2	−6	−5	−4	−3	−2	−2	−2	0	0	0	0	0	0
−1	−5	−4	−3	−2	−1	−1	−1	0	0	0	0	0	0
0	−4	−3	−2	−1	0	0	0	0	0	1	2	3	4
1	0	0	0	0	0	0	1	1	1	2	3	4	5
2	0	0	0	0	0	0	2	2	2	3	4	5	6
3	0	0	0	0	2	3	4	4	4	4	5	5	6
4	0	0	0	2	4	5	6	6	6	6	6	6	6
5	0	0	0	2	2	3	6	6	6	6	6	6	6
6	0	0	0	0	0	0	6	6	6	6	6	6	6

Figure 6.4: Performance table adapted from Procyk and Mamdani (1979). Note the universes.

	CE												
E	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6
−6	−6	−6	−6	−6	−6	−6	−6	−5	−4	−3	−2	−1	0
−5	−6	−6	−6	−6	−5	−4	−4	−4	−3	−2	−1	0	0
−4	−6	−6	−6	−5	−4	−3	−3	−3	−2	−1	0	0	1
−3	−6	−6	−5	−4	−3	−2	−2	−2	−1	0	0	1	2
−2	−6	−5	−4	−3	−2	−1	−1	−1	0	0	1	2	3
−1	−5	−4	−3	−2	−1	−1	0	0	0	1	2	3	4
0	−5	−4	−3	−2	−1	0	0	0	1	2	3	4	5
1	−3	−2	−1	0	0	0	0	1	1	2	3	4	5
2	−2	−1	0	0	0	1	1	1	2	3	4	5	6
3	−1	0	0	0	1	2	2	2	3	4	5	6	6
4	0	0	0	1	2	3	3	3	4	5	6	6	6
5	0	0	1	2	3	4	4	4	5	6	6	6	6
6	0	1	2	3	4	5	6	6	6	6	6	6	6

Figure 6.5: Performance table adapted from Yamazaki (1982).

Notice the subscripts: The time subscript n denotes the current sampling instant, and subscript $n - d$ denotes the sampling instant d samples earlier.

Table 6.1: Performance data for Example 6.2.1.

	Time instance				
Variable	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
E	6	3	1	0	−1
CE		−3	−2	−1	−1
u		0	−1	−1	−2
p		0	0	0	−1

Example 6.2.1 *Adjustment mechanism*

Assume $d = 2$ and variables as in Table 6.1 after a step in the setpoint. From $t = 1$ to $t = 4$, the plant moves up towards the setpoint. Apparently it follows the desired trajectory, because the performance specification p is 0. At $t = 5$, error *changes sign indicating an overshoot, and the performance table reacts by dictating $p_5 = -1$. Since d is 2, the entry to be adjusted in $\boldsymbol{F}$ will be at the position corresponding to $t = n - d = 5 - 2 = 3$. At that sampling instant, the state was $(E_3, CE_3) = (1, -2)$ and the element $\boldsymbol{F}(i, j)_3$ was $u_3 = -1$. The adjusted entry is $u_3 = u_3 + p_5 = -1 - 1 = -2$, which is to be inserted into $\boldsymbol{F}(i, j)_3$.*

6.3 A Linear Performance Measure

The original performance tables in Figures 6.4 and 6.5 were built by hand, by trial and error. Presumably, if the numbers in the table $\boldsymbol{P}$ are small in magnitude, many updates are required before $\boldsymbol{F}$ converges to a steady table. On the other hand, if the numbers in $\boldsymbol{P}$ are large in magnitude, the convergence should be faster, but it may also be unstable. The following analysis leads to a simplification of the adaptation mechanism.

Procyk and Mamdani (Figure 6.4) preferred to keep the zeros in a z-shaped region, while Yamazaki (Figure 6.5) kept the zeros in a more or less diagonal band. As the zeros indicate no penalty, those states are desired. Assuming that the system stays within a zero band along the diagonal, what does it imply?

Noticing the numbers on the axes, a zero diagonal is equivalent to keeping the sum of the rule base inputs at zero in that region. It is a discrete version of the continuous relation,

$$GE * e + GCE * \dot{e} = 0 \tag{6.5}$$

The right-hand side corresponds to the entries ’zero’ in the table, the term $GE * e$ is the rule base input E (error), and $GCE * \dot{e}$ is the rule base input CE (change in error). This is an ordinary differential equation that determines the ratio between the variable e and its own time derivative. Its solution is

$$e(t) = e(0)\exp(-\frac{t}{GCE/GE})$$

That is a first-order exponential decay with time constant GCE/GE; the error e will gradually decrease by a fixed ratio determined by the time constant, and after $t = GCE/GE$ seconds it decreases to $\exp^{-1}$ or 37 % of the initial value $e(0)$. Note that the equation concerns the error $e = Ref - y$, such that the plant output y after a step input will reach 63 % of its final value in $t = GCE/GE$ seconds.

To interpret, the modifier **M** tries to adapt the system to a first-order behaviour of the error.

The z-shaped table (Figure 6.4) is more generous, because it allows a zero slope to begin with, for instance $\boldsymbol{P}(100, 0) = 0$, and some overshoot near the final value, around the centre of the table. This behaviour is similar to a second-order transient response.

Apparently, a simpler way to build a performance table is to use Equation (6.5) replacing the zero on the right-hand side by p. The time constant of the desired response would thus be GCE/GE, but this is an impractical constraint, because we would like to tune GCE and GE independently, without affecting the desired time constant directly.

Instead we introduce the *desired time constant* τ. A simple approximation to the table $\boldsymbol{P}$ is the performance specification

$$p_{(n)} = \gamma \left(e_{(n)} + \tau * \dot{e}_{(n)}\right) * T_s \tag{6.6}$$

The *adaptation gain* γ affects the convergence rate. The index n is the time instant in discrete time, and T_s is the sample period. In fact, Equation (6.6) is an incremental update – because the output is a change to an existing value – and therefore the multiplication by the sample period T_s. The longer the sample period, the fewer the updates within a given time span and the larger the penalty per update in order to keep the convergence rate independent of the choice of sample period.

6.4 Example with a Long Deadtime

Consider the plant

$$y(s) = \exp^{-9s} \frac{1}{s(s+1)^2} u(s)$$

It is difficult to control, because the deadtime of 9 seconds is large (approximately 1/3) compared to the apparent time constant of the plant, and the plant contains an integrator, which has a destabilizing effect. The strategy is to do the following:

1. tune a fuzzy controller with a linear control surface;
2. start adaptation without changing the tuning; and
3. measure the performance of the resulting response.

At time $t = 0$, the reference changes abruptly from 0 to 1 and after 500 seconds, a load of 0.05 units is forced on the system. We are using the sampling time $T_s = 1$ seconds.

Tuning

Since the test system includes a load, it will be necessary to maintain a non-zero control signal in the steady state while the load is on; therefore, the controller must contain integral action. The choice is between a fuzzy PD+I controller and a fuzzy incremental controller, FInc; the latter is chosen, because it has one gain less to tune, and derivative action is of little help when there is a large deadtime. The control table is a 21×21 lookup table, which is an arbitrary choice of resolution. The initial table is linear and the lookup table is with interpolation.

A loose hand-tuning of an incremental PD controller, equivalent to a PI controller, resulted in the gains

$$K_p = 2 * 10^{-3}$$
$$T_d = 20$$

The gain K_p is the proportional gain and T_d is the differential gain, but since the controller output is integrated, T_d is in effect the gain on the error term and K_p is the gain on the integral term, corresponding to the integral gain $1/T_i$. The equivalent fuzzy gains are

$$GE = 100$$
$$GCE = GE * T_d = 2000$$
$$GCU = \frac{Kp}{GE} = 2 * 10^{-5}$$

Figure 6.6 shows the response. The FInc response is the same as the incremental PD controller response, which is omitted to save space. During the initial seconds of the response, the deadtime appears as a horizontal section. When the load comes on at $t = 500$, there is a large dip in the response followed by some oscillation. It is possible to tune the system better, and achieve less oscillation and less overshoot, but it is fairly difficult to achieve damping of both the reference step response and the load response at the same time.

Adaptation

Three parameters control the adaptation: the desired time constant τ in seconds, the delay-in-penalty T_{dip} in seconds, and the adaptation gain γ. The dashed line in Figure 6.6 shows a

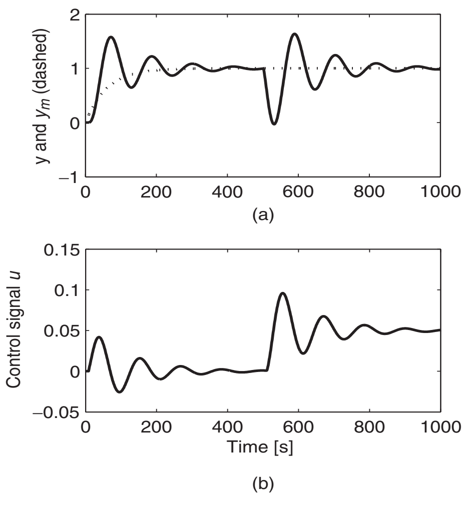


Figure 6.6: Incremental PD control of plant $e^{-9s}/s\,(s+1)^2$. The plant output y, desired response y_m (top), and the control signal u (bottom) before sef-organization. (figsocdelay.m)

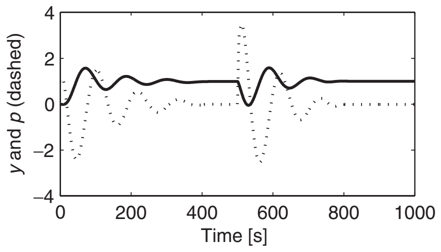


Figure 6.7: SOC performance. Step response after first run of self-organization (solid) in comparison with the performance measure (dashed). (figsocdelay.m)

response corresponding to a desired time constant of $\tau = 60$ s. This choice is arbitrary, but it appears to be achievable. If the desired time constant is too fast, the response will begin to oscillate during adaptation. The desired response is slower than the current response, but without the overshoot and oscillations.

A plot (Figure 6.7) of the performance specification, Equation (6.6), with an arbitrary adaptation gain to be determined in a moment, shows the phase advance of the performance specification p relative to the plant output y. The figure shows the response of the plant, like the previous figure, but this time with the performance specification as a dashed line. A closer study shows that the performance specification peaks about 14–17 s earlier than the plant output, due to the differential term in Equation (6.6).

By inspection it is seen that the time constant of the closed-loop system is about 27 s. This is the time it takes to reach 63 % of the steady-state value, after the deadtime of 9 s, during the initial seconds of the response. We may thus reason that it takes approximately $9 + 27 = 36$ s for a control signal to show on the output of the plant. We can now make a guess at the delay-in-penalty T_{dip}, but we observed that the performance specification is *ahead* of the plant output; thus T_{dip} may be less than 36 s. Indeed, $T_{\text{dip}} = 20$ s is chosen; the difference is 16 s, which is within the interval 14–17 s. With a sampling period of 1 s, the delay-in-penalty is $d = T_{\text{dip}}/T_s = 20/1 = 20$ samples.

The adaptation gain is, by trial and error, set to $\gamma = 1.5$. The strategy is to choose it to be as high as possible, but still keep the response steady. With these choices, the adaptation can be switched on.

Performance

The SOC is able to dampen the oscillations after a few runs, and the performance improves after each run. Figure 6.8 shows the step response after 29 runs. The oscillations have disappeared and the load dip at $t = 500$ is smaller. The system is clearly nonlinear. The plant more or less follows the desired response, but owing to the deadtime and the higher order of the plant, it is unable to follow the desired response perfectly.

The modified control surface contains several jagged peaks (see Figure 6.9). During each run the controller visited a number of cells in the lookup table, sometimes the same cell several times; the accumulated changes result in the sharp peaks. One might expect

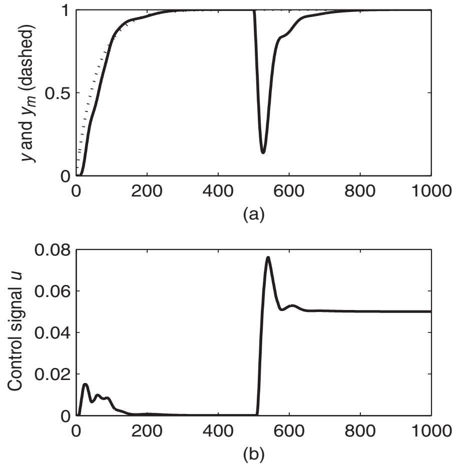


Figure 6.8: After 29 runs. The plant output y is close to the desired output y_m (a). The control signal u (b indicates less control effort than during the first run. (figsocdelay.m)

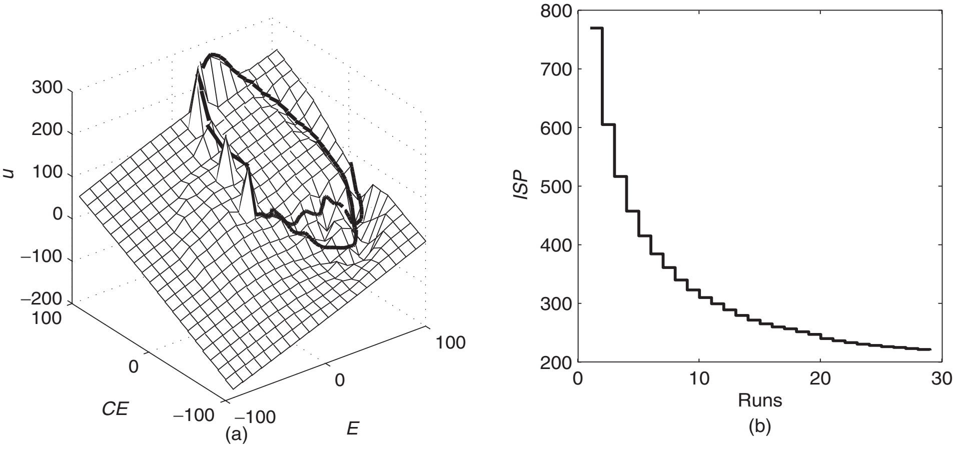


Figure 6.9: Convergence. After 29 runs the control surface (a) has an irregular shape. The trajectory passes through regions with large changes. Only visited regions are changed. The decreasing performance measure ISP (b) indicates convergence. (figsocdelay.m)

jumps in the control signal as a consequence, but in fact it is rather smooth thanks to the integrator in the output end of the incremental controller.

When the response is close to the desired response there will be only small corrections to the control table. The SOC seeks to minimize the performance specification p_n at each time instant n, and the discretized Integral Squared Performance (ISP) index,

$$ISP = \sum_n p_n^2 * T_s$$

the activity during each run. The smaller the ISP the better. The plot of ISP for each run (Figure 6.9b) levels out, indicating that the adaptation is stable.

Experiments on laboratory models have shown that various plants can be stabilized, but the rule modifier is sensitive to noise on the plant output – it cannot tell if a poor performance specification is due to noise or an undesired state.

6.5 Tuning and Time Lock

In practice, additional questions arise, for example, how to tune the gains, how to choose the design parameters, how to stop the adaptation if it behaves unintentionally, and how to cope with noise. Such problems may seem inferior, but they do deserve attention, since they may finally stand in the way for a successful implementation. The SOC works 'surprisingly well' to quote Mamdani, but cases exist where the SOC becomes impractical.

Tuning of the SOC parameters

We associate the term *tuning* with the adjustment of the gains and design parameters, while the term *self-organization* is associated with adjusting the parameters of the control table.

GE, GCE, G(C)U The controller gains must be set near some sensible settings, with the exact choice being less important compared to a conventional fuzzy controller. Imagine, for example, that the output gain is lowered between two training sessions. The adjustment mechanism will then compensate by producing an $\boldsymbol{F}$-table with numbers of larger magnitude. Even if the input gains were changed, it would still manage to adapt. It is therefore possible to start with a linear $\boldsymbol{F}$-table, and set the gains loosely according to a PID tuning rule or hand-tuning. That is a good starting point for the self-organization.

Desired time constant $\boldsymbol{\tau}$ The smaller the τ, the faster the desired response. If it is too small, however, the inner loop cannot possibly follow the desired trajectory, but the modifier will try anyway. As a result the $\boldsymbol{F}$-table winds up, and the consequence is an overshoot. The lower bound for τ is when this overshoot starts to occur. A plant with a time constant τ_p and a deadtime T_p requires that

$$T_p \leq \tau \leq T_p + \tau_p$$

A value somewhat smaller than the right-hand side of the inequality is often achievable, because the closed-loop system is usually faster than the open-loop system.

Delay-in-penalty d Parameter d, measured in samplings, should be chosen with due respect to the sample period. The delay should, in principle, be the desired time constant divided by the sample period and rounded to the nearest integer:

$$d = round(\tau / T_s)$$

The results are often better, however, with a somewhat smaller value.

Adaptation gain $\boldsymbol{\gamma}$ The larger the γ, the faster the $\boldsymbol{F}$-table winds up, but if it is too large, the training becomes unstable. A reasonable magnitude of a large p is less than one-fifth of the maximum value in the output universe. This rule results in the upper bound:

$$\gamma \leq \frac{0.2 * |\mathbf{F}(i, j)|_{\max}}{|(e_n + \tau * \dot{e}_n)|_{\max} * T_s}$$

Compared to conventional fuzzy control, the tuning task is shifted from having to tune accurately $\{GE, GCE, G(C)U\}$ to tuning $\{\tau, d, \gamma\}$ loosely.

Time lock

The delay-in-penalty d causes a problem when the reference or the load changes abruptly.

Consider this case. If the error and the change in error, for a time period longer than d samples, have been zero, then the controller is in an allowed state (the steady state). Suddenly there is a disturbance from the outside, the performance specification p becomes non-zero, and the modifier will modify the $\boldsymbol{F}$-table d samples back in time. It should not do so, however, because the state was acceptable there. The next time the controller visits that state, the control signal will fluctuate. The problem appears also after step changes in the reference or the load.

A solution is to implement a *time lock* (Jespersen 1981). The time lock stops the self-organization for the next d samples; if it is activated at the sampling instant T_n, self-organization stops until the sampling instant $T_n + d + 1$. In order to trigger the time lock, it is necessary to detect disturbances, abrupt changes in the load, and abrupt changes in the reference. If these events cannot be measured directly and fed forward into the SOC, it is necessary to try and detect it in the plant output. If the output changes more than a predefined threshold, or if the combination of *error* and *change in error* indicates an abrupt change, then activate the time lock.

The time lock is conveniently implemented by means of a *queue,* a vector of length $d + 1$ samples.

$$\mathbf{q} = \left\langle (i, j)_{n-d}, \ldots, (i, j)_{n-1}, (i, j)_n \right\rangle$$

The element to update in $\boldsymbol{F}$ is indicated by the front (leftmost) element in $\mathbf{q}$. At the next sampling instant, the current index pair is appended to the (rightmost) end of the queue, while the first index pair is removed. If an event triggers the time lock, flush the queue; that is, reset all cells to zero. New updates to $\boldsymbol{F}$ will only be possible after the queue is full again, that is, when the first element in the queue is non-zero, $d + 1$ samplings after the flush.

An extra precaution is to protect the centre of $\boldsymbol{F}$ from updates; this is the steady state $(E, CE) = (0, 0)$ in which the control action must always be zero.

If necessary the time lock can be activated each time the modifier makes a change in $\boldsymbol{F}$. In other words, the modifier waits to see the influence of the change before making a new change.

6.6 Analytical Derivation of the Adaptation Law*

The adjustment mechanism can be interpreted from the viewpoint of model reference adaptive control, which provides a deeper insight into the mechanism.

Reference model

Equation (6.6) expresses the adjustment to the parameters. When the adjustment is zero, the inner-loop behaviour is ideal, or

$$0 = \gamma(e_m + \tau s e_m) \tag{6.7}$$

We have switched to continuous time for convenience, and s is the Laplace variable. The subscript m refers to the model and $e_m = Ref - y_m$. Insertion and division by γ yields

$$\begin{aligned} 0 &= Ref - y_m + \tau s(Ref - y_m) \Rightarrow \\ y_m &= \frac{1}{\tau s + 1} Ref \end{aligned} \tag{6.8}$$

We assumed that *Ref* is constant in the last step, so that its derivative vanishes. Thus the desired behaviour, specified by a zero penalty in Equation (6.6), is equivalent to having a reference model, which returns a first-order desired response y_m to a step in the reference *Ref* in accordance with Equation (6.8).

Adjustment mechanism

The question now is, what is the interpretation of p_n in Equation (6.6) when it is *non*zero? In continuous time, Equation (6.6) is

$$p = \gamma(e + \tau s e) \tag{6.9}$$

Subtracting Equation (6.7) from (6.9) we have

$$\begin{aligned} p &= \gamma(e + \tau s e) - \gamma(e_m + \tau s e_m) \\ &= \gamma(Ref - y + \tau s\,(Ref - y)) - \gamma(Ref - y_m + \tau s(Ref - y_m)) \\ &= \gamma((y_m - y) + \tau s(y_m - y)) \end{aligned}$$

By insertion of the model error $\varepsilon = y - y_m$,

$$\begin{aligned} p &= \gamma((-\varepsilon) + \tau s(-\varepsilon)) \Leftrightarrow \\ p &= -\gamma(\varepsilon + \tau s \varepsilon) \end{aligned}$$

We have thus transformed the performance specification p into an expression in the model error ε. The expression in parentheses is the model error ε plus a term proportional to the change in error $s\varepsilon$; this can be interpreted as the first-order prediction $\widehat{\varepsilon}$ of the future model error τ seconds ahead of current time. Thus

$$p = -\gamma\widehat{\varepsilon}, \qquad \widehat{\varepsilon} = (\varepsilon + \tau s \varepsilon) \tag{6.10}$$

*Can be skipped in a first reading.

To recapitulate, the performance specification p is equivalent to a parameter adjustment proportional to the predicted model error $\widehat{\varepsilon}$, with adaptation gain γ. Since the adjustment p is an incremental adjustment $\mathrm{d}\theta/\mathrm{d}t$, the parameter θ itself is the integral of Equation (6.10) plus the initial value. The adjustment mechanism is thus a proportional-integral type, instead of just integral.

What happens if we use $\widehat{\varepsilon}$ (instead of ε) from Equation (6.10) in the objective function? For comparison, we will take the previously worked out first-order example, and proceed in the same manner, step by step.

Error function The model error function is

$$\widehat{\varepsilon} = \varepsilon + \tau s \varepsilon = y - y_m + \tau s(y - y_m) \tag{6.11}$$

Objective function We choose the quadratic objective function

$$J(\theta) = \frac{1}{2}\widehat{\varepsilon}^2$$

Minimizing $J(\theta)$ will minimize $\widehat{\varepsilon}$. It may happen that $\widehat{\varepsilon}$ is zero while ε is non-zero. Nevertheless $\varepsilon + \tau s \varepsilon = 0$ is a differential equation that will drive ε to zero, given that τ is positive. The adaptation stops, but the model error tends to zero in an equilibrium.

MIT rule The MIT rule suggests the parameter change

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = -\gamma\frac{\partial J}{\partial \theta} = -\gamma\widehat{\varepsilon}\frac{\partial \widehat{\varepsilon}}{\partial \theta}$$

Model The model is, as before,

$$\dot{y}_m(t) = -a_m y_m(t) + b_m Ref(t) \tag{6.12}$$

Control law We choose the same control law,

$$u(t) = \theta_1 Ref(t) - \theta_2 y(t)$$

Closed-loop dynamics With that control law, the closed-loop dynamics of the inner loop is

$$\dot{y}(t) = (-a - \theta_2)y(t) + b\theta_1 Ref(t) \tag{6.13}$$

On transfer function form,

$$y(s) = \frac{b\theta_1}{s + a + b\theta_2} Ref(s) \tag{6.14}$$

Perfect model-following We know from the previous investigation that the control law allows perfect model-following.

Sensitivity derivative The sensitivity derivative is

$$\frac{\partial \widehat{\varepsilon}}{\partial \theta} = \frac{\partial (y - y_m + \tau s(y - y_m))}{\partial \theta} = \frac{\partial (y + \tau s y)}{\partial \theta} = (1 + \tau s)\,\frac{\partial y}{\partial \theta}$$

since the model output y_m is independent of the controller parameter θ. Differentiating y in Equation (6.14) with respect to the controller parameters,

$$\frac{\partial y}{\partial \theta_1} = \frac{b}{s + a + b\theta_2} Ref(s)$$

$$\frac{\partial y}{\partial \theta_2} = -\frac{b^2\theta_1}{(s + a + b\theta_2)^2} Ref(s)$$

Closed-loop substitution In the equation for $\partial y / \partial \theta_2$ we make use of the inner-loop transfer function from Equation (6.13),

$$\frac{\partial y}{\partial \theta_2} = -\frac{b^2\theta_1}{(s + a + b\theta_2)^2} Ref(s) = -\frac{b}{s + a + b\theta_2} y(s)$$

Adaptation law We arrive at the following adaptation laws for adjusting the parameters:

$$\frac{\mathrm{d}\theta_1}{\mathrm{d}t} = -\gamma \widehat{\varepsilon} \frac{\partial \widehat{\varepsilon}}{\partial \theta_1} = -\gamma \widehat{\varepsilon}\,(1 + \tau s) \left[\frac{b}{s + a + b\theta_2} Ref(s) \right]$$

$$\frac{\mathrm{d}\theta_2}{\mathrm{d}t} = -\gamma \widehat{\varepsilon} \frac{\partial \widehat{\varepsilon}}{\partial \theta_2} = -\gamma \widehat{\varepsilon}\,(1 + \tau s) \left[-\frac{b}{s + a + b\theta_2} y(s) \right]$$

Approximation Again, we approximate the inner-loop characteristic polynomial with the model characteristic polynomial,

$$s + a + b\theta_2 \approx s + a_m$$

After approximation, the adaptation laws are therefore

$$\frac{\mathrm{d}\theta_1}{\mathrm{d}t} \approx -\gamma \widehat{\varepsilon}\,(1 + \tau s) \left[\frac{b}{s + a_m} Ref(s) \right]$$

$$\frac{\mathrm{d}\theta_2}{\mathrm{d}t} \approx \gamma \widehat{\varepsilon}\,(1 + \tau s) \left[\frac{b}{s + a_m} y(s) \right]$$

Disposal of plant parameters Since the unknown parameter b is constant, it can be absorbed in the adaptation gain, but we have to know the sign of b in order to use the correct sign for the adaptation gain. By the substitution $\gamma b = \gamma' sign(b) a_m$,

$$\frac{\mathrm{d}\theta_1}{\mathrm{d}t} \approx -\gamma' sign(b) \widehat{\varepsilon}\,(1 + \tau s) \left[\frac{a_m}{s + a_m} Ref(s) \right] \tag{6.15}$$

$$\frac{\mathrm{d}\theta_2}{\mathrm{d}t} \approx \gamma' sign(b) \widehat{\varepsilon}\,(1 + \tau s) \left[\frac{a_m}{s + a_m} y(s) \right] \tag{6.16}$$

The square brackets are filtered versions of the signals *Ref* and y, and the steady-state gain of the filter is 1.

Cancellation Since τ is the prediction horizon for the predicted model error, it is a design parameter. If we choose

$$\tau = \frac{1}{a_m}$$

the adaptation laws are simply,

$$\frac{d\theta_1}{dt} \approx -\gamma' sign(b)\widehat{\varepsilon}(t)Ref(t) \tag{6.17}$$

$$\frac{d\theta_2}{dt} \approx \gamma' sign(b)\widehat{\varepsilon}(t)y(t) \tag{6.18}$$

We notice that the adaptation law Equation (6.17) is similar to Equation (6.10), except for the factor $sign(b)$, which is just a convenience, and the factor *Ref*. We have earlier assumed that *Ref* is constant, and therefore it can be absorbed in γ'.

We can conclude that our choice of error function $\widehat{\varepsilon}$ in Equation (6.11) caused a cancellation of the denominators in Equations (6.15) and (6.16), which simplified the adaptation law; this is possible whenever the polynomial of the model error function Equation (6.11) is chosen to be proportional to the denominator polynomial of the model transfer function G_m.

Controller

Does this work for a fuzzy controller, and how does it compare with the SOC? So far we have chosen a control law that has a structure different from the one used in fuzzy PD control. Let us now try a PD structure and model the SOC adjustment by an increment δ to the control table.

Error function The model error function is

$$\widehat{\varepsilon} = \varepsilon + \tau s \varepsilon = y - y_m + \tau s(y - y_m) \tag{6.19}$$

Objective function We choose the quadratic objective function

$$J(\theta) = \frac{1}{2}\widehat{\varepsilon}^2$$

Minimizing $J(\theta)$ will minimize $\widehat{\varepsilon}$ and ε.

MIT rule The MIT rule suggests the parameter changes

$$\frac{d\theta}{dt} = -\gamma\frac{\partial J}{\partial\theta} = -\gamma\widehat{\varepsilon}\frac{\partial\widehat{\varepsilon}}{\partial\theta}$$

Model The model is, as before,

$$\dot{y}_m(t) = -a_m y_m(t) + b_m Ref(t) \tag{6.20}$$

with the transfer function

$$\frac{y_m(s)}{Ref(s)} = \frac{b_m}{s + a_m}$$

Control law We choose the control law,

$$u = (GE * e + GCE * \dot{e} + \delta) * GU$$

Notice the adjustable parameter δ. We assume the SOC starts from a linear controller with $\delta = 0$. For each cell in the lookup table, δ models the adjustment that the SOC makes. The control law is a local model of each cell in the lookup table. There are thus a number of δ's, one for each cell, but we will only consider one such cell in order to keep the derivations simple and transparent.

Closed-loop dynamics With that control law, the closed-loop dynamics of the inner loop are

$$\begin{aligned}\dot{y}(t) &= -ay(t) + bu(t)\\ &= -ay(t) + b * (GE * e + GCE * \dot{e}(t) + \delta * u_0(t)) * GU\\ &= -b * GCE * GU * \dot{y}(t) - (a + b * GE * GU)y(t)\\ &\quad +b * \delta * u_0(t) * GU + b * GE * GU * Ref(t)\end{aligned} \tag{6.21}$$

Here we have assumed that $Ref(t)$ is constant, so that its time derivative vanishes. We have modelled δ as a gain on an external input signal u_0, which is always equal to 1. Solving with respect to y, and using the Laplace variable,

$$\begin{aligned}y(s) = &\frac{b * \delta * GU}{(1 + b * GCE * GU)s + (a + b * GE * GU)}u_0(s)\\ &+\frac{b * GE * GU}{(1 + b * GCE * GU)s + (a + b * GE * GU)}Ref(s)\end{aligned} \tag{6.22}$$

Perfect model-following We hope to achieve perfect model-following. Assume for a moment that $\delta = 0$. By a comparison of Equations (6.21) and (6.20),

$$\frac{a + b * GE * GU}{1 + b * GCE * GU} = a_m$$
$$\frac{b * GE * GU}{1 + b * GCE * GU} = b_m$$

Notice that GU multiplies gains GE and GCE, so we can regard it as a scaling factor. Reorganize into two equations in two unknowns,

$$b * GE * GU - a_m * b * GCE * GU = a_m - a$$
$$b * GE * GU - b_m * b * GCE * GU = b_m$$

The determinant $-b^2 b_m + b^2 a_m$ is non-zero if $b_m \neq a_m$ and then the system has a unique solution for $GE * GU$ and $GCE * GU$. Therefore, perfect model-following is possible as long as the model is such that $b_m \neq a_m$. The solution requires certain settings of especially GE and GCE, and since these are tuned by other means, it is unlikely that we will actually achieve perfect model-following.

Sensitivity derivative The sensitivity derivative is

$$\frac{\partial \widehat{\varepsilon}}{\partial \theta} = \frac{\partial (y - y_m + \tau s(y - y_m))}{\partial \theta} = \frac{\partial (y + \tau s y)}{\partial \theta} = (1 + \tau s)\frac{\partial y}{\partial \theta}$$

since the model output y_m is independent of the parameter. Differentiating y in Equation (6.22) with respect to the adjustable parameter,

$$\frac{\partial y}{\partial \delta} = \frac{b * GU}{(1 + b * GCE * GU)s + (a + b * GE * GU)} u_0(s)$$

Closed-loop substitution We do not make use of the inner-loop transfer function.

Adaptation law We arrive at the following adaptation law for adjusting the parameter:

$$\frac{\mathrm{d}\delta}{\mathrm{d}t} = -\gamma \widehat{\varepsilon} \frac{\partial \widehat{\varepsilon}}{\partial \delta} = -\gamma \widehat{\varepsilon}(1 + \tau s)\left[\frac{b * GU}{(1 + b * GCE * GU)s + (a + b * GE * GU)} u_0(s)\right]$$

Approximation We approximate the inner-loop characteristic polynomial, the denominator in square brackets, with the model characteristic polynomial

$$s + a_m$$

After insertion, the adaptation law is

$$\frac{\mathrm{d}\delta}{\mathrm{d}t} = -\gamma \widehat{\varepsilon}(1 + \tau s)\left[\frac{b * GU}{s + a_m} u_0(s)\right]$$

Disposal of plant parameters Since the unknown parameter b is constant, it can be absorbed in the adaptation gain, but we have to know the sign of b in order to use the correct sign for the adaptation gain. By the substitution $\gamma b = \gamma' sign(b) a_m$,

$$\frac{\mathrm{d}\delta}{\mathrm{d}t} \approx -\gamma' sign(b) \widehat{\varepsilon}(1 + \tau s)\left[\frac{a_m}{s + a_m} u_0(s)\right] * GU \qquad (6.23)$$

The square bracket above is a filtered version of the signal u_0, and the steady-state gain of the filter is 1.

Cancellation Since τ is the prediction horizon for the predicted model error, it is a design parameter. If we choose

$$\tau = \frac{1}{a_m}$$

the adaptation law is simply ($u_0 = 1$)

$$\frac{\mathrm{d}\delta}{\mathrm{d}t} \approx -\gamma' sign(b) \widehat{\varepsilon}(t) * GU$$

This is similar to the adaptation law in the original SOC, apart from the factors GU and $sign(b)$. These could be absorbed in the adaptation gain, to make the two expressions identical, but they are left out for convenience and as an emphasis.

To summarize, the derivations are based on three observations. First, the performance specification Equation (6.9) implicitly expresses the desired behaviour ($p = 0$), which can be interpreted as a first-order reference model. Second, the performance specification ($p \neq 0$) is interpreted as the predicted model error τ seconds ahead of current time. Third, choosing τ equal to the model time constant $1/a_m$ simplifies the adaptation law.

The theoretical considerations show that the SOC adjustment mechanism minimizes an objective function. Since we are adjusting only parameter δ, and not the gains, we cannot guarantee perfect model-following or stability.

6.7 Summary

The adjustment mechanism is simple, but, in practice, the design is complicated by the time lock and noise precautions. Also, if the resolution of the control table is too low, the adaptive system may become unsteady or even unstable. Although the tuning of the traditional input and output gains (GE, GCE, $G(C)U$) is made less cumbersome by the adjustment mechanism, it introduces other parameters that are to be tuned (delay-in-penalty, learning gain, desired time constant). The tuning task is nevertheless easier, because these can be loosely tuned.

The SOC is based on the assumption that the plant responds monotonously (is minimum-phase), and an example showed that it works for a plant with a large deadtime and an integrator. There is no guarantee, however, that the adaptive system will be stable. To prove stability, further theoretical research is necessary, but will be difficult because the updates to the control table are local updates rather than global with regard to the whole state space. The following chapter looks at stability from a fundamental point of view.

One may ask where fuzzy logic enters the picture, since the adjustment law is an equation and the control table is automatically adjusted. Fuzzy logic is absent, but historically the design sprang from the fuzzy controller. Any control table, derived from fuzzy rules, can be used as the initial table to be adapted. It is conceivable, also, to use a performance specification based on fuzzy rules.

6.8 Notes and References

It is possible, and this has been done in practice, to adjust the consequence singletons in a Sugeno controller. This will make the adjustments less local, because each singleton affects a region of the phase plane determined by the premise membership functions of that rule. The corresponding adjustments to the control table cover several cells at the same time, and the modified control surface will be smoother. If there is sufficient computing time to execute a rule base, the control table does not have to be calculated at all.

Adaptive fuzzy controllers can be classified according to what they adjust. Systems that adjust the gains are called self-tuning, and systems that adjust the rule base are self-organizing. For an overview of the techniques, see Driankov *et al.* (1996). A third class adjusts the parameters of the membership functions, both on the premise side and on the conclusion side (e.g. Jang *et al.* 1997).

The performance measure in Equation (6.9)

$$p = \gamma(e + \tau se)$$

is similar to the switching function

$$s(y, \dot{y}) = my + \dot{y}$$

in sliding mode control (Edwards and Spurgeon 1998). It would be interesting to explore whether there is a mathematical connection.

Research in adaptive control started in the early 1950s. Control engineers have tried to agree on a formal definition of adaptive control. For example, in 1973 a committee under the Institute of Electrical and Electronics Engineers (IEEE) proposed a vocabulary including the terms 'self-organizing control (SOC) system', 'parameter adaptive SOC', 'performance adaptive SOC', and 'learning control system' (Åström and Wittenmark 1995). These terms were never widely accepted, however, but the story does explain why the adaptive *fuzzy* controller is called a self-organizing controller (SOC).

The fuzzy self-organizing controller (Mamdani and Baaklini 1975, Procyk and Mamdani 1979, Yamazaki and Mamdani 1982), was developed specifically to cope with deadtime. To the inventors, it was a further development of the original fuzzy controller (Assilian and Mamdani, 1974a, 1974b).

7

Stability Analysis by Describing Functions

A *describing function* is an approximate transfer function for a nonlinear element, and, in general, it depends on both frequency and amplitude. It can be viewed as an equivalent linear gain, amplitude and frequency dependent, which is optimal in the sense of a least squares linearization of the nonlinearity. From the describing function, we achieve the *frequency response*.

A sinusoidal input to a linear system in steady state generates a sinusoidal response of the same frequency. Even though it is of the same frequency, it differs in amplitude and phase angle from the input. The difference is a function of the frequency, and the magnitude frequency response is the ratio of the output sinusoid magnitude to the input sinusoid magnitude. The phase response is the difference in phase angle between the output and the input sinusoids. The frequency response is powerful, since it describes the system performance completely.

We are concerned with a feedback loop with a structure as in Figure 7.1. The controller is based on the error signal e, and it is placed in the forward path. The controller is in general nonlinear.

The response of a *nonlinear* system is a composite function depending on amplitude as well as frequency. It is therefore necessary to examine the response by means of its expansion into a Fourier series, consisting of several sinusoidal terms.

The complex ratio of the fundamental frequency of the output to the sinusoidal input is the *describing function*.

The frequency response, particularly in the form of the Nyquist plot, provides (1) a clear criterion for stability, (2) a prediction of limit cycles, and (3) a graphical visualization. These are useful analysis tools.

7.1 Describing Functions

Certain assumptions have to be fulfilled in order to apply the describing function method (Slotine and Li 1991, Atherton 1975):

Foundations of Fuzzy Control Jan Jantzen
© 2007 John Wiley & Sons, Ltd. ISBN: 0-470-02963-3

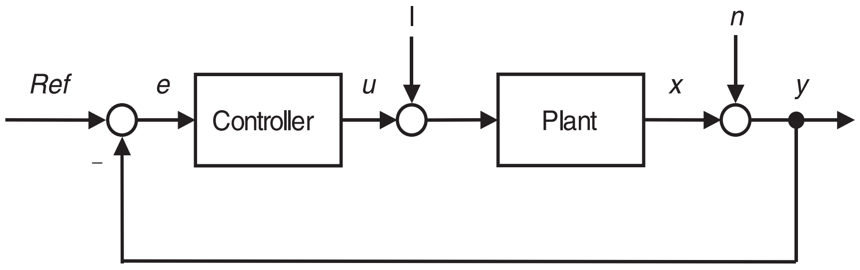


Figure 7.1: Feedback loop with load l and noise n.

Single nonlinear element The method is developed for a single nonlinearity, and in our case the controller is the nonlinearity (Figure 7.1). The method requires that the model of the plant be linear. If not, the block diagram must somehow be transformed.

Time-invariant nonlinearity The Nyquist criterion applies only to linear, time-invariant systems. We shall restrict the analysis to the FPD+I controller. It is dynamic, since it depends on the derivative of the error and the integral error; but it does not change over time, and is thus time-invariant.

Filtering hypothesis The describing function is only valid for sinusoidal inputs. In a closed-loop operation, the output signal, which contains higher-order components, is fed back and mixed with the reference signal. It is therefore unclear whether the input to the controller is sinusoidal, but for a step on the reference the assumption is valid if the plant acts like a filter. The output of the nonlinearity contains higher-order harmonics, besides the fundamental frequency, but we consider only the fundamental frequency as an approximation. The approximation is good when the plant acts as a low-pass filter, suppressing higher-order frequencies. Many physical plants have this property (the degree of the numerator polynomial is less than the degree of the denominator polynomial). For example, if the plant is a double integrator, the content of the nth (odd) harmonic fed back to the controller is $1/n^3$; the third harmonic content is thus only 3.7 % of the fundamental component.

Let the input $x(t)$ to the nonlinearity be a sinusoid:

$$x(t) = A\sin(\omega t)$$

with amplitude A and angular frequency ω. The output $y(t)$ is periodic, but not sinusoidal. Its Fourier expansion, however, is an infinite sum of sinusoidal signals:

$$\begin{aligned}y(t) &= \sum_{n=0}^{\infty}[a_n\cos(n\omega t) + b_n\sin(n\omega t)]\\ &= a_0 + \sum_{n=1}^{\infty}[a_n\cos(n\omega t) + b_n\sin(n\omega t)]\end{aligned}$$

The summation is over the n integer multiples of the fundamental frequency. The first term ($n = 0$), corresponding to the coefficient a_0, is the mean value of the signal $y(t)$, the next term ($n = 1$) is the *fundamental frequency*, and the remaining terms ($n > 1$) are higher *harmonics*.

By the filtering hypothesis stated earlier, we neglect all higher harmonics and consider only the fundamental,

$$y_1(t) = a_1 \cos(\omega t) + b_1 \sin(\omega t) = M \sin(\omega t + \varphi)$$

which is a sinusoid with the same frequency as the input signal and amplitude M. The ratio of the fundamental to the input is the describing function:

$$N(A, \omega) = \frac{y_1}{x}$$

Its magnitude is

$$|N(A, \omega)| = \frac{M}{A}$$

and its phase angle is

$$\angle N(A, \omega) = \angle y_1 - \angle x = \varphi$$

The Fourier coefficients are the convolution integrals

$$a_0 = \frac{1}{2\pi} \int_0^{2\pi} x(t) \mathrm{d}(\omega t) \tag{7.1}$$

$$a_n = \frac{1}{\pi} \int_0^{2\pi} x(t) \cos(n\omega t) \mathrm{d}(\omega t) \tag{7.2}$$

$$b_n = \frac{1}{\pi} \int_0^{2\pi} x(t) \sin(n\omega t) \mathrm{d}(\omega t) \tag{7.3}$$

Intuitively, coefficient b_n is large if the signal $x(t)$ varies in phase with $\sin(n\omega t)$ and the area under the product curve is large; then the nth harmonic *in-phase component* (in-phase with the input) is large. Similarly, coefficient a_n is large if $x(t)$ varies in phase with $\cos(n\omega t)$; then the nth harmonic *quadrature component* (proportional to the derivative of the input sinusoid) is large. Coefficient a_0 amounts to the mean value of the signal, which is often zero. Coefficient b_0 vanishes. Furthermore,

$$M = \sqrt{a_1^2 + b_1^2}$$

$$\varphi = \tan^{-1}\left(\frac{a_1}{b_1}\right)$$

In general, the describing function depends on both amplitude A and frequency ω, but for static linearities it only depends on A.

Static nonlinearity

Figure 7.2 shows an example of a static nonlinearity. It is a proportional controller with the gain k, but it saturates when the amplitude of the error signal is larger than a. It is called

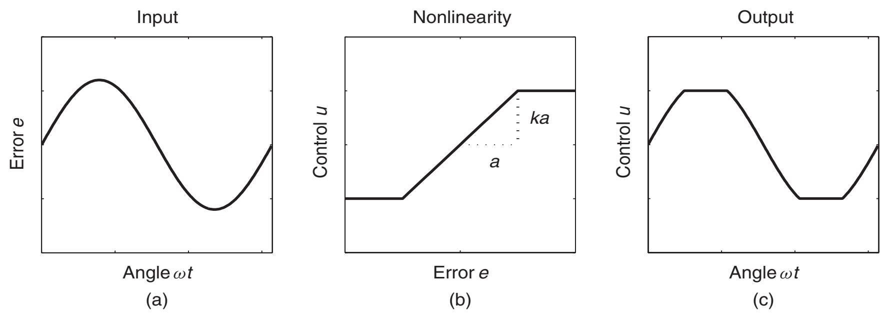


Figure 7.2: Ideal saturation $u = F(e)$. The nonlinearity (b) has the gain k in the linear region. A sinusoidal input $e = Asin(\omega t)$ (a) results in the nonlinear output $u = F(Asin(\omega t))$ (c). (figsat.m)

an *ideal saturation*, and is a *static nonlinearity*, because it depends on the error only, while a *dynamic nonlinearity* would depend on the time derivative of its input signal. The input $e(t)$ to the nonlinearity is a sinusoid:

$$e(t) = A\sin(\omega t)$$

with amplitude A and angular frequency ω. The resulting output $u(t)$ is periodic, but not sinusoidal.

Figure 7.3 shows the integrands of Equations (7.2) and (7.3) of the two components of the fundamental. The in-phase component is positive everywhere, while the quadrature component is symmetric about the zero line. A general property of nonlinearities described by an odd function $f(-x) = -f(x)$ is that all the coefficients a_n will be zero. The mean value, the DC-component (direct current) a_0, is zero since the output signal u is symmetric about the zero line. What remains is to integrate the sin integrand.

An analytical solution exists for the ideal saturation; it can be looked up in any textbook on nonlinear control (e.g. Slotine and Li 1991). The describing function for the ideal

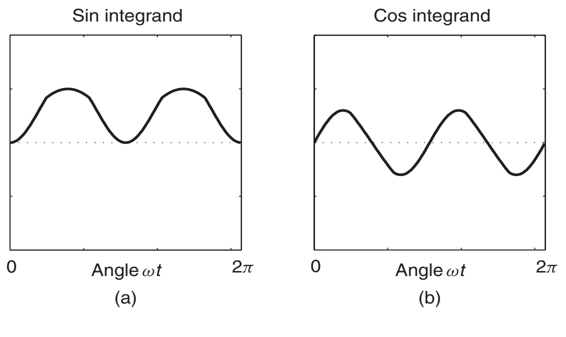


Figure 7.3: Fourier integrands. The in-phase integrand of the fundamental frequency (a) and the quadrature integrand (b). (figsat.m)

saturation is

$$N(A) = \frac{b_1}{A} = \frac{2k}{\pi}\left(\sin^{-1}\frac{a}{A} + \frac{a}{A}\sqrt{1 - \frac{a^2}{A^2}}\right)$$

Notice that the describing function, in this case, does not depend on the frequency ω. A plot of $N(A)/k$ versus A/a will show (Slotine and Li 1991) that (1) $N(A) = k$ if the amplitude A is within the linear range of the saturation, (2) $N(A)$ decreases as the amplitude A increases, and (3) there is no phase shift. The first observation is expected, and can be used to check the correctness of the integration. The second is intuitively correct, since for large amplitudes, the output has constant amplitude, and therefore the gain decreases. The third is also intuitively correct, because saturation does not cause any delay.

The fact that any signal can be expanded into a sum of sinusoidal components is extremely useful, and many researchers have calculated analytically or charted the describing functions of common linearities (see, for example, the tables in Atherton 1975, Šiljak 1968). Many other common nonlinearities have been calculated analytically, for example, quantizer, deadzone, relay, relay with deadzone, preload, piece-wise linear, gain changing, saturation with deadzone, hysteresis, and double-valued nonlinearities.

Numerical integration is another possibility. Although it is an approximation, rectangular integration can be a sufficiently good approximation if the sampling interval is small. Other more accurate integration methods exist (trapezoidal rule, Simpson's rule, see also function `quad` in Matlab).

7.2 Fuzzy PD Controller

We wish to find the describing function for the controller in Figure 7.1. It is a *dynamic nonlinearity*, since it depends on the derivative $\dot{e}$, or se in Laplace notation where $s = \mathrm{d}/\mathrm{d}t$ is the Laplace operator. It may also depend on the integrated error, but we will deal with that case later.

A fuzzy PD (FPD) controller is a nonlinearity $u = F(e, se)$ with two inputs, but the two inputs are dependent. We expect the describing function to be frequency dependent, owing to the differentiation, and therefore also the Fourier coefficient a_1 to be non-zero. Let the error signal be the sinusoid

$$e(t) = A\sin(\omega t)$$

Then the derivative signal is

$$\dot{e}(t) = \omega A\cos(\omega t)$$

and the controller output is

$$u = F(e, se) = F(A\sin(\omega t), \omega A\cos(\omega t))$$

Now rewrite the fundamental to reflect differentiation:

$$\begin{aligned} u_1(t) &= a_1\cos(\omega t) + b_1\sin(\omega t) \\ &= a_1\frac{1}{\omega}\frac{\mathrm{d}}{\mathrm{d}t}\sin(\omega t) + b_1\sin(\omega t) \end{aligned}$$

Using the Laplace operator s the describing function is

$$N(A,s) = \frac{u_1}{e} = \frac{b_1}{A} + \frac{a_1}{A}\frac{1}{\omega}s \tag{7.4}$$

This is a first-order transfer function depending on ω. The first term is the in-phase component and the second term is the quadrature component.

Its frequency response is obtained in the usual manner by the substitution $s = \mathrm{j}\omega$, where j is the complex variable, $j^2 = -1$,

$$N(A, j\omega) = \frac{b_1}{A} + j\frac{a_1}{A} \tag{7.5}$$

A notational detail indicates the difference between the transfer function in Equation (7.4) and the frequency response in Equation (7.5): $N(A,s)$ refers to the transfer function, and $N(A, j\omega)$ refers to the frequency response. To recall, the frequency response is the steady state response of an element to a sinusoidal input; it is determined by a mapping of the transfer function by replacing s by $j\omega$.

Example 7.2.1 *Linear FPD*

Let us try and find the describing function for a linear FPD controller, for which we already know the result. Given a linear fuzzy controller that acts like a summation, the controller signal is

$$U = F(GE * e, GCE * se) * GU$$

Here, GE and GCE are the input gains and GU is the output gain. Since the controller is linear, its transfer function is for small amplitudes (there is no saturation in the input universes)

$$\begin{aligned}\frac{U}{e} &= (GE + (GCE * s)) * GU \\ &= GE * GU + GCE * GU * s \end{aligned} \tag{7.6}$$

Given a sinusoidal input

$$e(t) = A\sin(\omega t)$$

the Fourier coefficient a_1 is from Equation (7.2),

$$\begin{aligned} a_1 &= \frac{1}{\pi}\int_0^{2\pi} F(e, se)\cos(\omega t)\,\mathrm{d}(\omega t) \\ &= \frac{1}{\pi}\int_0^{2\pi} GE * GU * A\sin(\omega t)\cos(\omega t)\,\mathrm{d}(\omega t) \\ &\quad + \frac{1}{\pi}\int_0^{2\pi} GCE * GU * \omega A\cos(\omega t)\cos(\omega t)\,\mathrm{d}(\omega t) \\ &= 0 + GCE * GU * \omega A\frac{4}{\pi}\int_0^{\pi/2}\cos^2(\omega t)\,\mathrm{d}(\omega t) \\ &= GCE * GU * \omega A\frac{4}{\pi} * \left[\frac{1}{2}(\omega t) + \frac{1}{4}\sin(2\omega t)\right]_0^{\pi/2} \\ &= GCE * GU * A\omega \end{aligned}$$

During integration, ω is considered a constant scaling parameter, since we integrate one period with ωt going from 0 to 2π. The coefficient b_1 is from (7.3)

$$\begin{aligned}
b_1 &= \frac{1}{\pi}\int_0^{2\pi} F(e, se)\sin(\omega t)\,\mathrm{d}(\omega t)\\
&= \frac{1}{\pi}\int_0^{2\pi} GE * GU * A\sin(\omega t)\sin(\omega t)\,\mathrm{d}(\omega t)\\
&\quad +\frac{1}{\pi}\int_0^{2\pi} GCE * GU * \omega A\cos(\omega t)\sin(\omega t)\,\mathrm{d}(\omega t)\\
&= GE * GU * A\frac{4}{\pi}\int_0^{\pi/2}\sin^2(\omega t)\,\mathrm{d}(\omega t) + 0\\
&= GE * GU * A\frac{4}{\pi}\left[\frac{1}{2}(\omega t) - \frac{1}{4}\sin(2\omega t)\right]_0^{\pi/2}\\
&= GE * GU * A
\end{aligned}$$

The describing function is thus

$$\begin{aligned}
N(A, s) &= \frac{b_1}{A} + \frac{a_1}{A}\frac{1}{\omega}s\\
&= GE * GU + GCE * GU * s
\end{aligned}$$

which is identical to Equation (7.6) as expected.

The example demonstrates how to evaluate the integrals. It also shows that the describing function reduces to the linear transfer function, when the controller is linear.

Example 7.2.2 *Linear FPD frequency response*

The FPD controller includes an ideal saturation since the input signals are confined to the input universes. We therefore expect a frequency response different from the crisp PD controller whenever saturation occurs. Let us examine the frequency response using numerical integration for convenience.

In the linear region, we derive from the previous example that the frequency response is

$$N(j\omega) = GE * GU + GCE * GU * (j\omega)$$

which is independent of amplitude. But when saturation is active, we expect the frequency response to be amplitude dependent. In terms of the Fourier coefficients, the frequency response is

$$\begin{aligned}
N(A, j\omega) &= \frac{b_1}{A} + j\frac{a_1}{A}\\
&= \frac{1}{A}\frac{1}{\pi}\int_0^{2\pi} F(A\sin(\omega t), \omega A\cos(\omega t))\sin(\omega t)\,\mathrm{d}(\omega t)\\
&\quad + j\frac{1}{A}\int_0^{2\pi} F(A\sin(\omega t), \omega A\cos(\omega t))\cos(\omega t)\,\mathrm{d}(\omega t)
\end{aligned}$$

With PD gains, say,

$$K_p = 1$$
$$T_d = 1$$

a set of equivalent fuzzy gain settings are

$$GE = 100$$
$$GU = K_p/GE = 0.01$$
$$GCE = T_d/GU = 100$$

The plot in Figure 7.4 shows the magnitude frequency response $|N(A, j\omega)|$ and the phase frequency response $\angle N(A, j\omega)$ for the amplitude $A = 1$. In comparison with the corresponding PD controller, the FPD controller has a constant magnitude above the frequency $\omega = 1$, while the crisp PD controller magnitude is ever increasing. The FPD controller has a maximum phase shift of about $52\,^\circ$, while the crisp PD has a maximal phase shift of $90\,^\circ$ at high frequencies. This is due to the saturation of the change in error signal. At $\omega = 1$ the change in error is

$$CE = GCE * se = GCE * A\omega\cos(\omega t) = 100 * 1 * 1 * \sin(t)$$

The amplitude is 100, which is the limit of the universe, and the phase is $45\,^\circ$ ($\omega = 1$). At higher frequencies, the saturation sets in.

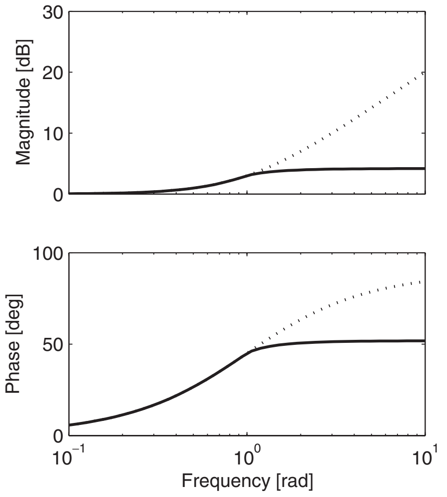


Figure 7.4: Bode diagram. Frequency response of a linear fuzzy PD controller (solid) in comparison with a crisp PD controller (dotted). The frequency axis is logarithmic and the magnitude response is in decibels (dB). (figfreqres.m)

The example clearly shows how the fuzzy PD cuts off at frequencies higher than a threshold determined by the input universe of change in error. The effect is useful for suppressing noise, but it also implies that the fuzzy PD achieves less phase advance at higher frequencies. At low frequencies, the performance is the same as that of the crisp PD controller.

Example 7.2.3 *Nonlinear FPD frequency response*

We can now compare different nonlinear controllers with respect to their frequency response. Figure 7.5 shows the frequency responses of the four standard fuzzy controllers. The four controllers have the same fuzzy gains:

$$GE = 100$$
$$GCE = 100$$
$$GU = 0.01$$

The amplitude is $A = 1$.The four surfaces are the standard surfaces: linear, saturation, deadzone, and quantizer (Chapter 5).

Figure 7.5 shows that overall, the saturation and the deadzone surfaces are the two extremes, with the linear and the quantizer in between. Relative to the linear surface, the saturation surface has a higher magnitude, while the deadzone surface has a lower magnitude. The phase advance for the saturation surface is high at low frequencies and low at higher frequencies, while for the deadzone surface the converse is true.

At $\omega = 1$ all surfaces have a phase advance of $45\,^{\circ}$. Thus, $a_1 = b_1$ ($\omega = 1, A = 1$, and $GE = GCE$).

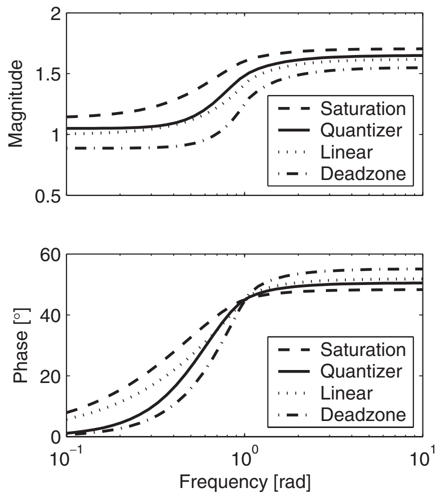


Figure 7.5: Bode diagram. Frequency response of the four standard control surfaces, one linear and three nonlinear. The frequency axis is logarithmic, but the magnitude and phase axes are linear. (figfreqnl.m)

Table 7.1: Frequency responses at low amplitude $A = 1$ and low, medium, and high frequency ω.

ω	PID	Linear	Saturation	Deadzone	Quantizer
0.1	$1 + j0.10$	$1 + j0.10$	$1.13 + j0.16$	$0.89 + j0.01$	$1.05 + j0.02$
1	$1 + j1$	$1 + j1$	$1.13 + j1.13$	$0.89 + j0.89$	$1.05 + j1.05$
10	$1 + j10$	$1 + j1.27$	$1.13 + j1.27$	$0.89 + j1.27$	$1.05 + j1.27$

Table 7.1 compares the frequency responses $N(A, j\omega)$ for the four surfaces numerically. The table reflects that the real part – corresponding to the proportional action – for each controller is independent of frequency. At high frequencies ($\omega = 10$), all surfaces have the same imaginary part, and at $\omega = 1$, all real parts are equal to the imaginary parts (45° phase).

Dividing the imaginary part by frequency in each case brings out each describing function.

The example shows that the four different surfaces have distinct characteristics and also that saturation in the universe of change in error occurs at higher frequencies.

7.3 Fuzzy PD+I Controller

Integral action affects the frequency response in a known manner for the PID controller, and naturally we would like to include integral action in the describing function of a fuzzy controller. We shall restrict ourselves to the fuzzy PD+I controller, since it encompasses the crisp PID controller.

To begin with, let us examine the ideal crisp PID controller

$$u = K_p \left(e + \frac{1}{T_i} \int e(t)\,\mathrm{d}t + T_d \frac{\mathrm{d}e}{\mathrm{d}t} \right)$$

with the transfer function

$$G_c(s) = \frac{u}{e} = K_p \left(1 + \frac{1}{T_i}\frac{1}{s} + T_d s \right) \tag{7.7}$$

Its frequency response is obtained by the substitution $s = j\omega$:

$$\begin{aligned} G_c(j\omega) &= K_p \left(1 + \frac{1}{T_i}\frac{1}{j\omega} + T_d j\omega \right) \\ &= Kp + jK_p \left(T_d\omega - \frac{1}{T_i\omega} \right) \end{aligned}$$

In other words, the integral term contributes phase lag to the frequency response at lower frequencies.

The fuzzy PD+I controller delivers the control signal

$$U = \left(F(GE * e, GCE * se) + GIE * \frac{1}{s} e \right) * GU$$

It is a linear combination. Therefore, the Fourier integrals can be evaluated for each summand and the results added together. Its describing function is simply

$$N(A,s) = \frac{U_1}{e} = \frac{b_1}{A} + \frac{a_1}{A}\frac{1}{\omega}s + GIE * GU * \frac{1}{s} \tag{7.8}$$

where a_1 and b_1 are the Fourier coefficients for the fuzzy PD controller. Compared with crisp PID control (see Equation (7.7)), the gain $GIE * GU$ plays the same role as $K_p * 1/T_i$, which checks with the equivalent gains for the linear fuzzy controller. Furthermore, we can deduce that b_1/A corresponds to K_p, and $a_1/\omega A$ to K_pT_d.

Its frequency response is obtained by the substitution $s = j\omega$,

$$\begin{aligned} N(A, j\omega) &= \frac{b_1}{A} + \frac{a_1}{A}\frac{1}{\omega}(j\omega) + GIE * GU * \frac{1}{j\omega} \\ &= \frac{b_1}{A} + j\left(\frac{a_1}{A} - \frac{GIE * GU}{\omega}\right) \end{aligned}$$

In comparison with the frequency response of the FPD controller, repeated here for convenience,

$$N(A, j\omega) = \frac{b_1}{A} + j\frac{a_1}{A} \tag{7.9}$$

the integral term contributes phase lag to the frequency response at lower frequencies.

7.4 The Nyquist Criterion for Stability

We are now in a position to study stability using the Nyquist criterion. Consider the system in Figure 7.1, and assume for a moment that it is linear. The closed-loop transfer function is

$$G_{cl} = \frac{G_cG_p}{1 + G_cG_p}$$

where G_c is the controller transfer function and G_p is the plant transfer function. The denominator is the characteristic polynomial, and the characteristic equation, which governs the stability, is

$$1 + G_cG_p = 0$$

The closed-loop system is marginally stable when the magnitude frequency response $|G_cG_p| = 1$, while, at the same time, the phase frequency response $\angle G_cG_p$ is $\pm 180^\circ$ degrees. This is equivalent to saying that the Nyquist diagram in the complex G_cG_p plane passes through the *critical point* $(-1, 0)$.

Figure 7.6 shows the Nyquist diagram of the plant $G_p = 1/(1+s)^3$. The curve does not pass through the critical point. The curve is drawn in the complex G_p-plane, which is a mapping of the complex s-plane. The plot is a polar plot, and each point on the curve represents a vector with magnitude $|G_p|$ and the angle $\angle G_p$ counted counterclockwise from the positive real axis. The frequency is implicit, but the upper half of the diagram corresponds to negative frequencies $(\omega < 0)$ while the lower half corresponds to positive frequencies $(\omega > 0)$. The arrows show the direction of increasing frequency. We focus on

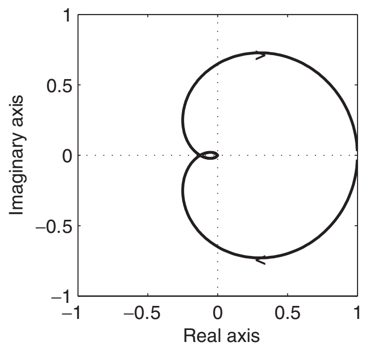


Figure 7.6: Nyquist diagram of plant $G_p = 1/(s+1)^3$. (fignyquist.m)

the lower half of the Nyquist diagram, because it is the plot of the frequency response $G_p(j\omega)$ with $\omega > 0$.

Inserting a proportional controller with the gain K_p in front of the plant will change the diagram such that the new diagram of K_pG_p will be a scaling of the G_p diagram by the factor K_p. If K_p is sufficiently large, corresponding to the ultimate gain K_u in the Ziegler–Nichols tuning method, the curve will pass through the critical point, and the system will be marginally stable. The frequency of oscillation will be the frequency that corresponds to the intersection point on $G_p(j\omega)$.

Stability

The Nyquist criterion relates the stability of the closed-loop system to the open-loop frequency response and open-loop pole locations. The Nyquist criterion is

$$Z = N + P$$

The symbols are explained below:

Z The number of roots of the characteristic equation in the right half-plane, or equivalently, the number of poles of the closed-loop transfer function in the right half-plane. For the closed-loop system to be stable, Z must be zero. Note that Z cannot be negative.

N The number of clockwise encirclements of the critical point by the Nyquist diagram. If the encirclements are in a counterclockwise direction, N is the negative of the number of encirclements. Thus, N may be either positive or negative. If the Nyquist diagram intersects the critical point, the closed-loop system has poles on the $j\omega$-axis.

P The number of poles of the forward path transfer function $G_c(s)G_p(s)$ in the right half-plane. Thus, P cannot be a negative number. Note that the closed-loop system may be stable ($Z = 0$) with the open-loop system unstable ($P > 0$), provided $N = -P$.

In Figure 7.6, the Nyquist diagram does not encircle the critical point. Hence $N = 0$, and since there are no open-loop poles in the right half-plane $P = 0$. Thus $Z = 0$ and the closed-loop system is stable. On the other hand, if $G_c = K_p = 8$, the Nyquist diagram will pass through the critical point, and the closed-loop system will be marginally stable.

Even if the model of a system is stable, the physical system could be unstable because of inaccuracies and disturbances. Therefore, the controller G_c should be designed with sufficient tolerance to variations in the plant dynamics. One stability indicator is the function $\left(1 + G_c G_p\right)$, called the *return difference*, which is the denominator of the closed-loop transfer function. In the Nyquist diagram of $G_c(j\omega)G_p(j\omega)$, the vector $\left[1 + G_c(j\omega)G_p(j\omega)\right]$ is a vector pointing from the critical point to the current point of the Nyquist diagram. Therefore, the minimum return difference is the shortest distance from the Nyquist diagram to the critical point, or

$$\frac{1}{M_s} = \inf\left|1 + G_c(j\omega)G_p(j\omega)\right|, \qquad 0 < \omega < \infty$$

A design criterion is therefore that the Nyquist diagram of $G_c G_p$ be outside a circle with its centre in the critical point and with the radius $1/M_s$. The sensitivity function M_s is the inverse of the shortest distance from the Nyquist diagram to the critical point, and a reasonable value of M_s is in the range 1.3–2 (Åström & Hägglund 1995, p 126), which corresponds to a radius in the range 0.5–0.77.

Limit cycle

A limit cycle is any isolated closed trajectory in the phase plane that is approached asymptotically both from within and without by other trajectories. A motion started on this curve will stay on it forever, circling periodically around the origin. The trajectory must be both closed, indicating periodicity, and isolated, indicating the limiting nature of the cycle. A limit cycle can be stable, unstable, or semi-stable.

Limit cycling is a unique feature of nonlinear systems, and the trajectory in the phase plane must enclose at least one equilibrium point. A limit cycle occurs if the closed-loop denominator is zero:

$$1 + N(A, j\omega)G_p(j\omega) = 0 \tag{7.10}$$

Here, $N(A, j\omega)$ is the frequency response of a nonlinear controller. When the frequency response locus $N(A, j\omega)G_p(j\omega)$ is plotted, any intersection with the critical point will be a solution of Equation (7.10). In general $N(A, j\omega)$ depends on both amplitude and frequency, and therefore a family of curves can be drawn, each curve corresponding to a particular amplitude. The amplitude A_0 and the frequency ω_0 corresponding to an intersection are the amplitude and frequency of the oscillation. If n curves intersect, then the system has n possible limit cycles. The oscillation may be stable or unstable, where, in this context, semi-stable cases are classified as unstable. If Equation (7.10) does not have a solution, then the nonlinear system has no limit cycles.

7.5 Closed-Loop Simulation Examples

To illustrate the effect of various control surfaces, we shall study the Nyquist plot related to four different plants: a third-order plant, a double integrator, a first-order plus deadtime

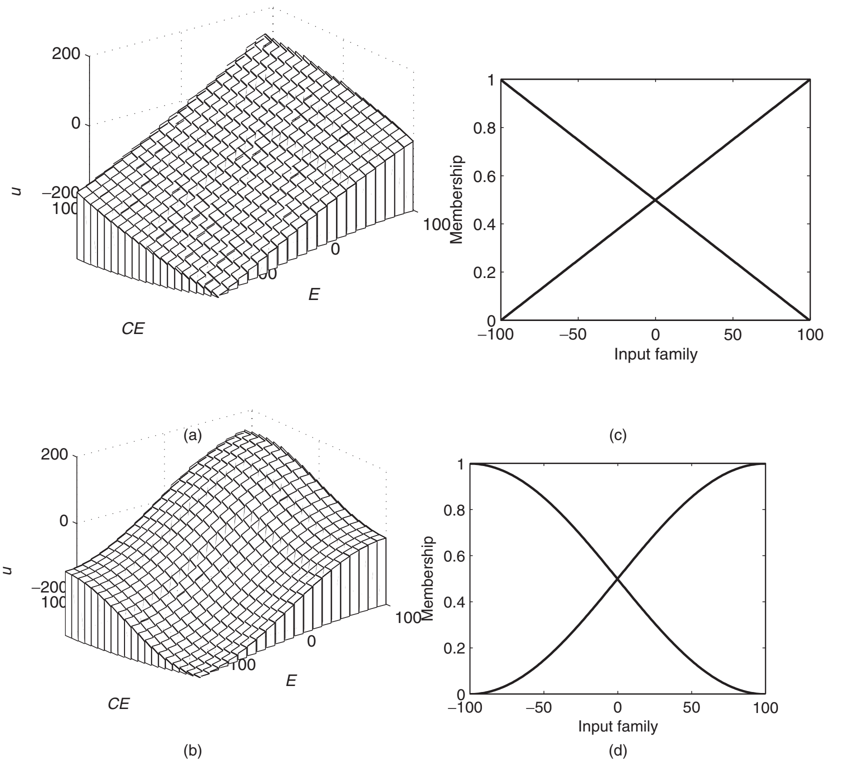


Figure 7.7: Linear surface (a) and saturation surface (b) with the input families that generated them (c and d respectively). (figsurfs.m)

(FODT) plant, and an oscillatory plant. For each plant, we loosely tune a PID controller, convert the gains to the equivalent fuzzy gains, and plot the Nyquist diagram for four different control surfaces.

Chapter 5 introduces the four standard control surfaces, repeated in Figures 7.7–7.8. They are odd functions, that is, $F(-e, -se) = -F(e, se)$ and the Fourier coefficient $a_0 = 0$. Each control surface is inserted in turn in the control loop without changing the tuning. Hand-tuning of a PID controller for each plant provides the settings of the gains for all controllers. The hand-tuning is loose, because the objective is to compare the relative performance of each standard control surface rather than find a controller that provides the best response.

The performance is measured by means of the sensitivity

$$M_s = 2$$

which is equivalent to a minimal distance to the critical point of 0.5. The farther the Nyquist curve is from the 0.5-circle, the more robust is the closed-loop system. In summary, we are only changing the shape of the control surface, not the tuning, in order to make a fair comparison. With four plants and four control surfaces, we set out to make

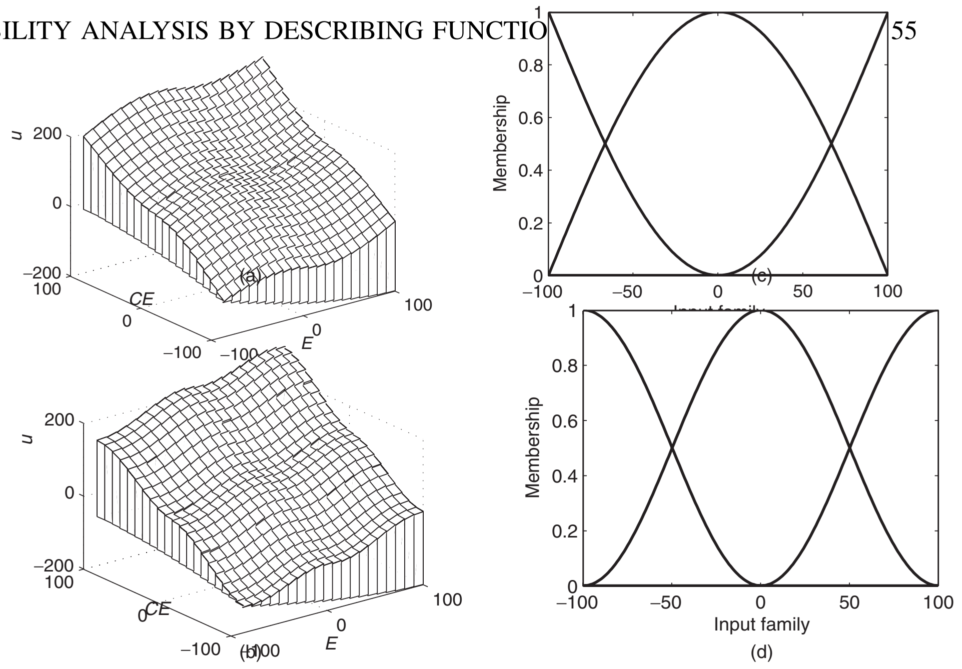


Figure 7.8: Deadzone surface (a) and quantizer surface (b) with the input families that generated them (c and d respectively). (figsurfs.m)

$4 \times 4 = 16$ frequency responses. All describing functions assume a sinusoidal amplitude $A = 1$.

Third-order plant

The third-order plant

$$G_p = \frac{1}{(s+1)^3}$$

is an example of a plant of higher order (greater than two). For the PID gains, we reuse the gains found earlier by Ziegler–Nichols tuning (Chapter 4):

$$K_p = 4.8$$
$$\frac{1}{T_i} = \frac{8}{15}$$
$$T_d = \frac{15}{32}$$

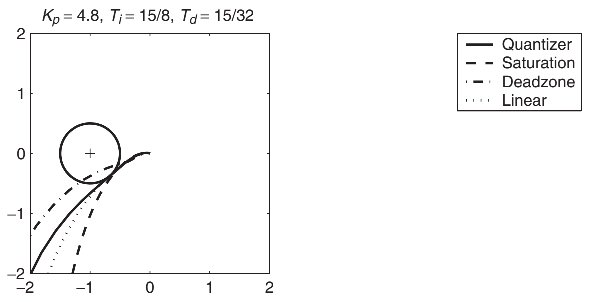


Figure 7.9: Nyquist diagram for the plant $1/(s + 1)^3$ with four FPD+I controllers. (figdf.m)

The equivalent fuzzy gains are

$$GE = 100$$

$$GCE = T_d * GE = 46.9$$

$$GIE = \frac{1}{T_i} * GE = 53.3$$

$$GU = \frac{K_p}{GE} = 0.048$$

Thus, the linear fuzzy PD+I controller acts like a summation, as long as there is no saturation in the input universes. The expression for the describing function is given in Equation (7.5).

The plot in Figure 7.9 shows the Nyquist diagram with the four controllers inserted in the loop. Only a portion of the plot is shown in order to study the behaviour as the Nyquist curve passes by the sensitivity circle. The frequency increases from $\omega = 0.1$ to $\omega = 100$, and the direction of increasing frequency is towards the origin (0, 0). The integrals for the Fourier coefficients are evaluated by means of numerical integration (rectangular integration).

The figure shows that the linear FPD+I controller touches the sensitivity circle tangentially. The deadzone controller cuts through the sensitivity circle, while its opposite, the saturation controller, has the best robustness of all four controllers. The quantizer is less robust than the linear controller. For lower frequencies, the saturation controller provides the most phase margin.

In this case, the saturation controller performs better than the linear controller.

Double integrator plant

The double integrator plant

$$G_p = \frac{1}{s^2}$$

is an example of a plant with integration. Thus a PD controller is sufficient, and by hand-tuning we arrive at the gains

$$K_p = 0.5$$
$$\frac{1}{T_i} = 0$$
$$T_d = 1$$

The equivalent fuzzy gains are

$$GE = 100$$
$$GCE = T_d * GE = 100$$
$$GIE = 0$$
$$GU = \frac{K_p}{GE} = 0.05$$

We use a fuzzy PD controller (without integral action), and the expression for the describing function is given in Equation (7.8).

The plot in Figure 7.10 shows the Nyquist diagram with the four surfaces. The deadzone controller provides the worst robustness, while the other three provide almost the same robustness. At lower frequencies, however, the saturation controller provides the most robustness.

In this case the saturation controller performs better than the linear controller.

Time delay plant

The FODT plant

$$G_p = \frac{e^{-2s}}{s+1}$$

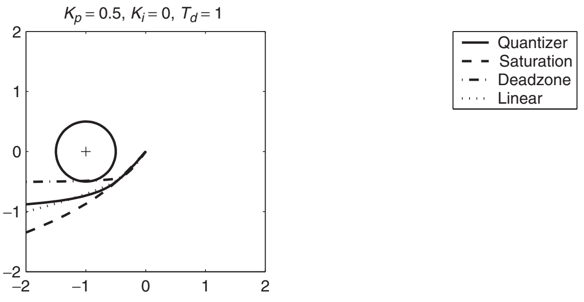


Figure 7.10: Nyquist diagram for the plant $1/s^2$ with four FPD controllers. (figdf.m)

is an example of a plant with a time delay. In order to simulate the system, the time delay is approximated by a polynomial, the first-order Pade approximation (e.g. the function `pade` in Matlab),

$$e^{-2s} \approx \frac{-s+1}{s+1}$$

The plant requires integral action in order to remove steady state error, and by hand-tuning,

$$K_p = 0.5$$
$$\frac{1}{T_i} = 1$$
$$T_d = 0.25$$

The equivalent fuzzy gains are

$$GE = 100$$
$$GCE = T_d * GE = 25$$
$$GIE = \frac{1}{T_i} * GE = 100$$
$$GU = \frac{K_p}{GE} = 0.05$$

The expression for the describing function is given in Equation (7.8).

The plot in Figure 7.11 shows the Nyquist diagram with the four controllers. The figure shows that the deadzone controller again provides the worst robustness, while the saturation controller provides the best robustness, but the differences are minor.

In this case, the saturation controller again performs better than the linear controller.

Oscillatory modes

The plant

$$G_p = \frac{25}{(s+1)(s^2+25)}$$

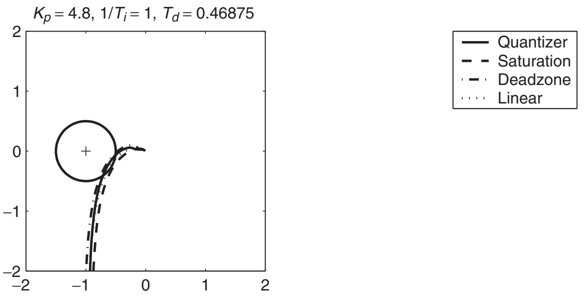


Figure 7.11: Nyquist diagram for the plant $e^{-2s}/(s+1)$ with four FPD+I controllers. (figdf.m)

is an example of a plant with two undamped poles on the imaginary axis ($\pm j5$) causing an oscillatory response. Apart from these two poles, the system is simply a first-order system. It can be controlled by a PI controller with the parameters (Åström and Hägglund 1995)

$$K_p = -0.25$$
$$\frac{1}{T_i} = -1$$

The equivalent fuzzy gains are

$$\begin{aligned} GE &= 100 \\ GCE &= 0 \\ GIE &= \frac{1}{T_i} * GE = -100 \\ GU &= \frac{K_p}{GE} = -0.0025 \end{aligned}$$

The expression for the describing function is given in Equation (7.8) with $a_1 = 0$.

The plot in Figure 7.12 shows the Nyquist diagram with the four controllers. The figure shows that in this case the deadzone controller provides the best robustness, whereas the saturation controller provides the worst robustness.

The deadzone controller does not dampen the oscillations, however, and it causes limit cycling (see the following text).

Limit cycle

The describing function depends on amplitude, but the preceding diagrams were all based on the amplitude $A = 1$. The diagrams change with a change in the amplitude of the input sinusoid. For example, consider the third-order plant

$$G_p = \frac{1}{(s+1)^3}$$

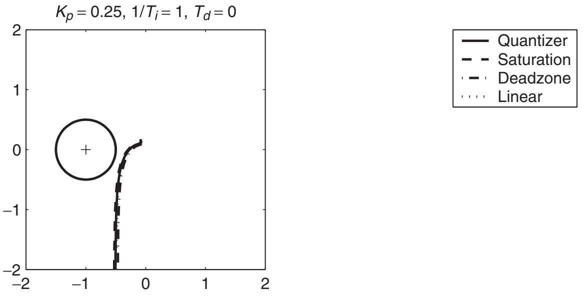


Figure 7.12: Nyquist diagram for the plant $25/(s+1)(s^2+25)$ with four fuzzy PI controllers. (figdf.m)

with the settings, as before,

$$K_p = 0.5$$
$$\frac{1}{T_i} = 0$$
$$T_d = 1$$

and the equivalent fuzzy gains

$$GE = 100$$
$$GCE = T_d * GE = 100$$
$$GIE = 0$$
$$GU = \frac{K_p}{GE} = 0.05$$

However, now the amplitude is chosen to be

$$A = 0.45$$

Figure 7.13 shows that the Nyquist diagram has moved closer to the critical point. In fact, the Nyquist curve corresponding to the deadzone surface passes through the critical point, indicating a limit cycle, and therefore the plot predicts a limit cycle with an amplitude of 0.45. The frequency corresponding to the frequency at the intersection of the critical point is $\omega = 1.04$ rad/s or 0.17 Hz (obtained from the run that produced the plot). The remaining three surfaces are more robust.

Figure 7.14 shows the step response with the deadzone controller, and indeed, a limit cycle is present. Closer inspection of the plot reveals that its amplitude is 0.46 (0.45 predicted) and its frequency is 0.17 Hz (0.17 Hz predicted).

The accuracy depends on whether the input to the nonlinearity is sinusoidal, or in other words, whether the filtering hypothesis is satisfied. The reference is constant, except for the initial step, and therefore the error signal has the shape of the controlled output y (Figure 7.14, upper left), which in fact looks sinusoidal by inspection.

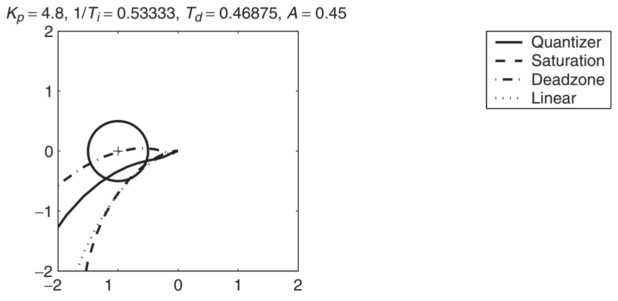


Figure 7.13: Nyquist diagram for the plant $1/(s + 1)^3$ with four FPD+I controllers and amplitude $A = 0.45$. (figdf.m)

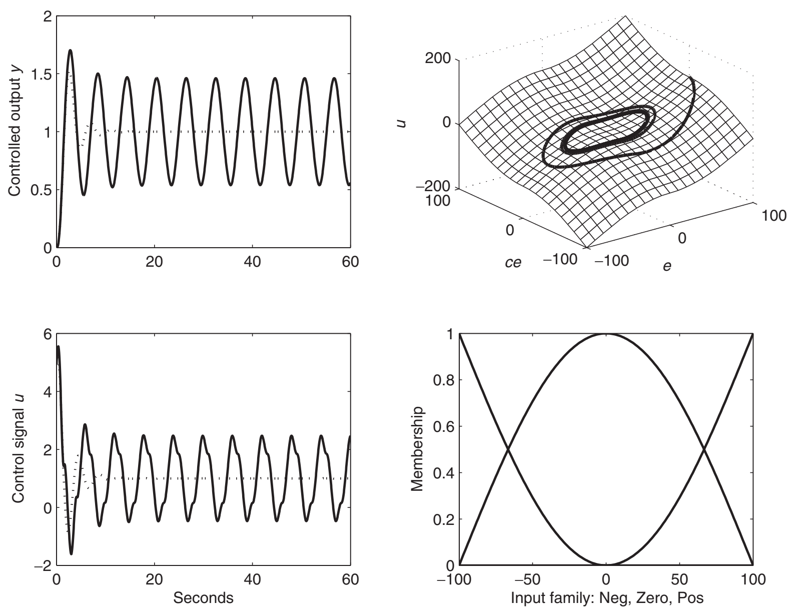


Figure 7.14: Step response (left column) of the plant $1/(s+1)^3$ with a deadzone FPD controller (right column). The dotted line (left) is the response with the equivalent PD controller. (figlimcyc.m)

The example shows that it was possible to predict a limit cycle, and that the deadzone surface in this case was less robust than the three other standard control surfaces. The predicted amplitude and frequency were quite accurate.

7.6 Analytical Derivation of the Describing Function*

It is possible to analytically derive the describing functions for the nonlinear standard control surfaces. Under the given conditions, formulated as design rules earlier, the Fourier integrals can be evaluated. We can therefore assess the accuracy of the numerically calculated frequency responses.

Assumptions

The following checklist summarizes the five conditions, formulated as design choices, under which the four surfaces are built:

1. Use triangular premise sets that cross at membership $\mu = 0.5$.
2. Build a rule base containing all possible $\wedge$-combinations of the premise terms.
3. Use multiplication $(*)$ for the $\wedge$-connective.

*Can be skipped in a first reading.

4. Use conclusion singletons, positioned at the sum of the peak positions of the premise sets.

5. Use sum-accumulation and *COGS* defuzzification.

For the nonlinear rule bases, item 1 is relaxed such that the premise sets are smooth versions of triangular sets that cross at $\mu = 0.5$. Under these assumptions, we were able to analytically simplify the inference procedure, and this enables the analytical derivation of the describing functions.

Saturation surface

The saturation surface (Figure 7.7) is based on the rule base with four rules:

If *error* is Neg and *change in error* is Neg then *control* is NB

If *error* is Neg and *change in error* is Pos then *control* is Zero

If *error* is Pos and *change in error* is Neg then *control* is Zero

If *error* is Pos and *change in error* is Pos then *control* is PB

and it was shown (Chapter 3) that the inferred controller output can be written directly as

$$u = (P_E + P_{CE} - 1) * s_{PB} \tag{7.11}$$

The symbol P_E is shorthand notation for $\mu_{Pos}(GE * e)$, the membership of error signal $E = GE * e$ of fuzzy set Pos; P_{CE} is shorthand notation for $\mu_{Pos}(GCE * se)$, the membership of the change in error signal $CE = GCE * se$ of fuzzy set Pos; and s_{PB} is shorthand notation for $\langle s_{PB}, 1\rangle$, the singleton corresponding to the conclusion PB. The value of s_{PB} is 200 when standard universes are used. The expression is valid even for nonlinear membership functions μ_{Pos} and μ_{Neg}, as long as $\mu_{Pos}(x) + \mu_{Neg}(x) = 1$. Therefore, we can evaluate the Fourier integrals as a sum of integrals.

Take the in-phase coefficient first:

$$\begin{aligned}
b_1 &= \frac{1}{\pi}\int_0^{2\pi} F(e, se)\sin(\omega t)\,\mathrm{d}(\omega t) \\
&= \frac{1}{\pi}\int_0^{2\pi} (P_E + P_{CE} - 1) * s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t) \\
&= \frac{1}{\pi}\int_0^{2\pi} \left(P_E - \frac{1}{2}\right) * s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t) \\
&\quad + \frac{1}{\pi}\int_0^{2\pi} \left(P_{CE} - \frac{1}{2}\right) * s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t)
\end{aligned}$$

We notice in passing that the integral related to P_{CE} will vanish, owing to its cosine content. The membership function μ_{Pos} is defined as a smooth trapezoidal membership function (Chapter 2):

$$\mu_{STrapezoid}(x; a, b, c, d) = \left\{ \begin{array}{ll} 0 & , x \le a \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-b}{b-a}\pi\right) & , a \le x \le b \\ 1 & , b \le x \le c \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-c}{d-c}\pi\right) & , c \le x \le d \\ 0 & , d \le x \end{array} \right\}, x \in \mathbb{R}$$

with the breakpoints $a = -100$, $b = 100$, $c = 100$, and $d = 100$. Using the standard universe $\mathcal{U} = [-100, 100]$, the fuzzy set Pos is thus defined by the membership function

$$\begin{aligned} \mu_{Pos}(x) &= \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-100}{100-(-100)}\pi\right) \\ &= \frac{1}{2} + \frac{1}{2}\sin\left(\pi\frac{x}{200}\right) \end{aligned}$$

which is valid for the whole universe $-100 \le x \le 100$. Observe that the describing function for a sinusoidal (*harmonic*) nonlinearity can be looked up in a table.

The membership function μ_{Neg} is defined as a smooth trapezoidal membership function, with the breakpoints $a = -100$, $b = -100$, $c = -100$, and $d = 100$. Thus,

$$\begin{aligned} \mu_{Neg}(x) &= \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x+100}{100-(-100)}\pi\right) \\ &= \frac{1}{2} - \frac{1}{2}\sin\left(\pi\frac{x}{200}\right) \end{aligned}$$

Indeed, the condition $\mu_{Pos}(x) + \mu_{Neg}(x) = 1$ is satisfied, since the sinusoidal terms cancel each other after summation.

We inject a sinusoidal input signal $e = A\sin(\omega t)$, $0 \le \omega t \le 2\pi$. Since $E = GE * e$, and $CE = GCE * se$, their firing strengths are $P_E = 1/2 + 1/2\sin(\pi * GE * e/200)$ and $P_{CE} = 1/2 + 1/2\sin(\pi * GCE * se/200)$. Thus,

$$\begin{aligned} b_1 &= \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\sin\left(\frac{\pi * GE * e}{200}\right) * s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t) \\ &+ \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\sin\left(\frac{\pi * GCE * se}{200}\right) * s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t) \\ &= s_{PB} * GU * \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\sin\left(\frac{\pi * GE * e}{200}\right) * \sin(\omega t)\,\mathrm{d}(\omega t) \\ &= s_{PB} * GU * \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\sin\left(\frac{\pi * GE}{200}A\sin(\omega t)\right) * \sin(\omega t)\,\mathrm{d}(\omega t) \end{aligned}$$

The integral occurs in the evaluation of the describing function for the harmonic nonlinearity (Atherton 1975), and we directly get

$$b_1 = s_{PB} * GU * \mathrm{J}_1\left(\frac{\pi}{2}\frac{GE}{100}A\right), \qquad -100 \le GE * A \le 100$$

Here J_1 is the so-called Bessel function of order 1, defined as the solution to the integral

$$\begin{aligned}\mathrm{J}_1(x) &= \frac{1}{2\pi}\int_0^{2\pi} \cos(\omega t - x\sin(\omega t))\,\mathrm{d}(\omega t)\\ &= \frac{1}{2\pi}\int_0^{2\pi} [\sin(\omega t)\sin(x\sin(\omega t)) + \cos(\omega t)\cos(x\sin(\omega t))]\,d(\omega t)\end{aligned}$$

Its power series expansion is (see also the Matlab function `besselj`),

$$\mathrm{J}_1(x) = \frac{x}{2} - \frac{x^3}{2^3 * 1!2!} + \frac{x^5}{2^5 * 2!3!} - \frac{x^7}{2^7 * 3!4!} + \frac{x^9}{2^9 * 4!5!} - \cdots$$

We have only considered amplitudes within the operating region $-100 \leq GE * A \leq 100$, that is, we assume there is no saturation in the input universe. It is possible to include larger amplitudes also (compare the introductory example of an ideal saturation), but it will complicate the expressions.

For the quadrature coefficient, we get

$$\begin{aligned}a_1 &= \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\sin\left(\frac{\pi * GE * e}{200}\right) * s_{PB} * GU * \cos(\omega t)\,\mathrm{d}(\omega t)\\ &\quad + \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\sin\left(\frac{\pi * GCE * se}{200}\right) * s_{PB} * GU * \cos(\omega t)\,\mathrm{d}(\omega t)\\ &= s_{PB} * GU * \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\sin\left(\frac{\pi * GCE * se}{200}\right) * \cos(\omega t)\,\mathrm{d}(\omega t)\\ &= s_{PB} * GU * \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\sin\left(\frac{\pi * GCE}{200}\omega A\cos(\omega t)\right) * \cos(\omega t)\,\mathrm{d}(\omega t)\end{aligned}$$

The integral corresponds to injecting a cosine with a frequency dependent amplitude ωA into the harmonic nonlinearity. The solution is

$$a_1 = s_{PB} * GU * \mathrm{J}_1\left(\frac{\pi}{2}\frac{GCE}{100}\omega A\right), \qquad -100 \leq GCE * \omega A \leq 100$$

Again, we have excluded, for simplicity, amplitudes where the signal saturates in the input universe.

We can now write the describing function in the form of Equation (7.4):

$$\begin{aligned}N(A,s) &= \frac{b_1}{A} + \frac{a_1}{A}\frac{1}{\omega}s\\ &= s_{PB} * GU * \frac{1}{A} * \mathrm{J}_1\left(\frac{\pi}{2}\frac{GE}{100}A\right)\\ &\quad + s_{PB} * GU * \frac{1}{\omega A} * \mathrm{J}_1\left(\frac{\pi}{2}\frac{GCE}{100}\omega A\right) * s\end{aligned}$$

valid for

$$-100 \leq GE * A \leq 100 \text{ and} - 100 \leq GCE * \omega A \leq 100$$

with $\mathrm{J}_1(x)$ being the Bessel function of order 1 given above. The frequency response appears after the substitution $s = j\omega$:

$$\begin{aligned} N(A, j\omega) &= \frac{b_1}{A} + j\frac{a_1}{A} \\ &= \frac{1}{A} * s_{PB} * GU * \left[\mathrm{J}_1\left(\frac{\pi}{2}\frac{GE}{100}A\right) + j * \mathrm{J}_1\left(\frac{\pi}{2}\frac{GCE}{100}\omega A\right)\right] \end{aligned}$$

A comparison with the frequency response in Figure 7.5, with $s_{PB} = 200$, $GU = 0.1$, $GE = 100$, $GCE = 100$, $A = 1$, and $0.1 \le \omega \le 1$, showed negligible deviations in the order of 10^{-15}.

Deadzone surface

The deadzone surface (Figure 7.8) is based on the rule base with nine rules, and it was shown (Chapter 3) that under the above assumptions, the inferred controller output can be written directly as

$$u = \frac{1}{2}\left(P_E - N_E + P_{CE} - N_{CE}\right) s_{PB}$$

Singleton $s_{PB} = 200$ when standard universes are used. The expression is valid even for nonlinear fuzzy membership functions μ_{Pos}, μ_{Zero} and μ_{Neg}, as long as $\mu_{Pos}(x) + \mu_{Zero}(x) + \mu_{Neg}(x) = 1$.

The membership function μ_{Pos} is a smooth trapezoidal membership function with the breakpoints $a = 0$, $b = 200$, $c = 200$, and $d = 200$, and

$$\mu_{Pos}(x) = \begin{array}{ll} \left\{ \begin{array}{l} 1 + \cos\left(\frac{x-200}{200-0}\pi\right) \\ = 1 - \cos\left(\frac{x}{100}\frac{\pi}{2}\right) \end{array} \right\} & \text{for } 0 \le x \le 100 \\ \quad 0 & \text{for } -100 \le x < 0 \end{array}$$

The membership function μ_{Zero} is a smooth trapezoidal membership function with the breakpoints $a = -200$, $b = 0$, $c = 0$, and $d = 200$, and

$$\mu_{Zero}(x) = \cos\left(\frac{x}{100}\frac{\pi}{2}\right) \quad \text{for } -100 \le x \le 100$$

The membership function μ_{Neg} is a smooth trapezoidal membership function with the breakpoints $a = -200$, $b = -200$, $c = -200$, and $d = 0$.

$$\mu_{Neg}(x) = \begin{array}{ll} \left\{ \begin{array}{l} 1 + \cos\left(\frac{x-(-200)}{0-(-200)}\frac{\pi}{2}\right) \\ = 1 - \cos\left(\frac{x}{100}\frac{\pi}{2}\right) \end{array} \right\} & \text{for } -100 \le x \le 0 \\ \quad 0 & \text{for } 0 < x \le 100 \end{array}$$

It is clear that indeed $\mu_{Pos}(x) + \mu_{Zero}(x) + \mu_{Neg}(x) = 1$.

Now we can proceed with the Fourier coefficients. Take the in-phase coefficient first:

$$\begin{aligned}
b_1 &= \frac{1}{\pi}\int_0^{2\pi} F(e, se)\sin(\omega t)\,\mathrm{d}(\omega t)\\
&= \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\,(P_E - N_E + P_{CE} - N_{CE})\,s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t)\\
&= \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}(P_E - N_E)s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t)
\end{aligned}$$

The problem can be restated as that of finding the Fourier expansion for a nonlinearity of the form $\mathrm{sgn}(x)\left[1 - \cos\left(\frac{x}{100}\frac{\pi}{2}\right)\right]$, where sgn is the signum function, which is -1 if x is negative and 1 otherwise. This is not straightforward, and therefore we resort to the power series expansion. Owing to symmetry, we can shrink the integration limit to $\pi/2$, and

$$\begin{aligned}
b_1 &= \frac{1}{\pi}4\int_0^{\pi/2} \frac{1}{2}P_E * s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t)\\
&= s_{PB} * GU * \frac{2}{\pi}\int_0^{\pi/2}\left(1 - \cos\left(\frac{GE * A\sin(\omega t)}{100}\frac{\pi}{2}\right)\right) * \sin(\omega t)\,\mathrm{d}(\omega t)
\end{aligned}$$

The cosine power series expansion is

$$\cos\varphi \approx 1 - \frac{\varphi^2}{2!} + \frac{\varphi^4}{4!} - \frac{\varphi^6}{6!} + \frac{\varphi^8}{8!} - \cdots \tag{7.12}$$

Thus the integral becomes an alternating series of powers of sinusoids of equal frequency, which makes integration feasible, although it is an approximation. The error is arbitrarily small, however; it depends on the number of terms included in the series expansion, and the error increases as the magnitude of φ increases. For the eighth-degree polynomial shown, and the worst case where the amplitude of the sinusoidal input is maximum ($\varphi = \pm\pi/2$), the error is $2.5 * 10^{-5}$. With $\varphi = GE * A * \pi/200 * \sin(\omega t) = A_b\sin(\omega t)$, the coefficient is approximately

$$\begin{aligned}
b_1 &\approx s_{PB} * GU * \frac{2}{\pi}\\
&\quad * \int_0^{\pi/2}\left(1 - \left(1 - \frac{\varphi^2}{2!} + \frac{\varphi^4}{4!} - \frac{\varphi^6}{6!} + \frac{\varphi^8}{8!}\right)\right)\sin(\omega t)\,\mathrm{d}(\omega t)\\
&= s_{PB} * GU * \frac{2}{\pi}\int_0^{\pi/2}\left(\frac{\varphi^2}{2!} + \frac{\varphi^4}{4!} - \frac{\varphi^6}{6!} + \frac{\varphi^8}{8!}\right)\sin(\omega t)\,\mathrm{d}(\omega t)
\end{aligned}$$

Insertion of the value $\varphi = A_b\sin(\omega t)$ and evaluation is a complex task and prone to error, and we shall resort to a symbolic math package (Maple, trademark of Waterloo Maple Inc). Thus,

$$b_1 \approx s_{PB} * GU * \frac{2}{\pi}\left(\frac{1}{3}A_b^2 - \frac{1}{45}A_b^4 + \frac{1}{1575}A_b^6 - \frac{1}{99\,225}A_b^8\right),$$

with

$$A_b = \frac{GE * A}{100}\frac{\pi}{2}$$

and

$$-100 \leq GE * A \leq 100$$

We have added the last constraint to avoid considering saturation in the input universe, for simplicity.

The quadrature coefficient is

$$\begin{aligned}
a_1 &= \frac{2}{\pi}\int_0^{2\pi} F(e, se)\cos(\omega t)\,\mathrm{d}(\omega t) \\
&= \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\,(P_E - N_E + P_{CE} - N_{CE}) * s_{PB} * GU * \cos(\omega t)\,\mathrm{d}(\omega t) \\
&= \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}(P_{CE} - N_{CE}) * s_{PB} * GU * \cos(\omega t)\,\mathrm{d}(\omega t) \\
&= s_{PB} * GU * \frac{2}{\pi}\int_0^{\pi/2} P_{CE}\cos(\omega t)\,\mathrm{d}(\omega t)
\end{aligned}$$

The membership function P_{CE} is the same as P_E, but the input to the nonlinearity is now $\varphi = GCE * se = GCE * \pi/200 * \omega A\cos(\omega t) = A_a\cos(\omega t)$. Therefore, we can approximate

$$a_1 \approx s_{PB} * GU * \frac{2}{\pi}\int_0^{\pi/2}\left(\frac{\varphi^2}{2!} + \frac{\varphi^4}{4!} - \frac{\varphi^6}{6!} + \frac{\varphi^8}{8!}\right)\cos(\omega t)\,\mathrm{d}(\omega t)$$

Using the symbolic math package, with $\varphi = A_a\cos(\omega t)$, we have

$$a_1 \approx s_{PB} * GU * \frac{1}{\pi}\left(\frac{1}{3}A_a^2 - \frac{1}{45}A_a^4 + \frac{1}{1575}A_a^6 - \frac{1}{99\,225}A_a^8\right),$$

with

$$A_a = \frac{GCE * \omega A}{100}\frac{\pi}{2}$$

and

$$-100 \leq GCE * \omega A \leq 100$$

We have added the last constraint to avoid considering saturation in the input universe, for simplicity.

Summarizing, we can write the describing function in the form of Equation (7.4):

$$\begin{aligned}
N(A, s) &= \frac{b_1}{A} + \frac{a_1}{A}\frac{1}{\omega}s \qquad (7.13) \\
&= \frac{1}{A} * s_{PB} * GU * \frac{2}{\pi} * S(A_b) \\
&\quad + \frac{1}{\omega A} * s_{PB} * GU * \frac{2}{\pi} * S(A_a) * s
\end{aligned}$$

with

$$\begin{aligned}
S(z) &= \frac{z^2}{3} - \frac{z^4}{45} + \frac{z^6}{1575} - \frac{z^8}{99\,225} \\
A_b &= \frac{GE * A}{100}\frac{\pi}{2} \\
A_a &= \frac{GCE * \omega A}{100}\frac{\pi}{2}
\end{aligned}$$

The describing function is valid for

$$-100 \leq GE * A \leq 100 \text{ and} - 100 \leq GCE * \omega A \leq 100$$

The frequency response appears after the substitution $s = j\omega$:

$$\begin{aligned}
N(A, j\omega) &= \frac{b_1}{A} + j\frac{a_1}{A} \\
&= \frac{1}{A} * s_{PB} * GU * \frac{2}{\pi} * (S(A_b) + jS(A_a))
\end{aligned}$$

A comparison with the frequency response in Figure 7.5, with $s_{PB} = 200$, $GU = 0.1$, $GE = 100$, $GCE = 100$, $A = 1$, and $0.1 \leq \omega \leq 1$, showed a maximum deviation of 0.0013 %, regarding both magnitude and phase.

Quantizer surface

The quantizer surface (Figure 7.8) is based on the rule base with nine rules, and again we exploit the fact that the inferred controller output can be written directly as

$$u = \frac{1}{2}(P_E - N_E + P_{CE} - N_{CE})\, s_{PB}$$

Singleton $s_{PB} = 200$ when standard universes are used. The expression is valid even for nonlinear fuzzy membership functions μ_{Pos}, μ_{Zero} and μ_{Neg}, as long as $\mu_{Pos}(x) + \mu_{Zero}(x) + \mu_{Neg}(x) = 1$.

The membership function μ_{Pos} is a smooth trapezoidal membership function with the breakpoints $a = 0$, $b = 100$, $c = 100$, and $d = 100$:

$$\mu_{Pos}(x) = \begin{cases} \left\{ \begin{array}{l} \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-100}{100-0}\pi\right) \\ = \frac{1}{2} - \frac{1}{2}\cos\left(\frac{x}{100}\pi\right) \\ = \sin^2\left(\frac{\pi}{2}\frac{x}{100}\right) \end{array} \right\} & \text{for } 0 \leq x \leq 100 \\ 0 & \text{for } -100 \leq x < 0 \end{cases}$$

The double angle relationship $\cos 2\alpha = 1 - 2\sin^2\alpha$ was applied in the above calculation. The membership function μ_{Zero} is a smooth trapezoidal membership function with the breakpoints $a = -100$, $b = 0$, $c = 0$, and $d = 100$,

$$\mu_{Zero}(x) = \left\{ \begin{array}{l} \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x}{100}\pi\right) \\ = \cos^2\left(\frac{\pi}{2}\frac{x}{100}\right) \end{array} \right\} \quad \text{for } -100 \leq x \leq 100$$

Another double angle relationship $\cos 2\alpha = 2\cos^2\alpha - 1$ was applied above. The membership function μ_{Neg} is a smooth trapezoidal membership function with the breakpoints $a = -100$, $b = -100$, $c = -100$, and $d = 0$.

$$\mu_{Neg}(x) = \begin{cases} \left\{ \begin{array}{l} \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-(-100)}{0-(-100)}\pi\right) \\ = \frac{1}{2} - \frac{1}{2}\cos\left(\frac{x}{100}\pi\right) \\ = \sin^2\left(\frac{\pi}{2}\frac{x}{100}\right) \end{array} \right\} & \text{for } -100 \le x \le 0 \\ 0 & \text{for } 0 < x \le 100 \end{cases}$$

By the Pythagorean relation $\sin^2\alpha + \cos^2\alpha = 1$, it is clear that indeed $\mu_{Pos}(x) + \mu_{Zero}(x) + \mu_{Neg}(x) = 1$.

Now we can proceed with the Fourier coefficients. Take the in-phase coefficient first:

$$\begin{aligned} b_1 &= \frac{1}{\pi}\int_0^{2\pi} F(e, se)\sin(\omega t)\,\mathrm{d}(\omega t) \\ &= \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\left(P_E - N_E + P_{CE} - N_{CE}\right) s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t) \\ &= \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}(P_E - N_E) s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t) \end{aligned}$$

The problem can be restated as that of finding the Fourier expansion for a nonlinearity of the form $\operatorname{sgn}(x)\left[\frac{1}{2} - \frac{1}{2}\cos\left(\frac{x}{100}\pi\right)\right]$, where sgn is the signum function. This is not straightforward, and therefore we resort to power series expansion. Owing to symmetry, we can shrink the integration limit to $\pi/2$, and

$$\begin{aligned} b_1 &= \frac{1}{\pi}4\int_0^{\pi/2} \frac{1}{2} P_E * s_{PB} * GU * \sin(\omega t)\,\mathrm{d}(\omega t) \\ &= s_{PB} * GU * \frac{2}{\pi}\int_0^{\pi/2} \left(\frac{1}{2} - \frac{1}{2}\cos\left(\frac{GE * A\sin(\omega t)}{100}\pi\right)\right) * \sin(\omega t)\,\mathrm{d}(\omega t) \end{aligned}$$

Using the cosine power series expansion again (Equation (7.12)) and with $\varphi = GE * A\sin(\omega t)\pi/100 = A_b\sin(\omega t)$, the coefficient is approximately

$$\begin{aligned} b_1 &\approx s_{PB} * GU * \frac{2}{\pi} \\ &\quad * \int_0^{\pi/2} \left(\frac{1}{2} - \frac{1}{2}\left(1 - \frac{\varphi^2}{2!} + \frac{\varphi^4}{4!} - \frac{\varphi^6}{6!} + \frac{\varphi^8}{8!}\right)\right)\sin(\omega t)\,\mathrm{d}(\omega t) \\ &= s_{PB} * GU * \frac{1}{\pi}\int_0^{\pi/2} \left(\frac{\varphi^2}{2!} - \frac{\varphi^4}{4!} + \frac{\varphi^6}{6!} - \frac{\varphi^8}{8!}\right)\sin(\omega t)\,\mathrm{d}(\omega t) \end{aligned}$$

We have evaluated the integral earlier, and get

$$b_1 \approx s_{PB} * GU * \frac{1}{\pi}\left(\frac{1}{3}A_b^2 - \frac{1}{45}A_b^4 + \frac{1}{1575}A_b^6 - \frac{1}{99\,225}A_b^8\right),$$

with

$$A_b = \frac{GE * A}{100}\pi$$

and

$$-100 \leq GE * A \leq 100$$

We have added the last constraint to avoid considering saturation in the input universe, for simplicity.

The quadrature coefficient is

$$\begin{aligned}
a_1 &= \frac{1}{\pi}\int_0^{2\pi} F(e, se)\cos(\omega t)\,\mathrm{d}(\omega t) \\
&= \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}\,(P_E - N_E + P_{CE} - N_{CE}) * s_{PB} * GU * \cos(\omega t)\,\mathrm{d}(\omega t) \\
&= \frac{1}{\pi}\int_0^{2\pi} \frac{1}{2}(P_{CE} - N_{CE}) * s_{PB} * GU * \cos(\omega t)\,\mathrm{d}(\omega t) \\
&= s_{PB} * GU * \frac{2}{\pi}\int_0^{\pi/2} P_{CE}\cos(\omega t)\,\mathrm{d}(\omega t)
\end{aligned}$$

The membership functions P_{CE} is the same as P_E, but the input to the nonlinearity is now $\varphi = GCE * \omega A\cos(\omega t)\pi/100 = A_a\cos(\omega t)$. Therefore, we can approximate

$$a_1 \approx s_{PB} * GU * \frac{1}{\pi}\int_0^{\pi/2}\left(\frac{\varphi^2}{2!} - \frac{\varphi^4}{4!} + \frac{\varphi^6}{6!} - \frac{\varphi^8}{8!}\right)\cos(\omega t)\,\mathrm{d}(\omega t)$$

with $x = \omega A_a\cos(\omega t)$. Using the symbolic math package,

$$a_1 \approx s_{PB} * GU * \frac{1}{\pi}\left(\frac{1}{3}A_a^2 - \frac{1}{45}A_a^4 + \frac{1}{1575}A_a^6 - \frac{1}{99\,225}A_a^8\right),$$

with

$$A_a = \frac{GCE * \omega A}{100}\pi$$

and

$$-100 \leq GCE * \omega A \leq 100$$

We have added the last constraint to avoid considering saturation in the input universe, for simplicity.

Summarizing, the describing function is

$$\begin{aligned}
N(A, s) &= \frac{b_1}{A} + \frac{a_1}{A}\frac{1}{\omega}s \qquad (7.14)\\
&= \frac{1}{A} * s_{PB} * GU * \frac{1}{\pi} * S(A_b) \\
&\quad + \frac{1}{\omega A} * s_{PB} * GU * \frac{1}{\pi} * S(A_a) * s
\end{aligned}$$

with

$$\begin{aligned} S(z) &= \frac{z^2}{3} - \frac{z^4}{45} + \frac{z^6}{1575} - \frac{z^8}{99\,225} \\ A_b &= \frac{GE * A}{100}\pi \\ A_a &= \frac{GCE * \omega A}{100}\pi \end{aligned}$$

The describing function is valid for

$$-100 \leq GE * A \leq 100 \text{ and} - 100 \leq GCE * \omega A \leq 100$$

The frequency response appears after the substitution $s = j\omega$,

$$\begin{aligned} N(A, j\omega) &= \frac{b_1}{A} + j\frac{a_1}{A} \\ &= \frac{1}{A} * s_{PB} * GU * \frac{1}{\pi} * (S(A_b) + jS(A_a)) \end{aligned}$$

A comparison with the frequency response in Figure 7.5, with $s_{PB} = 200$, $GU = 0.1$, $GE = 100$, $GCE = 100$, $A = 1$, and $0.1 \leq \omega \leq 1$, showed a maximum deviation of 0.54 % for both magnitude and phase.

Comparing the describing function of the quantizer with that of the deadzone (see Equations (7.13) and (7.14)), we find the expressions are identical. But the constants A_a, A_b for the quantizer are larger than the ones for the deadzone by a factor of two.

7.7 Summary

While stability of a linear system rests on a single concept, namely, the eigenvalues, the stability of a nonlinear system is more complex. The describing function of a nonlinearity is a simplification, a linear approximation to a transfer function. It enables a Nyquist analysis, which is convenient, because one single plot shows whether the closed-loop system is stable, and it shows the stability margin. It was thus possible to assess the relative robustness of the standard control surfaces: the linear, the deadzone, the saturation, and the quantization surfaces.

Since it is an approximation, the results may be inaccurate in certain cases. The applicability of the describing functions depends mainly on whether the input to the nonlinearity is really sinusoidal, during closed loop operation; this is *not* the case if, for instance, the reference varies in a non-sinusoidal manner. The results in the chapter are valid for symmetric, odd functions only. Non-symmetric control surfaces, for instance, one generated by an SOC mechanism, result in a non-zero average term in the Fourier series. This is a case for future research.

The describing functions are a step towards an answer to the more general question, What kind of control surface suits which kind of plant?

7.8 Notes and References

Aracil and Gordillo (2003) show in an article how describing functions may be applied to a saturation type of FPD controller, and Gordillo *et al.* (1997) examine the same relative to a quantizer type of controller. The article by Cuesta *et al.* (1999) is more rigorous.

For an easy introduction to describing functions, see the book by Slotine and Li (1991). For an impressive in-depth exposition, see Atherton (1975, 1982). Another option is the book by Gelb and Vander Velde (1968) available on the World Wide Web. Treatments of dynamic nonlinearities are sparse, but Šiljak (1969, p 126) mentions functions of the form $F(x, \dot{x})$, and Gelb and Vander Velde show the proportional plus derivative form of the describing function (1968, Ch. 2, Eq. 2.2–30).

Kickert and Mamdani (1978) attempted to apply describing functions on a fuzzy controller seen as a multi-level relay.

8

Simulation Study: Cart–Ball Balancer

The simulation study in this chapter concerns a controller for balancing a steel ball on a cart. The ball balancer, or cart–ball system is a demonstration of the basic concepts of a nonlinear, multivariable, and non-minimum phase control. It is an inverted pendulum problem, which is a widely used benchmark problem.

In this chapter, a linear controller stabilizes the ball balancer. With certain design choices, a fuzzy controller is equivalent to a summation, and thus it can replace the linear controller. The design approach makes it somewhat easier to design a fuzzy controller.

8.1 Laboratory Rig

Figure 8.1 shows the cart–ball system with the cart, and an arc made of two parallel pipes on which a steel ball rolls. The cart moves on a pair of tracks horizontally mounted on a heavy support. The control objective is to balance the ball on the top of the arc and at the same time place the cart in a desired position.

The controller design will be based on a linearized model of the system,

$$\begin{aligned}\dot{\mathbf{x}} &= \boldsymbol{A}\mathbf{x} + \boldsymbol{B}\mathbf{u} \\ \mathbf{y} &= \boldsymbol{C}\mathbf{x}\end{aligned} \tag{8.1}$$

The model is a linear state-space model, where $\mathbf{x}$ is a vector of state variables, $\dot{\mathbf{x}}$ is a vector of time derivatives $\mathrm{d}\mathbf{x}/\mathrm{d}t$, $\mathbf{u}$ is a vector of inputs to the system, and $\mathbf{y}$ is a vector of output variables. The matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are matrices of appropriate dimensions containing real numbers. We will develop a mathematical model from *first principles*, the basic laws of physics, and then linearize the model.

The laboratory rig is 1.5 m long, and is equipped with power supply and equipment for both analog and digital control. By pushing the cart left and right manually, it is possible to get the ball near the top of the arc, but it is impossible to position the cart at a particular position at the same time.

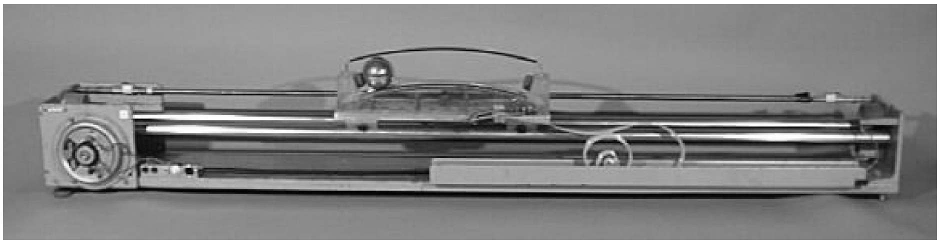


Figure 8.1: Laboratory rig. The cart moves back and forth, at the command of a controller, in order to balance the ball.

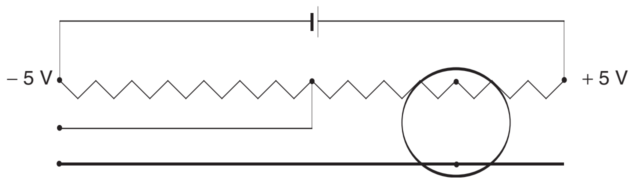


Figure 8.2: Measurement of the ball's position. The steel ball acts as a voltage divider.

However, an automatic-controller system is capable of achieving this balance. The position of the cart and angle of the ball from the vertical are measured variables, and the manipulated variable is the horizontal force acting on the cart.

The ball rolls on two curved pipes: one is made of aluminium, while the other is a coil of resistance wire. The angle of the ball from the vertical is determined by measuring its position on the pipes. The ball, being made of steel, connects the pipes electrically, and acts as a voltage divider, producing a voltage proportional to the position (Figure 8.2).

The position of the cart is measured the same way using a carbon wheel contact, mounted on the cart, which rolls on a coil alongside the rails.

The rails are cylindrical bars mounted on the support, and the wheels of the cart are small, low-friction ball-bearings that roll on the bars. A wire pulls the cart, passing over a pulley at one end and a wire drum at the other end, both attached to the support. The wire drum is driven by a current-driven direct current (DC) print-motor. Although the motor is current-driven, we assume that the voltage is proportional to the current and in turn that the force is proportional to the current. This is an approximation, but it is a relatively fast DC motor with small electrical and mechanical time-constants.

8.2 Mathematical Model

The current to the motor is a function of the controlled variables y, $\dot{y}$, φ, and $\dot{\varphi}$, where y is the position of the cart and φ is the angular deviation from the vertical of the position of the ball. The velocity signals are not directly measured, but obtained by differentiation in operational amplifiers.

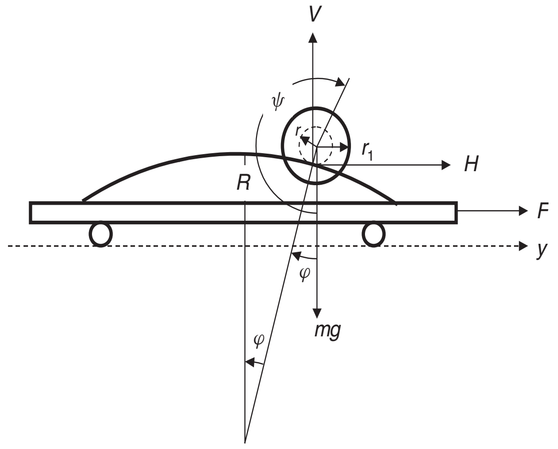


Figure 8.3: Definition of symbols and directions.

All directions are assumed positive towards the right. We apply the basic physical equations related to the vertical reaction force V and the horizontal reaction force H. Friction forces are neglected. The symbols are defined in Figure 8.3.

By Newton's second law, the horizontal movement of the ball is

$$m\frac{\mathrm{d}^2}{\mathrm{d}t^2}\left[y + (R+r)\sin\varphi\right] = H \tag{8.2}$$

the vertical movement of the ball is

$$m\frac{\mathrm{d}^2}{\mathrm{d}t^2}\left[(R+r)\cos\varphi\right] = V - mg \tag{8.3}$$

the rotational movement of the ball is

$$I\ddot{\psi} = r\left(V\sin\varphi - H\cos\varphi\right) \tag{8.4}$$

and the horizontal movement of the cart is

$$M\ddot{y} = F - H \tag{8.5}$$

The geometrical relationship between φ and ψ is

$$\varphi R = r\left(\psi - \varphi\right) \Leftrightarrow \psi = \frac{R+r}{r}\varphi \tag{8.6}$$

The variables ψ, V, and H can be eliminated from Equations (8.2)–(8.6), yielding two second-order differential equations in φ and y:

$$(M+m)\,\ddot{y} = -m\,(R+r)\left(\ddot{\varphi}\cos\varphi - \dot{\varphi}^2\sin\varphi\right) + F \tag{8.7}$$

$$I\frac{R+r}{r}\ddot{\varphi} = mr\,(R+r)\left(-\ddot{\varphi}\sin^2\varphi \; - \dot{\varphi}^2\cos\varphi\sin\varphi\right) \tag{8.8}$$
$$+mgr\sin\varphi + Mr\ddot{y}\cos\varphi - Fr\cos\varphi$$

They are nonlinear owing to the trigonometric functions. They are coupled such that $\ddot{y}$ appears on the left side of (8.7) and on the right side of (8.8), and the converse is true of $\ddot{\varphi}$. According to the first equation, the force F directly affects the acceleration of the cart $\ddot{y}$, which is also affected by the ball. According to the second equation, the force causes a torque $Fr\cos\varphi$ on the ball, affecting the angular acceleration of the ball directly. The acceleration of the cart $\ddot{y}$ also affects the angular acceleration of the ball.

We linearize at this point by introducing the following approximations:

$$\cos\varphi \approx 1, \sin\varphi \approx \varphi, \cos^2\varphi \approx 1, \sin^2\varphi \approx 0, \dot{\varphi}^2 \approx 0 \tag{8.9}$$

with $|\varphi| \leq 0.22$ rad. Consequently, we achieve a reduced system of equations:

$$(M+m)\,\ddot{y} = -m\,(R+r)\,\ddot{\varphi} + F$$

$$I\frac{R+r}{r}\ddot{\varphi} = mgr\varphi + Mr\ddot{y} - Fr$$

After rearranging,

$$\ddot{y} = -\frac{m^2r^2g}{MI+mI+mr^2M}\varphi + \frac{mr^2+I}{MI+mI+mr^2M}F$$

$$\ddot{\varphi} = \frac{mr^2g\,(M+m)}{(R+r)\left(MI+mI+mr^2M\right)}\varphi - \frac{mr^2}{(R+r)\left(MI+mI+mr^2M\right)}F$$

and introducing the substitution variables,

$$a = -\frac{m^2r^2g}{MI+mI+mr^2M}$$

$$b = \frac{mr^2+I}{MI+mI+mr^2M}$$

$$c = \frac{mr^2g\,(M+m)}{(R+r)\left(MI+mI+mr^2M\right)}$$

$$d = -\frac{mr^2}{(R+r)\left(MI+mI+mr^2M\right)}$$

and the state variables $x_1 = y$, $x_2 = \dot{y}$, $x_3 = \varphi$, and $x_4 = \dot{\varphi}$, a simple linear state-space model emerges:

$$\dot{x}_1 = \dot{y}$$

$$\dot{x}_2 = a\varphi + bF$$

$$\dot{x}_3 = \dot{\varphi}$$

$$\dot{x}_4 = c\varphi + dF$$

or, in matrix form,

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & c & 0 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ b \\ 0 \\ d \end{bmatrix}u \tag{8.10}$$

The input u comprises just one input, the driving force F, which can be substituted by the voltage U to the motor, as their relationship is 1 V to 1 N; this is how the motor and the gear were designed.

With the data in Table 8.1 the actual values of the constants are

$$\langle a, b, c, d\rangle = \langle -1.34, 0.301, 14.3, -0.386\rangle \tag{8.11}$$

A *signal flow graph* provides an overview. Given a state-space model with matrices containing zeros and non-zero elements, the flow of the signals can be mapped into a directed graph, or *digraph*; the digraph portrays the couplings in the model (Figure 8.4).

The node set is given by one input node and four state nodes; the arc set is given by the non-zero entries in the matrices, that is, if $a_{ij} \neq 0$, then there exists an arc from the jth state node to the ith state node; if $b_{ij} \neq 0$, then there exists an arc from the jth input node to ith state node. The numbers a_{ij} and b_{ij} are assigned to the arcs as *weights*. Each weight a_{ij} and b_{ij} is implicitly associated with an integration as well.

Table 8.1: Physical data for the cart–ball rig.

Object	Measure	Symbol	Ratings
Cart	Length		0.35 [m]
	Width		0.12 [m]
	Radius of the arc	R	0.50 [m]
	Mass (including equivalent mass of motor and transmission)	M	3.1 [kg]
Ball	Maximum angle	φ	±0.22 [rad]
	Radius	r_1	0.0275 [m]
	Rolling radius	r	0.025 [m]
	Mass	m	0.675 [kg]
	Moment of inertia, $\frac{2}{5}mr_1^2$	I	$0.204 * 10^{-3}$ [kg m^2]
Support	Bar length		1.4 [m]
	Bar diameter		0.025 [m]
	Motor power		Max 21 [W]
	Motor voltage	U	Max 13 [V]
	Motor transmission ratio	$U{:}F$	1:1
	Motor speed		3700 [r.p.m.]
	Gravity	g	9.81 [m/s^{-2}]

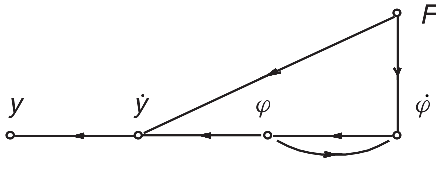


Figure 8.4: Signal flow in the state-space model.

A feedback connection from the node φ to the input node F will create a loop $\varphi - F - \dot{\varphi} - \varphi$. Alternatively, a feedback connection from the output node y to the input node F creates a larger loop, $y - F - \dot{\varphi} - \varphi - \dot{y} - y$, as well as another loop, $y - F - \dot{y} - y$. In the first case, there is no feedback from the cart $\{y, \dot{y}\}$ to the ball $\{\varphi, \dot{\varphi}\}$, while in the second case, there is feedback through the ball into the cart. Therefore, a ball controller can be designed and tuned independent of how the cart behaves, while a cart controller will be influenced by the behaviour of the ball.

Since state-space models are not unique – a given physical plant may be modelled by several state-space models – the digraph reflects the flow of signals in the model, and not necessarily the physical plant itself. However, in this case, the state variables have a physical interpretation.

The control task is to design a controller that balances the ball and places the cart in the middle of the track, given the following initial values (Table 8.1)

$$\mathbf{x}_0 = [0.525 \quad 0 \quad -0.22 \quad 0]^T$$

The cart is thus placed at the far right end of the track: the bar length is 1.40 m, the cart length is 0.35 m, and the position y is half the bar length minus half the cart length or 0.525 m. The ball is on the left, leaning against its endstop: the angle φ is negative, and the max angle is 0.22 rad. Therefore, the controller must initially pull the cart left to get the ball up on top.

8.3 Step 1: Design a Crisp PID Controller

To find a set of feedback gains, we shall apply state feedback control, before designing a PID controller. A state feedback controller generates a control signal from the values of the state variables,

$$F = \mathbf{k}^T \mathbf{x}$$

or,

$$F = k_1 y + k_2 \dot{y} + k_3 \varphi + k_4 \dot{\varphi} \tag{8.12}$$

The force can be viewed as having a cart component F_c and a ball component F_b, simply by placing two sets of parentheses,

$$F = F_c + F_b = (k_1 y + k_2 \dot{y}) + \big(k_3 \varphi + k_4 \dot{\varphi}\big)$$

The feedback gains $k_1, \ldots, k_4$ are tuning constants, and the linear control problem is the following: to design a set of feedback gains $\mathbf{k}$ such that the system

$$\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x} + \boldsymbol{B}F \tag{8.13}$$

$$F = \mathbf{k}^T \mathbf{x} \tag{8.14}$$

is stable. Note that at this point we approximate the real system with a linear model, and especially that the limitation of 13 V on the motor voltage is missing in the given formulation.

By inserting Equation (8.14) in Equation (8.13), the closed-loop system equations are obtained:

$$\dot{\mathbf{x}} = \boldsymbol{A}\mathbf{x} + \boldsymbol{B}\mathbf{k}^T\mathbf{x} = \big(\boldsymbol{A} + \boldsymbol{B}\mathbf{k}^T\big)\mathbf{x}$$

Stability of the closed-loop system is guaranteed if all eigenvalues of the closed-loop system matrix $\boldsymbol{A} + \boldsymbol{B}\mathbf{k}^T$ are in the left half of the complex plane. A closer investigation will show that all ks must be strictly positive (apply, for example, Routh's stability criterion). Consequently, all four state variables must be available to the controller. In the laboratory rig, only the positions are directly measurable, but the velocities are computed electronically, and therefore indirectly measurable.

It is important to realize that the cart has positive feedback. Since k_1 must be positive, a positive (negative) position y will generate a positive (negative) contribution F_c to the control signal. As all directions are assumed positive towards the right, the position term of F_c will try to push it right when it is on the right side of the centre of the track, that is, *away* from the centre. Whenever F_c pushes the cart away, the ball will fall to the opposite side, and F_b will provide a signal of opposite sign, pulling the cart back again towards the centre.

It is a tremendous task, if at all possible, to find four stabilizing gains by hand-tuning; this is a reason for taking this detour around state-space modelling. One possibility is to optimize the frequency response with respect to the sensitivity circle defined earlier (using the function `fminsearch` in Matlab). A controller with the sensitivity $M_s = 2$ results from the following values:

$$\mathbf{k}^T = \begin{bmatrix} 21 & 21 & 194 & 46 \end{bmatrix} \tag{8.15}$$

Figure 8.5 shows the response of both the cart and the ball. The two control signal components F_c and F_b are more or less in counter-phase. Other settings are possible, depending on the choice of sensitivity, and other design methods are also possible, for example, pole placement or optimal control (see, e.g. the `lqr` function in the Control Toolbox of Matlab).

We will assume

$$F_b = 194\varphi + 46\dot{\varphi},$$

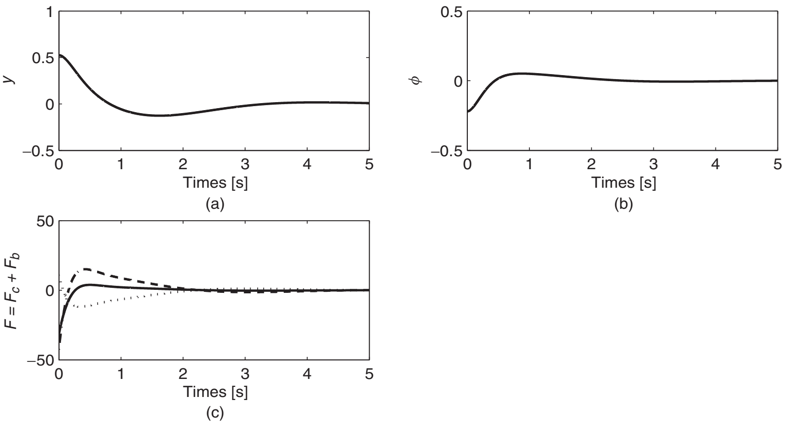


Figure 8.5: Response with $\mathbf{k}^T = \begin{bmatrix} 21 & 21 & 194 & 46 \end{bmatrix}$ and no limitation on the control signal. The cart (a) starts at 0.525 m and settles after about 4 seconds. The ball (b) starts at -0.22 rad and settles a little earlier. The control signal (c) is the sum of two large components. (figpend.m)

using the gains k_3 and k_4 as above, and design a PD controller for F_c on the basis of the gains k_1 and k_2 as above. Using Equation (8.13), we have

$$\begin{aligned}\dot{\mathbf{x}} &= \boldsymbol{A}\mathbf{x} + \boldsymbol{B}F \\ &= \boldsymbol{A}\mathbf{x} + \boldsymbol{B}\,(F_c + F_b) \\ &= \boldsymbol{A}\mathbf{x} + \boldsymbol{B}\,(194\varphi + 46\dot{\varphi}) + \boldsymbol{B}F_c\end{aligned}$$

For F_c, we use

$$F_c = K_p(e + T_d\dot{e})$$

which we can substitute with a fuzzy controller later. Define

$$e = Ref - y$$

With $Ref = 0$, corresponding to the centre of the track, substitution yields

$$\begin{aligned}F_c &= K_p(-y + T_d\,(-\dot{y})) \\ &= -K_p y - K_p T_d \dot{y}\end{aligned}$$

We see that $-K_p$ corresponds to k_1 and $-K_pT_d$ corresponds to k_2. Therefore,

$$\begin{aligned}K_p &= -21 \\ T_d &= 1 \\ k_3 &= 194 \\ k_4 &= 46\end{aligned}$$

Thus we have achieved a set of tuning parameters to start with.

8.4 Step 2: Replace It with a Linear Fuzzy

A linear fuzzy controller, built in accordance with the five particular design choices (Chapter 3), with the fuzzy gains

$$\begin{aligned}GE &= 21 \\ GU &= K_p/GE = -1 \\ GCE &= T_d * GE = 21\end{aligned}$$

provides the same response as the PD controller dealt with earlier. The magnitude of the position y is at most 0.525 m, and therefore the magnitude of the error $E = GE * e$ is at most $21 * 0.525 = 11$, which is inside the 100-limit of the universe. For the change in error $CE = GCE * \dot{e}$, the magnitude of $\dot{e}$ is at most 1.22 m/s (from simulation), and CE is at most $21 * 1.22 = 26$, which is also within the limit of the universe. Thus, there is no saturation in the universe, and the controller is equivalent to the PD controller. We could apply α-scaling to exploit the full universe, but it is not necessary when the controller is linear; the main point is to avoid saturation in the premise universes.

8.5 Step 3: Make It Nonlinear

The linear control surface can now be replaced by the nonlinear standard control surfaces successively.

The Nyquist plot in Figure 8.6 shows the relative robustness of the four surfaces. The linear frequency response touches the sensitivity circle, as desired. The saturation surface is less robust, and the two remaining surfaces cut through the sensitivity circle: we can disregard those from now on. The gain settings are as stated earlier. The linear surface is the most robust at these settings when amplitude equals 1.

8.6 Step 4: Fine-tune It

We now go on to study the step response in the time domain, with the control action limited by the amplifier to ± 13 V or, equivalently, ± 13 N. We fine-tune the gains by minimizing the integral absolute error (IAE),

$$IAE = \int |e| \mathrm{d}t$$

or rather its discrete approximation using rectangular integration. This can be done using multi-dimensional nonlinear optimization (`fminsearch` again), given a function $IAE = g(GE, GCE, GU)$. The function g simulates a response and calculates the IAE for a given set of tuning gains $\{GE, GCE, GU\}$. The optimizer adjusts all three gains to find the least IAE.

Figure 8.7 shows the response resulting from the linear surface, and Figure 8.8 shows the response resulting from the saturation surface.

For the linear surface, the optimizer finds a set of gains such that there is no saturation in the premise universes, and the response of the linear fuzzy is identical to the response resulting from the equivalent PD controller. For the saturation surface, the optimizer decreases the premise gains and increases the conclusion gain. The response is almost identical to the linear surface controller, and the IAE is only slightly improved. The optimizer chooses low values of GE, such that the control action is almost exclusively derivative action.

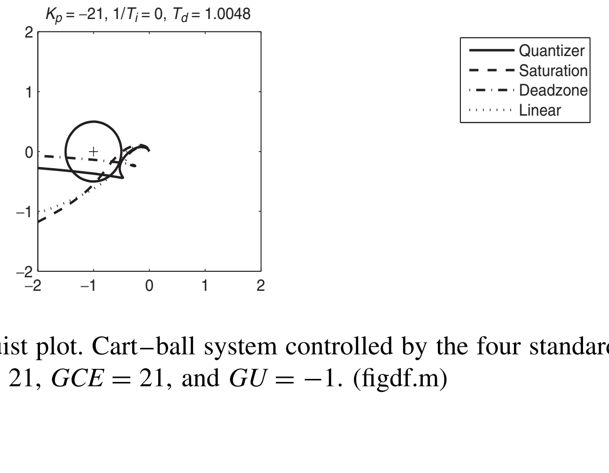


Figure 8.6: Nyquist plot. Cart–ball system controlled by the four standard control surfaces with gains $GE = 21$, $GCE = 21$, and $GU = -1$. (figdf.m)

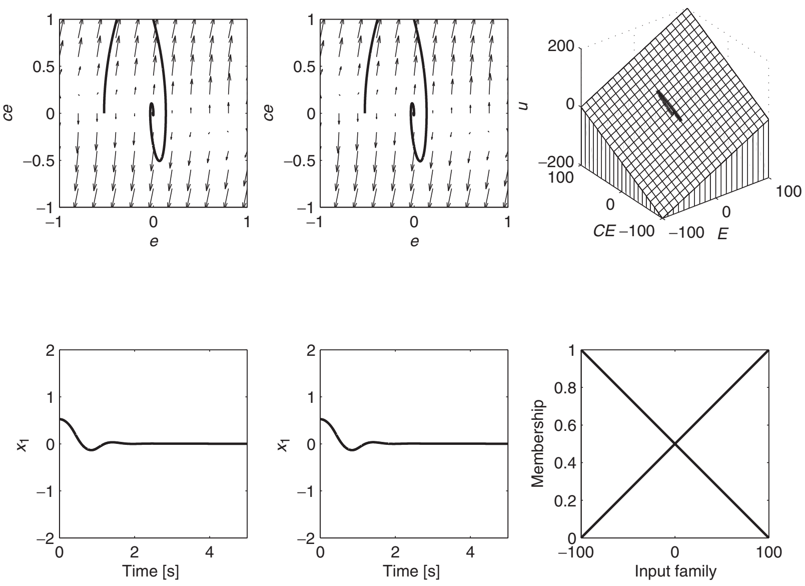


Figure 8.7: Response with settings $GE = 7.87$, $GCE = 28.9$, $GU = -1.07$, and $IAE = 0.2556$. The equivalent PD controller (first column), the FPD controller (second column), and the control surface (third column). (figpend.m)

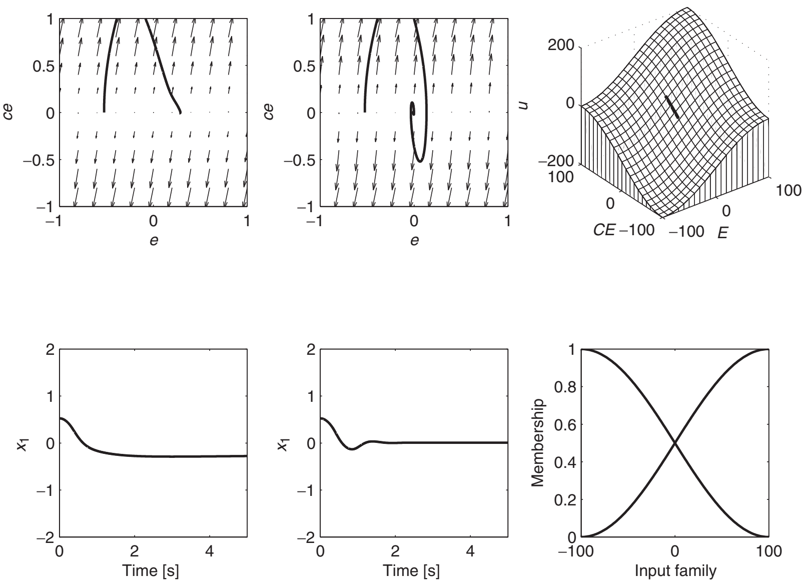


Figure 8.8: Response with $GE = 0.464$, $GCE = 15.8$, $GU = -1.23$, and $IAE = 0.2541$. The equivalent PD controller (first column), the FPD controller (second column), and the control surface (third column). (figpend.m)

Given the IAE criterion, the optimizer approach seems to be a fair comparison of controllers when there are several design factors present. In our case, the saturation surface provides an *IAE* = 0.2541 compared to an *IAE* = 0.2556 for the linear controller. The performance is thus better. But judging from the step response by inspection, the improvement is hardly noticeable.

8.7 Further State-Space Analysis*

The linear state-space model provides further insight into the controllability and stability of the system.

Open loop

Using Laplace notation $s = \mathrm{d}/\mathrm{d}t$, the general state-space model is

$$s\mathbf{x} = \boldsymbol{A}\mathbf{x} + \boldsymbol{B}\mathbf{u}$$

$$\mathbf{y} = \boldsymbol{C}\mathbf{x}$$

Rearranging the state equations,

$$(s\boldsymbol{I} - \boldsymbol{A})\mathbf{x} = \boldsymbol{B}\mathbf{u}$$

where $\boldsymbol{I}$ is the identity matrix, and solving for $\mathbf{x}$, we get

$$\mathbf{x} = (s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}\mathbf{u}$$

By substituting this value into the output equations , we obtain

$$\mathbf{y} = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}\mathbf{u}$$

Thus, the transfer functions from the inputs to the outputs are

$$\begin{aligned}\frac{\mathbf{y}}{\mathbf{u}} &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} \\ &= \boldsymbol{C}\frac{\operatorname{adj}(s\boldsymbol{I} - \boldsymbol{A})}{\det(s\boldsymbol{I} - \boldsymbol{A})}\boldsymbol{B} \qquad (8.16)\end{aligned}$$

where adj is the adjoint of the matrix and det is the determinant. It is a matrix expression representing several transfer functions, one from each input to each output in general. The denominator governs the stability of the system. Therefore, we are interested in the characteristic equation

$$\det(s\boldsymbol{I} - \boldsymbol{A}) = 0 \qquad (8.17)$$

It is a polynomial whose roots are the poles of the system, and it is an eigenvalue problem. An eigenvalue λ of $\boldsymbol{A}$ satisfies

$$\boldsymbol{A}\mathbf{v} = \lambda\mathbf{v}$$

where $\mathbf{v}$ is the eigenvector corresponding to λ. Rearranging, we get

$$(\boldsymbol{A} - \lambda\boldsymbol{I})\mathbf{v} = 0$$

*Can be skipped in a first reading.

For any proper solution to exist, the matrix $(\boldsymbol{A} - \lambda \boldsymbol{I})$ must be singular, or

$$\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0$$

which is equivalent to Equation (8.17). Thus, the eigenvalues of $\boldsymbol{A}$ are the solutions to the characteristic equation, and therefore the eigenvalues of $\boldsymbol{A}$ are also the poles of the system.

In our case, the characteristic polynomial is

$$\begin{aligned}
\det(s\boldsymbol{I} - \boldsymbol{A}) &= \det\left(\begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & s \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & c & 0 \end{pmatrix}\right) \\
&= \det\begin{pmatrix} \boxed{s} & -1 & 0 & 0 \\ 0 & \boxed{s} & -a & 0 \\ 0 & 0 & s & -1 \\ 0 & 0 & -c & s \end{pmatrix} \\
&= s^2(s^2 - c)
\end{aligned}$$

The matrix is block triangular, with three blocks in the diagonal, indicated by boxes, and the determinant of the whole is just the product of the determinants of each diagonal block. The roots are $\lambda = \{0, \pm\sqrt{c}\}$, and we observe that the constant a is not present. Thus, in the presence of uncertainty, an inaccurate a does not affect the stability of the model, while an inaccurate c does.

We can also directly relate the eigenvalues to physics. The double eigenvalue in zero results from the movement of the cart – compare Newton's second law Equation (8.5) – and the second-degree polynomial $(s^2 - c)$ results from the internal loop concerning the ball. We are thus dealing with a double integrator, which we have studied earlier, combined with an unstable second-order system.

The adjoint is

$$\begin{aligned}
\operatorname{adj}(s\boldsymbol{I} - \boldsymbol{A}) &= \operatorname{adj}\begin{pmatrix} \mathbf{s} & -1 & 0 & 0 \\ 0 & \mathbf{s} & -a & 0 \\ 0 & 0 & \mathbf{s} & \mathbf{-1} \\ 0 & 0 & \mathbf{-c} & \mathbf{s} \end{pmatrix} \\
&= \begin{pmatrix} -cs + s^3 & -c + s^2 & as & a \\ 0 & -cs + s^3 & as^2 & as \\ 0 & 0 & s^3 & s^2 \\ 0 & 0 & cs^2 & s^3 \end{pmatrix}
\end{aligned}$$

The adjoint of any n-by-n matrix $\boldsymbol{M}$ is the transpose of the matrix of cofactors of the elements m_{ij}. A cofactor is the signed minor of the element m_{ij} and the sign is $(-1)^{i+j}$. The minor is the determinant of the $(n-1)$-square submatrix obtained by deleting row i and column j. Because of the structure of $\boldsymbol{B}$,

$$\boldsymbol{B} = \begin{pmatrix} 0 \\ b \\ 0 \\ d \end{pmatrix}$$

we are only interested in the second and fourth column of the adjoint, and furthermore just the first row, in order to find the transfer function from the input to the cart position y, the first state variable. Thus,

$$\begin{aligned}
\frac{y}{F} &= \boldsymbol{C}\frac{\operatorname{adj}(s\boldsymbol{I}-\boldsymbol{A})}{\det(s\boldsymbol{I}-\boldsymbol{A})}\boldsymbol{B} \\
&= \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}\frac{\begin{pmatrix}-cs+s^3 & -c+s^2 & as & a \\ 0 & -cs+s^3 & as^2 & as \\ 0 & 0 & s^3 & s^2 \\ 0 & 0 & cs^2 & s^3\end{pmatrix}}{s^2(s^2-c)}\begin{pmatrix}0\\ b\\ 0\\ d\end{pmatrix} \\
&= \frac{\left(-c+s^2\right)b+ad}{s^2(s^2-c)}
\end{aligned}$$

The numerator is second degree, and the roots are

$$\pm\sqrt{\frac{-ad}{b}+c} = \pm 3.549$$

Thus, we have a zero in the right half-plane, and the system is a non-minimum phase system, which means that its step response will initially set out in the 'wrong' direction, away from the final value.

Closed loop

With a control law

$$\mathbf{u} = \mathbf{k}^T\mathbf{x}$$

the closed-loop system equations are

$$\begin{aligned}
s\mathbf{x} &= \boldsymbol{A}\mathbf{x} + \boldsymbol{B}\mathbf{u} \\
&= \boldsymbol{A}\mathbf{x} + \boldsymbol{B}\mathbf{k}^T\mathbf{x} \\
&= \left(\boldsymbol{A} + \boldsymbol{B}\mathbf{k}^T\right)\mathbf{x}
\end{aligned}$$

The control problem is to construct the vector **k** such that the closed-loop system matrix has all its eigenvalues in the left half-plane. If, and only if, the pair $(\boldsymbol{A}, \boldsymbol{B})$ is *controllable*, the eigenvalues can be assigned to any values in the complex plane arbitrarily by means of static state feedback. The pair is controllable if

$$\operatorname{rank}\left[\boldsymbol{A}^0\boldsymbol{B}, \boldsymbol{A}^1\boldsymbol{B}, \ldots, \boldsymbol{A}^{n-1}\boldsymbol{B}\right] = n$$

where n is the order of the system and the comma notation signifies that the matrices are concatenated ('glued') with each other. In our case,

$$\boldsymbol{A}^0\boldsymbol{B} = \boldsymbol{I}\boldsymbol{B} = \begin{pmatrix}0\\ b\\ 0\\ d\end{pmatrix}$$

$$\boldsymbol{A}^1\boldsymbol{B} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & c & 0 \end{pmatrix} \begin{pmatrix} 0 \\ b \\ 0 \\ d \end{pmatrix} = \begin{pmatrix} b \\ 0 \\ d \\ 0 \end{pmatrix}$$

$$\boldsymbol{A}^2\boldsymbol{B} = \begin{pmatrix} 0 & 0 & a & 0 \\ 0 & 0 & 0 & a \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & c \end{pmatrix} \begin{pmatrix} 0 \\ b \\ 0 \\ d \end{pmatrix} = \begin{pmatrix} 0 \\ ad \\ 0 \\ cd \end{pmatrix}$$

$$\boldsymbol{A}^3\boldsymbol{B} = \begin{pmatrix} 0 & 0 & 0 & a \\ 0 & 0 & ac & 0 \\ 0 & 0 & 0 & c \\ 0 & 0 & c^2 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ b \\ 0 \\ d \end{pmatrix} = \begin{pmatrix} ad \\ 0 \\ cd \\ 0 \end{pmatrix}$$

Thus, we test whether

$$\text{rank} \begin{pmatrix} 0 & b & 0 & ad \\ b & 0 & ad & 0 \\ 0 & d & 0 & cd \\ d & 0 & cd & 0 \end{pmatrix}$$

is equal to the system order. By inserting numbers, it is seen that the rank is indeed 4, and the system is therefore pole assignable. The constant d appears in many locations, and if d were close to zero, the determinant would be close to zero; this would occur if the mass of the cart was much larger than the mass of the ball.

Knowing that we can place the poles arbitrarily, we would like to place a set of poles in a 'good' location and work backwards to find a corresponding feedback vector **k**. The closed-loop characteristic polynomial is

$$\begin{aligned} P &= \det\left(s\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{B}\mathbf{k}^T\right) \\ &= \det\left(\begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & s \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & c & 0 \end{pmatrix} - \begin{pmatrix} 0 \\ b \\ 0 \\ d \end{pmatrix} \begin{pmatrix} k_1 & k_2 & k_3 & k_4 \end{pmatrix}\right) \\ &= \det \begin{pmatrix} s & -1 & 0 & 0 \\ -bk_1 & s - bk_2 & -a - bk_3 & -bk_4 \\ 0 & 0 & s & -1 \\ -dk_1 & -dk_2 & -c - dk_3 & s - dk_4 \end{pmatrix} \\ &= s^4 + s^3\left(-bk_2 - dk_4\right) + s^2\left(-c - bk_1 - dk_3\right) \qquad (8.18) \\ &\quad + s\left(bck_2 - adk_2\right) + bck_1 - adk_1 \end{aligned}$$

The characteristic polynomial can also be written in terms of the desired poles:

$$P = (s-\lambda_1)(s-\lambda_2)(s-\lambda_3)(s-\lambda_4) \tag{8.19}$$

$$\begin{aligned} &= s^4 \\ &+ s^3(-\lambda_1-\lambda_2-\lambda_3-\lambda_4) \\ &+ s^2(\lambda_1\lambda_2+\lambda_1\lambda_3+\lambda_1\lambda_4+\lambda_2\lambda_3+\lambda_2\lambda_4+\lambda_3\lambda_4) \\ &+ s(-\lambda_1\lambda_2\lambda_3-\lambda_1\lambda_2\lambda_4-\lambda_1\lambda_3\lambda_4-\lambda_2\lambda_3\lambda_4)+\lambda_1\lambda_2\lambda_3\lambda_4 \end{aligned} \tag{8.20}$$

Matching the coefficients in Equations (8.18) and (8.19), we have four linear equations in the unknowns $k_1, \ldots, k_4$:

$$\begin{aligned} -bk_2 - dk_4 &= -\lambda_1-\lambda_2-\lambda_3-\lambda_4 \\ -c - bk_1 - dk_3 &= \lambda_1\lambda_2+\lambda_1\lambda_3+\lambda_1\lambda_4+\lambda_2\lambda_3+\lambda_2\lambda_4+\lambda_3\lambda_4 \\ bck_2 - adk_2 &= -\lambda_1\lambda_2\lambda_3-\lambda_1\lambda_2\lambda_4-\lambda_1\lambda_3\lambda_4-\lambda_2\lambda_3\lambda_4 \\ bck_1 - adk_1 &= \lambda_1\lambda_2\lambda_3\lambda_4 \end{aligned}$$

or, in matrix form,

$$\begin{pmatrix} 0 & -b & 0 & -d \\ -b & 0 & -d & 0 \\ 0 & bc-ad & 0 & 0 \\ bc-ad & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{pmatrix}$$

$$= \begin{pmatrix} -\lambda_1-\lambda_2-\lambda_3-\lambda_4 \\ \lambda_1\lambda_2+\lambda_1\lambda_3+\lambda_1\lambda_4+\lambda_2\lambda_3+\lambda_2\lambda_4+\lambda_3\lambda_4 \\ -\lambda_1\lambda_2\lambda_3-\lambda_1\lambda_2\lambda_4-\lambda_1\lambda_3\lambda_4-\lambda_2\lambda_3\lambda_4 \\ \lambda_1\lambda_2\lambda_3\lambda_4 \end{pmatrix} + \begin{pmatrix} 0 \\ c \\ 0 \\ 0 \end{pmatrix}$$

The system of equations has a solution since the determinant $d^2(bc-ad)^2$ is non-zero (the coefficient matrix is upper triangular, and thus the determinant is the product of the diagonal elements). Given a set of desired poles $\lambda_1, \ldots, \lambda_4$, we can thus find a unique solution $k_1, \ldots, k_4$.

How do we then decide what 'good' pole locations are? As a rule of thumb, the farther away the pole is from the origin, the larger is the magnitude of the control signal required. If a pole λ is located at $\sigma + j\omega$, then, roughly speaking, ω affects the frequency of the response and σ affects the damping. Thus, poles on the real axis cause purely exponential terms in the impulse response, and complex poles may cause sinusoidal components. For a stable system, the closed-loop poles must be in the left half of the complex plane. Poles at the origin are allowed, but should be restricted in number; a single pole at the origin generates a constant term in the impulse response, two poles generate a ramp and a constant term, and so on. Poles on the imaginary axis are not desirable, since they generate undamped sinusoidal components.

A simple way to select the poles is to use prototype response poles (Franklin *et al.* 1991, p 388, Table 6.1b),

$$\lambda_1 = -0.6573 - j0.8302$$
$$\lambda_2 = -0.6573 + j0.8302$$
$$\lambda_3 = -0.9047 - j0.2711$$
$$\lambda_4 = -0.9047 + j0.2711$$

These refer to a prototype response of a fourth-order system with little overshoot. The four poles can be multiplied by a constant w in order to obtain some variation in the speed of the response. With $w = 1$, a set of gains is

$$k_1 = 0.26$$
$$k_2 = 0.84$$
$$k_3 = 48.6$$
$$k_4 = 8.74$$

These are rather low values, compared to the ones we used earlier in Equation (8.15), and consequently the control action will be rather low. A faster response can be obtained by applying the desired eigenvalues $\{w\lambda_1, w\lambda_2, w\lambda_3, w\lambda_4\}$ with $w > 1$. In practice, the obtainable speed of the response is limited by the limits of the amplifier (± 13 V), and w should be in the range $1 < w < 2$ to minimize the IAE performance index.

Nonlinear equations

The motor has friction between the brushes and commutator as well as in the bearings. There is also friction in the drum grooves, where the transmission wire is seated. Both static and dynamic friction must be expected, and sustained limit cycles in closed loop are a clear indication. The linear model ignores friction.

As a first approximation, we may model friction as viscous friction proportional to the velocity of the cart directed in the direction opposite to that of the driving force. In the equations we may then apply, instead of the driving force F, a reduced force

$$F - f\dot{y} \tag{8.21}$$

Here, f is a physical constant that models the magnitude of the frictional effect. The constant f can be measured at constant velocity ($F = 0$). We are, however, just interested in discovering how friction affects the system matrix $\boldsymbol{A}$.

Substitution of Equation (8.21) for F in (8.13) results in two new non-zero terms in $\boldsymbol{A}$,

$$\boldsymbol{A}(2, 2) = -bf$$
$$\boldsymbol{A}(4, 2) = -df$$

As a result, one eigenvalue will remain zero, whereas the other three will be non-zero. Friction in fact has a stabilizing effect.

Another question is the magnitude of the error introduced by the linearization by Equation (8.9). If we instead omit the approximation, and rearrange Equations (8.7) and

(8.8), we obtain the nonlinear state-space equations:

$$
\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -\frac{m^2r^2g\cos x_3 \sin x_3 - \left(RmI + mrI + m^2r^3 + Rm^2r^2\right) x_4^2 \sin x_3}{MI + mI + mr^2M + m^2r^2\sin^2 x_3} \\
&\quad + \frac{mr^2 + I}{MI + mI + mr^2M + m^2r^2\sin^2 x_3} F \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \frac{mr^2g\,(M+m)\sin x_3 - m^2r^2\,(R+r)\,x_4^2\cos x_3 \sin x_3}{(R+r)\left(MI + mI + mr^2M + r^2m^2\sin^2 x_3\right)} \\
&\quad - \frac{mr^2\cos x_3}{(R+r)\left(MI + mI + mr^2M + r^2m^2\sin^2 x_3\right)} F
\end{aligned}
\tag{8.22}
$$

A simulation using the nonlinear state Equations (8.22) rather than the linear Equations (8.10) shows that the difference in the peak values of a response is less than 3%.

8.8 Summary

The simulation study demonstrates there is much to gain from the linear control theory. The recommendation is to extract as much process knowledge as possible, when a mathematical model is available. For example, knowing that the system is pole assignable, and that all four feedback gains in the closed-loop linear model must be strictly positive, is valuable information. Furthermore, the linear model enables several approaches for tuning the gains.

Again, we can conclude that the fuzzy controller performs at least as well as the PD controller. It has a more complex structure, but then it provides more options for improving the performance.

8.9 Notes and References

The ball balancer was built for teaching students of electrical engineering automatic control, originally with a focus on state-space control theory. It is educational, because the laboratory rig is sufficiently slow for visual inspection of different control strategies and the mathematical model is sufficiently complex to be challenging.

The system was built during an M.Sc. project, and later the mathematical model was published in an educational journal (Jørgensen 1974). A simulator of the same system was built in Matlab much later for a course on the Internet (Jantzen 2003).

9

Supervisory Control*

Human operators in the process industry are faced with nonlinear and time-varying behaviour, many inner loops, and much interaction between the control loops. Owing to sheer complexity it is impossible, or at least very expensive, to build a mathematical model of the plant, and furthermore the control is normally a combination of sequential, parallel, and feedback control actions.

Operators, however, are able to control complicated plants using their experience and training, and thus fuzzy control is a relevant method within supervisory control. The cement kiln controller by FL Smidth (FLS) was based on rules of thumb for manual control of a cement kiln. In a further development, several controllers are combined in a hierarchy by means of a priority management system.

9.1 Process Control

By *process control* we shall understand the automation of a large-scale, industrial plant of such complexity, that the satisfaction of a control specification requires compromise. Typical examples are the control of distillation columns, glass, cement, and plastic production, and electric power plants. Some typical goals for a *supervisory controller* are safe operation, highest product quality, and economic operation. The three goals are in conflict, so they must be prioritized, with human safety accorded the highest priority. A *supervisory system* is a system that evaluates whether local controllers satisfy prespecified performance criteria, diagnoses causes for deviation from the performance criteria, plans actions, and executes the planned actions (Yazdi 1997).

The so-called *Supervisory Control Theory* (Ramadge and Wonham in Yazdi 1997) is based on finite state machines. Here, the supervisor is viewed as a controller that restricts the plant behaviour to a subset of all possible behaviours by starting or stopping events. To avoid generating the large space of possible behaviours, a more compact logic-based approach can be applied. *Petri nets* for discrete event systems allow intuitive modelling of the plant; see the survey by David and Alla (1992). The Petri net is the basis for

*Can be skipped in a first reading.

the *GRAFCET*, a graphical language for modelling sequential controllers (David 1995). A version of GRAFCET, *Sequential Function Charts*, has been adopted as an international standard (IEC 2002) that is widely used in connection with programmable logic controllers (PLCs). Another modelling technique that is also associated with a graphical language, *Multi-Level Flow Modelling*, requires an explicit goal for each task and emphasizes the means to reach that goal (Lind 1994, 1999).

The FLS controller

A cement kiln is a rotating chamber with a light slope. It is sometimes 160 m long, although modern kilns are shorter because of preheating. Ground limestone, clay, and sand react chemically at temperatures around 1430 °C. The material advances slowly down the kiln in three to four hours. The process is difficult to control, because only a few measurements of the internal state are possible because of the heat and the complexity of the chemical reaction. Nevertheless, a skilled operator can be rather successful in maximizing clinker output (the product), while minimizing fuel and raw material consumption.

The operator monitors mainly four quantities: oxygen and carbon monoxide content in the exhaust gases, the temperature of the exhaust, free-lime content (indicator for the temperature in the burning zone and the quality of the product), and the change in kiln drive torque. The operator then applies 40–50 rules of thumb to control the coal feed rate, and the flow of air into the kiln. An example of a rule is the following:

If the oxygen percentage is low, and the temperature is in the upper part of the range,

then decrease the air flow and reduce fuel slightly.

The FLS controller contains predefined fuzzy terms – *Low*, *High*, and *OK* – for the measured quantities. Similarly, the program contains terms for the control actions: Medium, Negative, Large Negative, Small Negative, Medium Positive, and so on. The operator can display the rules, and the rule above may appear as

```
IF LOW(O2) AND (OK(TEMP) OR HIGH(TEMP))
THEN MNEG(DAMPER), MNEG(COAL)
```

The controller weights the control action from each rule depending on the degree of match. It determines the resulting control signal as a weighted average of all actions dictated by the rules.

An operator screen and keyboard are normally placed in the control room as an integral part of the control desk. The operator can request colour displays of time series, selected measure points, alarm surveys, and diagrams. A printer produces 24-hour reports, alarm reports, and hardcopies of graphical screens. In order to trace the data flow, one side of the screen displays the current value of each input or output variable, as well as the current force of each rule. Changing a rule requires special authority.

A skilled operator must tune the controller for a new kiln. Each kiln has its own operating behaviour, so the operator monitors its initial performance and adds or deletes rules of thumb as necessary. Since the rule language is somewhat close to natural language, the operator needs little or no computer expertise to instruct the controller.

Operators have reported savings in fuel and a more stable product quality and operation. The controller also eliminates differences in operation among various operators. FL Smidth has implemented similar process controllers for a variety of other industrial processes.

Further details are available in the articles by Holmblad and Ostergaard (1982, 1995).

Control tasks

To structure the process knowledge, a practical approach is to decompose the supervisory problem into hierarchical sub-systems or *tasks*, each having a specific purpose. For instance, to reach a specific state of the plant. Yazdi (1997) specified a standard *control task* in terms of a set of necessary properties:

Specification of a control task

Name. A name describing a task goal.

Goal. For example: safe operation, high quality, low energy consumption.

Strategic conditions. A condition set that only has to be valid for initiating the task. The strategic condition set is normally a process requirement before the task starts.

Execution conditions. A set of conditions that has to be valid during task execution. The condition set is normally related to the operational constraints, for example, physical limitations and safety conditions. Information about these properties is derived by combining operator knowledge of the operational conditions with design knowledge.

Initial actions. A set of actions that has to be carried out to prepare the task for control structuring. This property contains most binary actions (on/off, start/stop).

Control actions. A set of sensors and actuators through which proper control can be designed in order to achieve the task objective. This property describes the resources of the control function.

Achievement indicator. An indicator for the degree of goal achievement during the task operation.

Final action. A task can end with a set of final actions, initiated when the task goal has been achieved.

A control task should be defined for each (fuzzy) controller. Each control task describes a detailed strategy for reaching a specific sub-goal. Execution of a control task is only allowed if *execution* conditions are valid at each sampling instant, while *strategic* conditions are checked only at the first sampling of the task.

9.2 High-Level Fuzzy Control

The problem is to ensure that a particular overall goal is reached when all control tasks are combined. The FLS design procedure, associated with the company's second-generation fuzzy controller, Fuzzy II (Østergaard 1990, 1996), includes, for this reason, a priority management system.

A *high-level* controller works on the level below that of the human operator. It coordinates control loop setpoints, which were previously coordinated by a human operator. The basic component of fuzzy high-level control is a set of rules for automatic operation based on practical experience and knowledge about manual control. Examples of fuzzy control rules are as follows:

IF Temperature is High and Pressure is Ok THEN Medium Flow

IF Temperature is Ok and Pressure is Ok THEN Small Flow

Since the condition of a rule is fulfilled to a degree, each rule will influence the result of the set of rules in accordance with its activation.

This heuristic design approach is useful – even if it is not the only practical approach – when the process is partly unknown, difficult to describe by a mathematical model, if few measurements are available, or if the process is highly nonlinear.

High-level control performance

The meaning of improved performance is not always obvious, and it may depend on local conditions such as the present market situation, raw material costs, and overall strategic goals.

In most cases, performance will relate to profit in terms of reduced costs or increased productivity. The consumption of energy and raw materials depends on the supervisory control. In general, improved performance can be defined as

- having a more stable (i.e. steady) operation;
- running closer to the limits for acceptable product quality; and
- running closer to the environmental emission limits.

Steady operation is the most important key for improving performance. Oscillations require energy and raw materials, reduce the quality of the product, and increase emissions. If the process oscillates, average values must be kept within safe bounds. The standard deviation *STD* may be calculated on a daily basis, and an average standard deviation is then calculated for a period that is representative of the performance.

If the measurements and the quality parameters have target values, then a more feasible measure of steady operation is obtained by calculating the *target value deviation* (*TVD*) (Østergaard in Jantzen *et al.* 1999) around the target value *SP*, instead of the variations around the average value *AVR*, as in *STD*. The index is calculated as follows:

$$TVD = \sqrt{STD^2 + (SP - AVR)^2} \tag{9.1}$$

For a cement plant, it is realistic to expect reductions of 50 %, or more in *STD* and/or *TVD*, as the result of a high-level control system.

High-level control configurations

Fuzzy controllers are combined with other controllers in various configurations in Figure 9.1. The PID block consists of independent or coupled PID loops, and the fuzzy block employs a high-level control strategy. Normally, both the PID and the fuzzy blocks have more than one input and one output.

Fuzzy replaces PID In this configuration (Figure 9.1a), the operator may select between a high-level control strategy and conventional control loops. The operator has to decide which of the two most likely produces the best control performance.

Fuzzy replaces operator This configuration (Figure 9.1b) represents the original high-level control idea, where manual control carried out by a human operator is replaced by automatic control. Normally, the existing control loops are still active, and the high-level control strategy makes adjustments of the controller setpoints in the same way as the operator does. Again it is up to the operator to decide whether manual or automatic control will result in the best possible operation of the process, which, of course, may create conflicts.

Fuzzy adjusts PID parameters In this configuration (Figure 9.1c), the high-level strategy adjusts the parameters of the conventional control loops. A common problem with linear PID control of highly nonlinear processes is that the set of controller parameters are satisfactory only when the process is within a narrow operational window. Outside this, it is necessary to use other parameters or setpoints, and these adjustments may be done automatically by a high-level strategy.

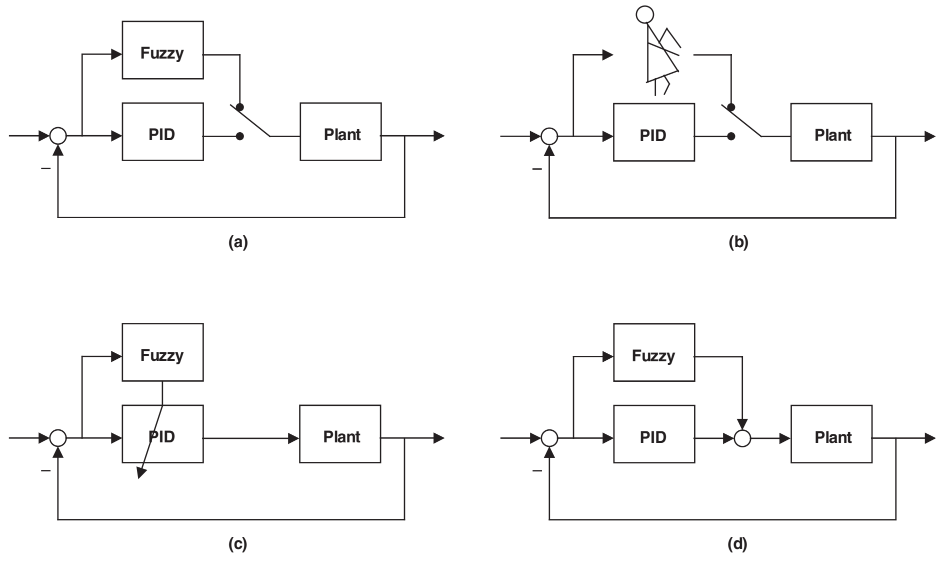


Figure 9.1: Fuzzy controller configurations. Fuzzy replaces PID (a), fuzzy replaces operator (b), fuzzy adjusts PID parameters (c), and fuzzy adds to PID control.

Fuzzy adds to PID control Normally, control systems based on PID controllers are capable of controlling the process when the operation is steady and close to normal conditions. However, if sudden changes occur, or if the process enters abnormal states, then the configuration in Figure 9.1(d) may be applied to bring the process back to normal operation as fast as possible. For normal operation, the fuzzy contribution is zero, whereas the PID outputs are compensated in abnormal situations, often referred to as *abnormal situation management* (ASM).

Configurations (a) and (b) directly change the routines of the operator, which is a crucial point to take into account when the system is developed and installed.

The FLS design procedure

For an operator, control of the process consists in achieving various goals, more or less well-defined, such as, maximum output, minimum consumption of raw materials and energy, high product quality, and safe process operation. Different processes have different control objectives but, in general, good process control may be defined through a list of control objectives that should be fulfilled to the extent possible. The concept of control objectives is a key element in a high-level control strategy.

For a cement kiln, typical control objectives are the following: stable (steady) operation, good cement clinker quality, high production, complete combustion, low fuel consumption, and low energy consumption. As control objectives are frequently in conflict, a high-level coordination is required.

In other words, priorities have to be assigned to the various control objectives, specifying which objectives are considered the most important to fulfil. The elements of the FLS design procedure for a process control strategy (Østergaard 1990, 1996) are cited below:

State index calculations Find the current process state. The calculation combines measurements into an index for the current process stability, product quality, production level, and so on. Normally, a state index combines various measurements into a single figure. The degree of process stability, the product quality, and the production level are typical examples of state indices for a kiln control strategy.

Control groups Arrange the overall control strategy into groups of control objectives. A control group, a subset of the control strategy, is a group of objectives that are related through priority numbers.

Priority management Determine the extent to which the control actions should be executed to fulfil the individual objectives. The priority management system manages the scheduling of control actions in the order of importance.

Control objectives Specify the goals of the control strategy. A control objective consists of tasks.

The state indices are important to the structure of the FLS design scheme for a high-level control strategy, as they form the basis for dividing the overall strategy into control groups that can be treated independently. The state indices are used to coordinate control actions from the various control groups.

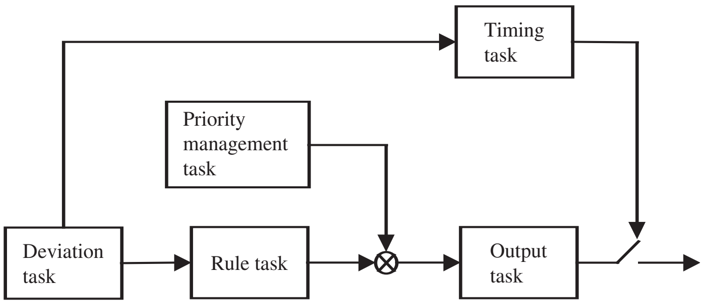


Figure 9.2: Control objective module in Fuzzy II.

Every control objective is implemented in accordance with the so-called *objective module*, which consists of five tasks, as shown in Figure 9.2.

Deviation task This task involves calculating and evaluating the degree to which the objective is fulfilled. Normally, the calculation results in an error value $e_i \in [-1, 1]$, which expresses how far the current process state is from the objective; a value of 0 signifies that the objective is fulfilled.

Rule task Normally, this block contains a set of fuzzy control rules, and the output of the rule block is normally a *change* in action in the interval $[-1, 1]$. Other techniques may also be used, such as PID, neural nets, and mathematical models.

Priority management task The rule block for each control objective results in control actions, which are multiplied by a weight factor between 0 and 1. The weight factor w_i generated by objective i is calculated as

$$w_i = 1 - |e_i| \tag{9.2}$$

The weight factor is thus a function of the deviation e_i of the objective. The smaller the weight factor, the more the lower control actions, below objective i, are suppressed. The total weighting of an objective's output is the product of all higher priority weight factors (Figure 9.3); it will be 1 if all objectives with a higher priority are fulfilled, and 0 if one or several objectives are not fulfilled. The priorities reflect built-in knowledge about optimal interaction of rule blocks.

Output task The output task involves evaluation of process constraints and selection among alternative control actions based upon the current index values; in this task, fuzzy output is converted to engineering units, that is, denormalized physical units. The logic for selecting alternative adjustments may be fuzzy or non-fuzzy depending on whether a gradual or a hard switch is the most appropriate. In most cases, the fuzzy-logic approach gives the best control performance, simply because no physical process operates with sudden changes between alternative control actions.

Timing calculation task The timing calculation consists in determining when and how often control actions are to be executed. It is just as important as the rule block for

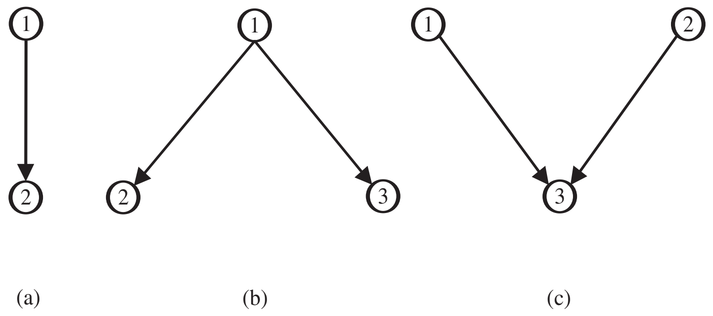


Figure 9.3: Priority management: (a) objective 1 higher than 2; (b) objective 1 affects two objectives on a lower level; (c) objectives 1 and 2 both affect an objective on a lower level.

determining the proper function of the control strategy. Even the timing calculation is normally fuzzy in the sense that the time interval between control actions changes gradually as a function of the deviation value. The larger the deviation, the more frequent the control actions.

Each objective has several tuning parameters: an output gain, input normalization, and tuning of the timing calculation. The control objective output must be a *change* in control action, an incremental controller. A suppressive weight $w_i = 0$ thus results in no change, which is equivalent to maintaining status quo.

9.3 Summary

This chapter illustrates the problems in practical process control due to the complexity of multi-loop systems, rather than the robustness of single-loop control.

On a small scale, the suggested fuzzy controller configurations (Figure 9.1) are appropriate; on a medium scale, the FLS priority system is appropriate and, on a large scale, the GRAFCET can be an option. Compared to the FLS priority system, GRAFCET is a more general type of priority manager.

9.4 Notes and References

The original work on supervisory control is due to Hassan Yazdi and Jens-Jørgen Østergaard. Yazdi developed a design procedure during his PhD project, supervised by Jens-Jørgen Østergaard, Sten Bay-Jørgensen, and myself.

There is a chapter on fuzzy supervisory control in the textbook by Passino and Yurkovich (1998).

Bibliography

von Altrock C 1995 *Fuzzy Logic and Neurofuzzy Applications Explained*. Prentice Hall.

von Altrock C 1996 *Fuzzy Logic and Neurofuzzy Applications in Business and Finance*. Prentice Hall PTR.

Aracil J and Gordillo F 2004 Describing function method for stability analysis of PD and PI fuzzy controllers. *Fuzzy Sets and Systems* **143**, 233–249.

Assilian S and Mamdani EH 1974a Learning control algorithms in real dynamic systems *Proceedings of the Fourth International Conference on Digital Computer Applications to Process Control, IFAC/IFIP*. Springer, Zürich, pp. 13–20.

Assilian S and Mamdani EH 1974b An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies* **7**(1), 1–13.

Åström KJ and Hägglund T 1995 *PID Controllers: Theory, Design, and Tuning*, 2nd edn. Instrument Society of America, ISA, The International Society for Measurement and Control, Research Triangle Park.

Åström K, Hang C, Persson P and Ho W 1992 Towards intelligent PID control. *Automatica* **28**(1), 1–9.

Åström KJ and Wittenmark B 1984 *Computer Controlled Systems –Theory and Design*. Prentice Hall.

Åström KJ and Wittenmark B 1995 *Adaptive Control*, 2nd edn. Addison-Wesley.

Atherton DP 1975 *Nonlinear Control Engineering*, unabridged edn. Van Nostrand Reinhold Company.

Atherton DP 1982 *Nonlinear Control Engineering*, student edn. Van Nostrand Reinhold Company.

Babuška R 1998 *Fuzzy Modeling for Control*. Kluwer Academic Publishers.

Bennett S 1993 Development of the PID controller. *IEEE Control Systems* **13**(5), 58–65.

Bezdek J and Pal SK 1992 *Fuzzy Models for Pattern Recognition*. IEEE Press, New York. (Selected reprints).

Braae M and Rutherford D 1979a Selection of parameters for a fuzzy logic controller. *Fuzzy Sets and Systems* **2**, 185–199.

Braae M and Rutherford D 1979b Theoretical and linguistic aspects of the fuzzy logic controller. *Automatica* **15**, 553–577.

Cohen G and Coon G 1953 Theoretical consideration of retarded control. *Transactions of American Society of Mechanical Engineers, ASME*, **75**, 827–34.

Cominos P and Munro N 2002 PID controllers: Recent tuning methods and design to specification. *IEE Proceedings – Control Theory and Applications* **149**(1), 46–53.

Cuesta F, Gordillo F, Aracil J and Ollero A 1999 Stability analysis of nonlinear multivariable Takagi-Sugeno fuzzy control systems. *IEEE Transactions on Fuzzy Systems* **7**(5), 508–520.

David R 1995 Grafcet: A powerful tool for specification of logic controllers. *IEEE Transactions on Control Systems Technology* **3**(3), 253–268.

Foundations of Fuzzy Control Jan Jantzen
© 2007 John Wiley & Sons, Ltd. ISBN: 0-470-02963-3

David R and Alla H 1992 *Petri Nets and Grafcet: Tools for Modeling Discrete Event Systems*. Prentice Hall.

Driankov D, Hellendoorn H and Reinfrank M 1996 *An Introduction to Fuzzy Control*, 2nd edn. Springer-Verlag.

Edwards C and Spurgeon SK 1998 *Sliding Mode Control: Theory and Applications* Systems and Control Book Series. Taylor & Francis.

Farinwata SS, Filev D and Langari R (eds.) 2000 *Fuzzy Control: Synthesis and Analysis*. Wiley.

Franklin GF, Powell JD and Emami-Naeini A 1991 *Feedback Control of Dynamic Systems*, Electrical and Computer Engineering: Control Engineering, 2nd edn. Addison-Wesley.

Franksen OI 1979 Group representation of finite polyvalent logic In *Proceedings 7th IFAC Triennial World Congress* (ed. Niemi A). International Federation Of Automatic Control, IFAC, Pergamon Press, Helsinki.

Fukami S, Mizumoto M and Tanaka K 1980 Some considerations of fuzzy conditional inference. *Fuzzy Sets and Systems* **4**, 243–273.

Gelb A and Vander Velde WE 1968 *Multiple-Input Describing Functions and Nonlinear System Design*. McGraw-Hill, Also available from: http://ocw.mit.edu/OcwWeb/Aeronautics-and-Astronautics/16-30Spring2004/Readings/ [cited 7 Jul 2006].

Gordillo F, Aracil J and Álamo T 1997 Determining limit cycles in fuzzy control systems *Proceedings of 6th International Fuzzy Systems Conference*, vol. 1. IEEE, pp. 193 –198.

Gupta MM and Sinha NK (eds.) 1996 *Intelligent Control Systems: Theory and Practice*. IEEE Press.

Gyöngy IJ and Clarke DW 2006 On the automatic tuning and adaptation of PID controllers. *Control Engineering Practice* **14**, 149–163.

Holmblad LP and Østergaard JJ 1982 Control of a cement kiln by fuzzy logic In *Fuzzy Information and Decision Processes* (eds. Gupta MM and Sanchez E). North-Holland, Amsterdam, pp. 389–399. (Reprint in: FLS Review No 67, FLS Automation A/S, Høffdingsvej 77, DK-2500 Valby, Copenhagen, Denmark).

Holmblad LP and Østergaard JJ 1995 The FLS application of fuzzy logic. *Fuzzy Sets and Systems* **70**, 135–146.

IEC 2000 Programmable controllers – part 7: Fuzzy control programming. Technical report IEC 61131, International Electrotechnical Commission (IEC). Draft available from http://www.fuzzytech.com/binaries/ieccd1.pdf [cited on 10 Aug 2006].

IEC 2002 Grafcet specification language for sequential function charts. Technical report 60848, International Electrotechnical Commission.

Isaksson A and Hägglund T (eds.) 2002 *Special Section on PID Control*, IEE Proceedings – Control Theory And Applications, vol. 149.

Jang JSR, Sun CT and Mizutani E 1997 *Neuro-Fuzzy and Soft Computing* Matlab Curriculum Series. Prentice Hall.

Jantzen J 1995 Array approach to fuzzy logic. *Fuzzy Sets and Systems* **70**, 359–370.

Jantzen J 2003 Internet learning in control: A fuzzy control course In *Preprints of the ACE 2003, the 6th IFAC Symposium on Advances in Control Education* (ed. Lindfors J). IFAC, pp. 27–33.

Jantzen J, Verbruggen H and Østergaard JJ 1999 Fuzzy control in the process industry: Common practice and challenging perspectives In *Practical Applications of Fuzzy Technologies*, The Handbooks of Fuzzy Sets Series (eds. Zimmermann HJ, Dubois D and Prade H). Kluwer Academic Publishers, chapter 1, pp. 3–56.

Jespersen T 1981 Self-organizing fuzzy logic control of a pH-neutralisation process. Technical report 8102, Department of Electric Power Engineering, Technical University of Denmark.

Jørgensen V 1974 A ball-balancing system for demonstration of basic concepts in the state-space control theory. *International Journal of Electrical Engineering Education* **11**, 367–376.

Kickert W and Mamdani EH 1978 Analysis of a fuzzy logic controller. *Fuzzy Sets and Systems* **1**, 29–44.

Kickert WJM and Van Nauta Lemke HR 1976 Application of a fuzzy controller in a warm water plant. *Automatica* **12**(4), 301–308.

Kiszka JB, Kochanska ME and Sliwinska DS 1985 The influence of some fuzzy implication operators on the accuracy of a fuzzy model. *Fuzzy Sets and Systems* **15**, (Part1) 111–128; (Part 2) 223–240.

Kosko B 1992 *Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence*. Prentice Hall.

Larsen PM 1981 Industrial applications of fuzzy logic control In *Fuzzy Reasoning and its Applications* (eds. Mamdani EH and Gaines BR). Academic Press, London, pp. 335–342.

Lee CC 1990 Fuzzy logic in control systems: Fuzzy logic controller. *IEEE Transactions on Systems, Man, and Cybernetics* **20**(2), 404–435.

Li HX and Gatland HB 1995 A new methodology for designing a fuzzy logic controller. *IEEE Transactions on Systems, Man, and Cybernetics* **25**(3), 505–512.

Lin CT and Lee CSG 1996 *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice Hall PTR.

Lind M 1994 Modeling goals and functions of complex industrial plants. *Applied Artificial Intelligence* **8**(2), 259–283.

Lind M 1999 Plant modeling for human supervisory control. *Transactions of the Institute of Measurement and Control* **21**(4-5), 171–180.

Mamdani EH 1977 Application of fuzzy logic to approximate reasoning using linguistic synthesis. *IEEE Transactions on Computers* **C-26**(12), 1182–1191.

Mamdani EH and Baaklini N 1975 Prescriptive method for deriving control policy in a fuzzy-logic controller. *Electronics Letters* **11**(25/26), 625–626.

Mathworks 2006 *Fuzzy Logic Toolbox for Use with Matlab: User's Guide*, version 2 online edn. The MathWorks Inc. Available from http://www.mathworks.com/access/helpdesk/help/pdf_doc/fuzzy/fuzzy.pdf [cited 14 Aug 2006]].

Mizumoto M 1992 Realization of PID controls by fuzzy control methods *First International Conference on Fuzzy Systems*. The Institute of Electrical and Electronics Engineers, San Diego, pp. 709–715.

Mizumoto M, Fukami S and Tanaka K 1979 Some methods of fuzzy reasoning In *Advances in Fuzzy Set Theory Applications* (eds. Gupta MM, Ragade RK and Yager RR). North-Holland, New York.

Møller G 1986 A logic programming tool for qualitative system design. *APL Quote Quad (APL86 Conference Proceedings)* **16**(4), 266–271.

Møller GL 1998 *On the Technology of Array-Based Logic*, PhD thesis, Technical University of Denmark, Electric Power Engineering Department, DK-2800 Lyngby, Denmark. Also available from: http://www.arraytechnology.com/downloads/lic.pdf [cited 3 Aug 2006]. (2nd ed.).

Murakami S, Takemoto F, Fulimura H and Ide E 1989 Weld-line tracking control of arc welding robot using fuzzy logic controller. *Fuzzy Sets and Systems* **32**(2), 221–237.

Nauck D, Klawonn F and Kruse R 1997 *Foundations of Neuro-Fuzzy Systems*. John Wiley & Sons.

Nguyen HT and Walker EA 2000 *A First Course in Fuzzy Logic*, 2nd edn. Chapman & Hall, New York.

Østergaard JJ 1977 Fuzzy logic control of a heat exchanger system In *Fuzzy Automata and Decision Processes* (eds. Gupta MM, Saridis GN and Gaines BR). North-Holland, Amsterdam, pp. 285–320.

Østergaard JJ 1990 Fuzzy II: The new generation of high level kiln control. *Zement Kalk Gips (Cement-Lime-Gypsum)* **43**(11), 539–541.

Østergaard JJ 1996 High level control of industrial processes In *Proceedings of TOOLMET'96* (eds. Yliniemi L and Juuso E). University of Oulu, Control Engineering Laboratory, Linnanmaa, Oulu, pp. 1–12.

Palm R, Driankov D and Hellendoorn H 1997 *Model Based Fuzzy Control*. Springer.

Passino KM and Yurkovich S 1998 *Fuzzy Control*. Addison Wesley Longman, Inc.

Pedrycz W 1993 *Fuzzy Control and Fuzzy Systems*, 2nd edn. John Wiley & Sons.

Procyk TJ and Mamdani EH 1979 A linguistic self-organizing process controller. *Automatica* **15**, 15–30.

Qiao W and Mizumoto M 1996 PID type fuzzy controller and parameters adaptive method. *Fuzzy Sets and Systems* **78**, 23–35.

Ross T 1995 *Fuzzy Logic with Engineering Applications*. McGraw-Hill.

Rundqwist L 1991 Anti-reset windup for pid controllers In *Proceedings of the 11th Triennial World Congress of the International Federation of Automatic Control, IFAC* (eds. Jaakso U and Utkin VI). Pergamon Press, pp. 453–458.

Sala A, Guerra TM and Babuska R 2005 Perspectives of fuzzy systems and control. *Fuzzy Sets and Systems* **156**, 432–444.

Self K 1990 Designing with fuzzy logic. *IEEE Spectrum* **27**(11), 42–44, 105.

Siler W and Ying H 1989 Fuzzy control theory: The linear case. *Fuzzy Sets and Systems* **33**, 275–290.

Šiljak D 1968 *Nonlinear Systems: The Parameter Analysis and Design*. John Wiley & Sons.

Slotine JJE and Li W 1991 *Applied Nonlinear Control*. Prentice Hall.

Smith LC 1979 Fundamentals of control theory. *Chemical Engineering* **86**(22), 11–39. (Deskbook issue).

Stoll RR 1979 *Set Theory and Logic*, dover edn. Dover Publications, New York. (org 1963).

Sugeno M (ed.) 1985 *Industrial Applications of Fuzzy Control*. North-Holland.

Sugeno M, Murofushi T, Mori T, Tatematsu T and Tanaka J 1989 Fuzzy algorithmic control of a model car by oral instructions. *Fuzzy Sets and Systems* **32**(2), 207–219.

Takagi T and Sugeno M 1985 Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics* **15**(1), 116–132.

Tanaka K, Sano M and Suzuki K 1991 A new tuning method of fuzzy controllers *Proceedings of IFSA91*. IFSA, pp. 207–210.

Tso SK and Fung YH 1997 Methodological development of fuzzy-logic controllers from multivariable linear control. *IEEE Transactions on Systems, Man, and Cybernetics* **27**(3), 566–572.

Wang LX 1997 *A Course in Fuzzy Systems and Control*, international edn. Prentice Hall PTR.

Wenstøp F 1980 Quantitative analysis with linguistic values. *Fuzzy Sets and Systems* **4**(2), 99–115.

Yamakawa T and Miki T 1986 The current mode fuzzy logic integrated circuits fabricated by the standard CMOS process. *IEEE Transactions on Computers* **35**(2), 161–167.

Yamazaki T 1982 *An Improved Algorithm for a Self-Organising Controller and its Experimental Analysis*, PhD thesis, Queen Mary College, London Department of Electrical and Electronic Engineering.

Yamazaki T and Mamdani EH 1982 On the performance of a rule-based self-organizing controller *Proceedings of the IEEE Conference on Applications of Adaptive and Multivariable Control*, Hull.

Yasunobu S, Miyamoto S and Ihara H 1983 Fuzzy control for automatic train operation system *Proceedings of the International Congress on Control in Transportation Systems*. IFAC/IFIP/IFORS, Baden-Baden.

Yazdi H 1997 *Control and Supervision of Event-Driven Systems*, PhD thesis, Technical University of Denmark Department of Chemical Engineering. Also available from http://fuzzy.iau.dtu.dk/download/yazdi97.pdf [cited 12 Aug 2006].

Zadeh LA 1965 Fuzzy sets. *Information and Control* **8**, 338–353.

Zadeh LA 1973 Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics* **1**, 28–44.

Zadeh LA 1975 The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences* **8**, 43–80.

Zadeh LA 1984 Making computers think like people. *IEEE Spectrum*, **21**, 26–32.

Zadeh LA 1988 Fuzzy logic. *IEEE Computer* **21**(4), 83–93.

Ziegler J and Nichols N 1942 Optimum settings for automatic controllers. *Transactions of the American Society of Mechanical Engineers (ASME)* **64**, 759–768.

Ziegler J and Nichols N 1943 Process lags in automatic-control circuits. *Transactions of the American Society of Mechanical Engineers (ASME)* **65**, 433–444.

Zimmermann HJ 1993 *Fuzzy Set Theory – and its Applications*, 2nd edn. Kluwer Academic Publishers, Boston.

Zimmermann HJ (ed.) 1999 *Practical applications of fuzzy technologies*. *The Handbooks of Fuzzy Sets*. Kluwer Academic Publishers.

Index