A Primer on Fuzzy Control

Johansson, Mikael

1996

Document Version:
Publisher's PDF, also known as Version of record

Citation for published version (APA):
Johansson, M. (1996). *A Primer on Fuzzy Control*. (Techical Reports TFRT-7545). Department of Automatic Control, Lund Institute of Technology (LTH).

Total number of authors:
1

ISSN 0280–5316
ISRN LUTFD2/TFRT--7545--SE

A Primer on Fuzzy Control

Mikael Johansson

Department of Automatic Control
Lund Institute of Technology
January 1996

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
INTERNAL REPORT

Date of issue
January 1996

Document Number
ISRN LUTFD2/TFRT--7545--SE

Author(s)
Mikael Johansson

Supervisor

Sponsoring organisation
Institute of Applied Mathematics (ITM)

Title and subtitle
A Primer on Fuzzy Control

Abstract

The aim of this report is to present a well organized tutorial on fuzzy logic for control. The report has been written for an engineering audience, requiring only the most basic prerequisites in logic, mathematics and control. The report begins with a preliminary review of classical logic, and shows how fuzzy logic extends the concepts of classical logic. To apply fuzzy logic to engineering applications, interfaces have to be added around the fuzzy logic reasoning mechanism. This leads to the concept of fuzzy systems. From an input-output point-of-view, fuzzy systems are nonlinear mappings that can be interpreted as a set of fuzzy IF-THEN rules. With this as a basis, we show how fuzzy systems describe nonlinear mappings and derive closed form expressions for some classes of fuzzy system nonlinearities. We also discuss how fuzzy systems can be used for control.

Key words
Fuzzy Logic, Fuzzy Modelling, Function Approximation, Nonlinear Control, Fuzzy Control

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
61

Recipient's notes

Security classification

Contents

1. Preface

This report was part of the first progress report in the ITM project "Design and Tuning of Fuzzy Controllers Based on Nonlinear Control Theory". The aim of this report is to present a well organized tutorial on fuzzy logic for control. The report has been written for an engineering audience, requiring only the most basic prerequisites in logic, mathematics and control. It is the author's intention that a few hours of concentrated reading will leave the reader with a thorough understanding of fuzzy systems, and with some important insight in how fuzzy systems can be used for control. Although parts of the report are working manuscripts, we feel that the report constitutes an accessible and logical introduction to fuzzy logic for control, and as such, is important to publish.

This text owes much to valuable discussions with friends, students and colleagues. My supervisor Karl-Erik Årzén has been a constant source of support and encouragement during the writing of the report. Parts of this manuscript have been read by Jörgen Malmborg and Ulf Jönsson, whose comments and remarks have greatly improved the quality of this work. I would also like to thank Tan Kan of Hong Kong University of Science and Technology, whose enthusiasm over an early version of the text made the writing of this report a lot of fun.

The text starts out with a preliminary review of classical logic. In Chapter 2, we describe how fuzzy sets and fuzzy logic extend ideas of classical logic. To apply fuzzy logic to engineering applications such as control, interfaces have to be added around the fuzzy logic reasoning mechanism. This leads to the concept of fuzzy systems, described in Chapter 3. From a control engineer's point-of-view, fuzzy systems are nonlinear mappings parameterized so that we can interpret them in terms of fuzzy IF-THEN rules. In Chapter 4, we show how fuzzy systems describe nonlinear mappings, and how it is possible to derive closed form expressions of the fuzzy system mappings. In Chapter 5, we define a general architecture of the fuzzy controller and indicate how fuzzy systems can be used for control.

2. Preliminaries: Fundamentals of Logic

Aim: To review some basic concepts of mathematical logic

Logic is considered to be a subdivision of philosophy, and as such unfortunately not included in a traditional engineering education. It may therefore be of little help to say that fuzzy logic is a generalization of logic, and that most concepts in fuzzy logic are (more or less) intuitive generalizations of the concepts of logic. However, this short primer on logic is here to help the reader to get acquainted with the terminology and mathematics of logic. Hopefully, this will help the reader to penetrate the theory of fuzzy logic in an intuitive manner.

2.1. What is Logic ?

Logic is the scientific study of the process of reasoning. The ability to reason, or infer, is simply the ability to draw appropriate conclusions from given evidence. We can view the inference process as a process of going from what we know (the premises) to what we previously did not know (the conclusion).

In order for a computer to reason with logic, however, we must formulate the knowledge and the reasoning process into a form that is suitable for manipulations by a computer. The result is what is known as symbolic or mathematical logic – methods that seek to reduce reasoning to calculation.

In order to reason around a set of statements, we must first decide what kind of statements we are concerned with. This leads us to define propositions and predicates.

2.2. Propositions and Predicates

A *proposition* is simply a statement that is either true or false. The statements

$$2 \text{ is a prime number}$$
$$4 = 6$$
$$5 \geq 0$$

are all propositions (which are true, false and true respectively). The statement

$$x \geq 0$$

is not a proposition. It becomes a proposition, however, as soon as any number is substituted for x. Formulas like this are called *predicates*.

2.3. Sets and Characteristic Functions

In mathematical logic, a proposition is represented by a *set.* Recall that a set is a collection of elements that share a common property. We will refer to these elements as *members* of the set.

Clearly, a set is completely characterized by its members. The problem of characterizing a set is therefore a problem of specifying its members. This specification usually takes the form of a rule; "a set A is the set of all elements x in U that have the property P". We write

$$A = \{x \in U | \ x \text{ has the property } P\}$$

Another way to characterize a set is to use a *characteristic function*, $\mu_A(x)$, defined by

$$\mu_A(x) = \begin{cases} 1, & x \in A \\ 0, & \text{otherwise} \end{cases}$$

where we have introduced the *truth value* 1 for a true statement, and 0 for a false statement.

EXAMPLE 2.3.1 (SPECIFICATION OF A SET)
Consider the problem of characterizing the integer numbers greater than 2. The set A of integer numbers greater than 2 can thus be described by

$$A = \{x \in Z \mid x > 2\}$$

Alternatively, we can use the characteristic function

$$\mu_A(x) = \begin{cases} 1, & x > 2 \\ 0, & \text{otherwise} \end{cases}$$

This function is shown in Figure 2.1. □

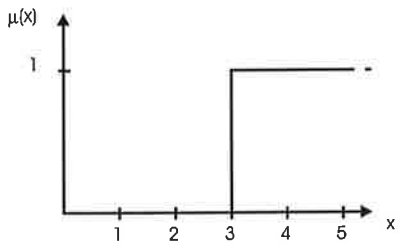


Figure 2.1 Characteristic function for the statement "$x > 2$".

There are three special sets;

1. the *universal set* (or *universe of discourse*), U, which contains all elements
2. the *empty set*, $\emptyset$, that contains no elements
3. the *singleton set*, δ, that contains only one element.

EXAMPLE 2.3.2 (UNIVERSE OF DISCOURSE)
The universe of discourse in Example 2.3.1 is the set of natural numbers, Z. □

2.4. Compound Propositions and Logic Formulas

Compound propositions are formed by combining simple propositions using the logic connectives "and", "or" and "not". In mathematical logic, these connectives are usually symbolized in the following way. If p and q are propositions,

$p \cap q$ symbolizes "p and q",

$p \cup q$ symbolizes "p or q" and

$\overline{p}$ symbolizes "not p".

It is important that the compound propositions are built in a way that makes sense. Such propositions are called *well-formed.* The well-formed propositions can be defined recursively in the following steps

(a) The truth values 1 and 0 are propositions.

(b) If p is a proposition, then $\overline{p}$ is also a proposition.

(c) If p and q are propositions, then $p \cap q$ and $p \cup q$ are propositions.

(d) The only propositions are those defined by (a)–(c).

EXAMPLE 2.4.1 (COMPOUND PROPOSITIONS)
The following two statements are propositions

$$\begin{array}{ll} (p \cap q) \cup r & \{\text{"}(p \text{ and } q) \text{ or } r\text{"}\} \\ p\bar{q} & \{\text{"}(p \text{ not } q)\text{"}\} \end{array}$$

However, only the first proposition is well-formed. □

Logic does not get really interesting until we introduce the *logic formulas.* Logic formulas are propositions or propositions combined using the connectives "implies" and "is equivalent to". If p and q are propositions, we write

$p \rightarrow q$ to symbolize "p implies q" and

$p \equiv q$ to symbolize "p is equivalent to q".

EXAMPLE 2.4.2 (LOGIC FORMULAS)
The statements

$$\begin{array}{ll} (p \cap q) \cup r & \{\text{"}(p \text{ and } q) \text{ or } r\text{"}\} \\ (p \cap (p \rightarrow q)) \rightarrow q & \{\text{ "}(p \text{ and } (p \text{ implies } q)) \text{ implies } q\text{" }\} \end{array}$$

are logic formulas. □

Well-formed logic formulas are defined analogous to well-formed propositions. Simply note that in addition to the well-formed formulas $p \cap q$ and $p \cup q$, we also have $p \rightarrow q$ and $p \equiv q$.

We summarize the five logic connectives in a *truth table*:

p	q	$p \cap q$	$p \cup q$	$\overline{p}$	$p \rightarrow q$	$p \equiv q$
0	0	0	0	1	1	1
0	1	0	1	1	1	0
1	0	0	1	0	0	0
1	1	1	1	0	1	1

The implication $p \rightarrow q$ defined in the truth table is called material implication, and its definition may at first glance seem strange. Note, however, that it is only in the case when p is true but q is false that we can conclude that p does not imply q. Equivalent logic formulations of the material implication thus include

$$\begin{aligned} p \rightarrow q \equiv & \quad \overline{p \cap \overline{q}} \\ p \rightarrow q \equiv & \quad \overline{p} \cup q \\ p \rightarrow q \equiv & \quad \overline{p} \cup (p \cap q) \end{aligned}$$

This can be verified using the truth table above.

2.5. Operations on Sets

This far we have seen that the atomic part of a statement is the proposition conveniently represented by a set. We have also seen how we can form compound propositions by connecting simple propositions using the logic connectives "and", "or" and "not". To derive the set representation of a compound proposition, we must define set operations corresponding to the logic connectives "and", "or" and "not". These operations are called *intersection*, *union* and *complement* respectively.

In the following, let A and B be two sets defined on the same universe of discourse, U. We can then define our three basic set operations as follows:

The *intersection* of A and B, written $A \cap B$, is defined by

$$A \cap B = \{x \in U \mid x \in A \text{ and } x \in B\}$$

The *union* of A and B, written $A \cup B$, is defined by

$$A \cup B = \{x \in U \mid x \in A \text{ or } x \in B \text{ or both}\}$$

The *complement* of A, written $\overline{A}$, is defined as

$$\overline{A} = \{x \in U \mid x \notin A\}$$

We can also derive rules for the computation of the characteristic function of compound propositions, as illustrated by the next example.

EXAMPLE 2.5.1 (CHARACTERISTIC FUNCTION FOR SET OPERATIONS)
It is easy to verify that the following formulas give the characteristic functions resulting from the set operations above.

$$\begin{aligned} \mu_{A \cap B}(x) = & \quad \min\{\mu_A(x), \mu_B(x)\} \\ \mu_{A \cup B}(x) = & \quad \max\{\mu_A(x), \mu_B(x)\} \\ \mu_{\overline{A}}(x) = & \quad 1 - \mu_A(x) \end{aligned}$$

□

To facilitate the discussion of interrelationships between elements, we need one more set operation: the cartesian product. The cartesian product is used to arrange the elements from two sets into one set of ordered pairs. More formally, the cartesian product of A and B (not necessarily defined on the same universe), written $A \times B$, is defined as

$$A \times B = \{(x, y) \mid x \in A \text{ and } y \in B\}$$

Note the word *ordered* pair; the order of the elements in the pair is important.

2.6. Rules and Reasoning

Having defined the basic characteristics and operations on sets, we can now turn our attention to the problem of knowledge representation and reasoning. We will use two concepts for the representation of knowledge; *rules* and *relations*. It is often natural to represent knowledge as implications, formulated as *IF-THEN rules*

$$\text{IF} < \underset{\textit{antecedent}}{\textit{proposition}} > \text{THEN} < \underset{\textit{consequent}}{\textit{proposition}} >$$

The proposition in the IF-part of the rule is called *antecedent* (what comes before) while the proposition in the THEN-part is called *consequent* (what comes after).

To reason about knowledge stated as IF-THEN rules, we must first formulate *inference rules*; rules that tell us how to manipulate logic formulas to obtain new formulas. There are eight basic inference rules, all derived from *tautologies*. A tautology is a logic statement that is always true.

EXAMPLE 2.6.1 (TAUTOLOGIES)
By simple manipulations of the truth-table, we can verify that the following logic formulas are tautologies

$$(p \cap (p \rightarrow q)) \rightarrow q$$
$$(\bar{q} \cap (p \rightarrow q)) \rightarrow \bar{p}$$
$$((p \rightarrow q) \cap (q \rightarrow r)) \rightarrow (p \rightarrow r)$$

□

From the tautologies in the example above, we can derive the following three *inference rules*:

Modus Ponens

Observation	:	x is A
Knowledge	:	IF x is A THEN y is B
Conclusion	:	y is B

Modus Tollens

Observation	:	y is not B
Knowledge	:	IF x is A THEN x is A
Conclusion	:	y is not B

Hypothetically Syllogism

Knowledge	:	IF x is A THEN y is B
Knowledge	:	IF y is B THEN z is C
Conclusion	:	IF x is A THEN z is C

2.7. Relations

Another way of representing knowledge is to use *relations*. Relations describe interrelationships between elements. The following statements

$$x \equiv y$$
$$x \rightarrow y$$
$$x \leq y$$

are all relations. Note especially that the implication $p \rightarrow q$ is a relation. To symbolize the predicate "x has the relation R to y", we write xRy.

As in the case of propositions, we represent the relation with a set. Recall that the Cartesian product forms a new set by arranging the elements of two sets into one set of ordered pairs. The set representing the relation is thus a subset of the Cartesian product of the sets representing x and y. In terms of the characteristic function, we have

$$\mu_{xRy}(x,y) = \begin{cases} 1, & xRy \\ 0, & \text{otherwise} \end{cases}$$

EXAMPLE 2.7.1 (A RELATION AND ITS RELATIONAL MATRIX)
Given two sets of numbers, $A = \{1,5,7\}$ and $B = \{2,6,10\}$ we can define the relation "a is greater than b". We then have $R = \{(5,2),(7,2),(7,6)\} \subset A \times B$. The characteristic function of a relation R is often shown as a *relational matrix*. In this example, we obtain the relational matrix

		B		
		2	6	10
	1	0	0	0
A	5	1	0	0
	7	1	1	0

□

For later use, we will define three operations that are useful when working with relations. These operations are the *composition*, *extension* and *projection* operations.

The *composition operation* is used to compose several relations into one, and can be defined as follows. The composition of two relations

$$R_1 \subseteq A \times B$$
$$R_2 \subseteq B \times C$$

written $R_1 \circ R_2$, are those elements (a,c) of $A \times C$ for which we can find an "intermediate" element $b \in B$ related to both a and c, i.e., aR_1b and bR_2c.

The extension and projection operations are useful when we want to manipulate the universe of a set. If A is a subset of the universe U, we define the *extension*, $e(A)$, of A into $U \times V$ as

$$e(A) = \{(u, v) \in U \times V \mid u \in A\}$$

or in terms of the characteristic function

$$\mu_{e(A)}(u, v) = \mu_A(u)$$

The extension operation has a clear geometric interpretation, as indicated in the Figure 2.2.

		V
	0	· · ·
U	1	· · ·
	1	· · ·

		V		
	0	0	0	0
U	1	1	1	1
	1	1	1	1

Figure 2.2 Extending $A \in U$ into the domain $U \times V$

If W is a set defined on the product space $U \times V$, we define the *projection*, $p(W)$, of W onto V as

$$p(W) = \{v \in V \mid (u, v) \in W\}$$

In terms of the characteristic function, we have

$$\mu_{p(W)}(v) = \max_{u \in U} \mu_W(u, v)$$

The geometric interpretation of the projection operation is illustrated in Figure 2.3

	V		
	·	·	·
	0	0	0
U	0	0	1
	0	1	1

	V		
	0	1	1
	0	0	0
U	0	0	1
	0	1	1

Figure 2.3 Projecting $W \in U \times V$ onto V.

2.8. Summary

In this chapter, we have made an informal review of some basic concepts from logic. The presentation is intentionally kept to a minimum, and the interested reader is referred to the references [Klenk, 1989], [Brown, 1990] for a more in-depth exposition of logic. However, the tiny proportion of ideas from logic is precisely what we need in order to make the fuzzy logic of control accessible and understandable.

3. From Logic to Fuzzy Logic

Aim: To explain the fuzzy set theory we need to develop a fuzzy controller

---

In the previous section, we surveyed some concepts and terminology from symbolic logic. As we will see shortly, this basic knowledge of logic will become very valuable when we in this section try to penetrate some of the more important concepts of fuzzy logic. Hopefully, this section will also make clear that fuzzy logic is not necessarily a "fuzzy" logic, but rather a logic for reasoning with fuzzy or vague statements.

3.1. Why Fuzzy Logic ?

Symbolic logic is a very elegant framework for reasoning with precise statements. Although elegancy is certainly a strength, the requirement that statements must be either true or false poses a strong restriction on the information binary logic can handle. Let us illustrate the discussion by a simple example.

EXAMPLE 3.1.1 (RESTRICTIONS OF BINARY LOGIC)
Consider the following two rules for driving a car at constant speed

IF *speed* is *too low*	THEN *apply more force to accelerator*
IF *speed* is *ok*	THEN *apply the same force to accelerator*
IF *speed* is *too high*	THEN *apply less force to accelerator*

For this to be a valid set of logic rules, we must specify sharp boundaries for when the speed of the car goes from being "*too low*" via "*ok*" to "*too high*". This is certainly possible, and could result in the modified rules

IF $v \leq v_{lim} - 5$	THEN $F_{acc} = F_{acc} + corr$
IF $v_{lim} - 5 < v < v_{lim} + 5$	THEN $F_{acc} = F_{acc}$
IF $v \geq v_{lim} + 5$	THEN $F_{acc} = F_{acc} - corr$

where v_{lim} denotes the current speed limit and $corr$ a correctional action.

Admittedly, the result of these modified rules is pretty far from the driving strategy most of us would adopt based on the original rules. □

Many everyday rules are stated in imprecise terms, just as in the example above. This kind of rules rely on a piece of common sense for their correct interpretation. Although no logic system has common sense in itself, it is clear that binary logic does not provide the methods for adequately describing the meaning of vague terms. In other words, we need a way of saying what we mean or rather, formalize the meaning of what we say.

This observation is the basic motivation for various extensions of logic and alternative approaches to reasoning. In this chapter, we will focus on how fuzzy logic approaches the problem of describing and reasoning with imprecise, or "fuzzy", information.

3.2. Fuzzy Sets and Membership Functions

The main problem with binary logic is the requirement that statements must be either true or false. When it comes to describing vague terms, it gets hard to say when a certain predicate goes from being false to true or vice versa. Fuzzy logic solves this problem by allowing the characteristic function to take any value in the interval $[0, 1]$, thus allowing predicates to move gradually from being false to being true.

In terms of sets, this means that the boundary separating elements of a set from elements not belonging to the set becomes gradual. We have the following definition:

DEFINITION 3.2.1 (FUZZY SET) *A fuzzy set, A, is a set whose characteristic function $\mu_A(x)$ takes values in the interval* $[0, 1]$. □

In fuzzy logic literature, the characteristic function of a fuzzy set is always called the *membership function* of the fuzzy set. We interpret the membership value of an element to a specific set as the degree to which the corresponding proposition applies.

Let us illustrate this discussion by returning to the problem of driving a car at constant speed, as presented in Example 3.1.1

EXAMPLE 3.2.1 (CONSTANT SPEED REVISITED)
Using the concept of a fuzzy set, we can try to specify what we mean when we say that the speed of the car is "*too low*", "*ok*", or "*too high*".

Similarly to the logical rules, we can say that keeping the speed limit is certainly "*ok*", and so, the speed $v = v_{lim}$ should have membership value 1 in the fuzzy set "*ok*". Large deviations from the speed limit are either "*too low*" or "*too high*". Consequently, speeds $v >> v_{lim}$ and $v << v_{lim}$ should have membership values 1 in the fuzzy sets "*too high*" and "*too low*" respectively. Typical membership functions are shown in Figure 3.1.

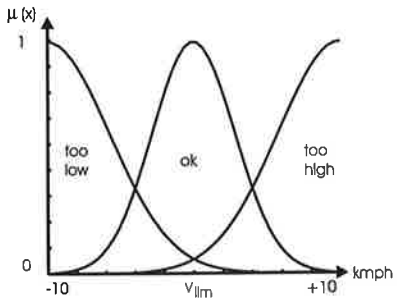


Figure 3.1 Membership functions for the predicates "*too low*", "*ok*" and "*too high*".

□

Although there are no theoretical restrictions on the shape of fuzzy sets, it is practical to use membership functions that are parameterized in a flexible way, using few parameters. We conclude this section by giving examples of the most common classes of membership functions

EXAMPLE 3.2.2 (TRAPEZOIDAL)
The trapezoidal membership function is described by

$$g(x) = \begin{cases} 0 & , x \leq x_l^- \\ \dfrac{x - x_l^-}{x_u^- - x_l^-} & , x_l^- < x \leq x_u^- \\ 1 & , x_u^- \leq x < x_u^+ \\ \dfrac{x_u^+ - x}{x_l^+ - x_u^+} & , x_u^+ \leq x < x_l^- \\ 0 & , x \geq x_l^+ \end{cases} \tag{3.1}$$

A few different trapezoidal membership functions are illustrated in Figure 3.2.2. □

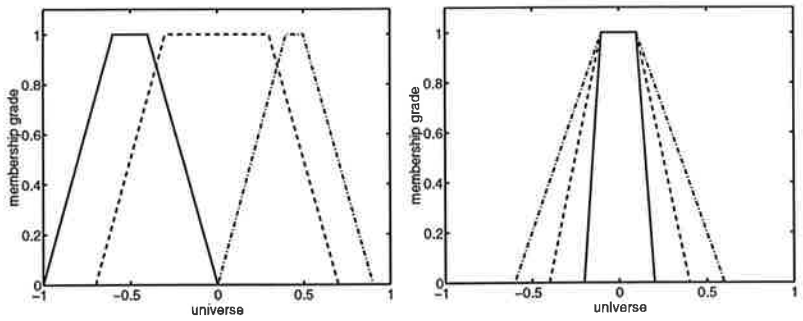


Figure 3.2 Trapezoidal membership functions with varying widths and centers (left), and varying slopes (right).

EXAMPLE 3.2.3 (TRIANGULAR)
The triangular membership function is described by

$$g(x) = \begin{cases} 0 & , x \leq x_l^- \\ \dfrac{x - x_l^-}{x_u - x_l^-} & , x_l^- < x \leq x_u \\ \dfrac{x_u - x}{x_l^+ - x_u} & , x_u \leq x < x_l^- \\ 0 & , x \geq x_l^+ \end{cases} \tag{3.2}$$

Note that the triangular membership function is a special case of the trapezoidal function, obtained by setting $x_u^- = x_u^+ = x_u$ in (3.1). Some triangular membership functions are illustrated in Figure 3.2.3. □

EXAMPLE 3.2.4 (GAUSSIAN)
A gaussian membership function is described by

$$g(x) = \exp\left(-\left(\frac{x - \bar{x}}{\sigma}\right)^2\right) \tag{3.3}$$

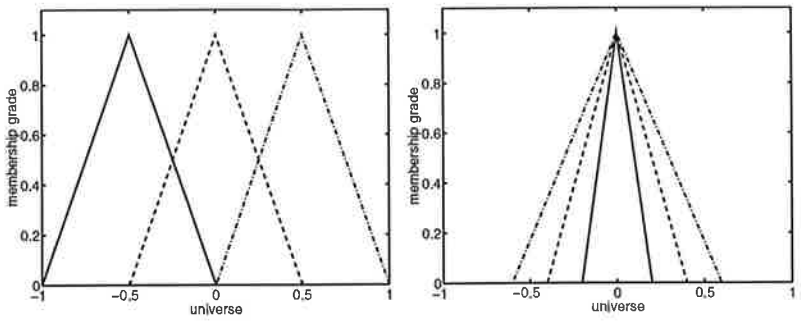


Figure 3.3 Some triangular membership function with varying centers (left) and slopes (right).

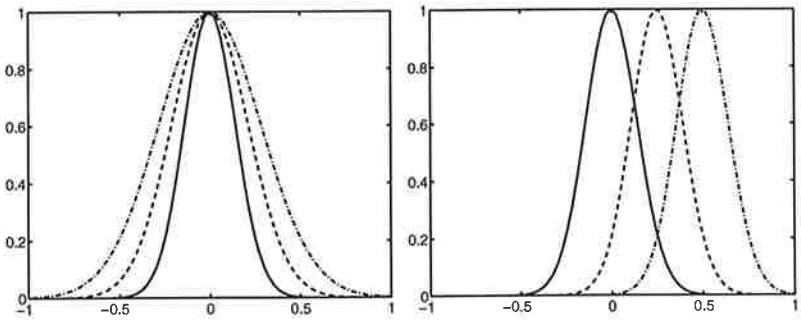


Figure 3.4 Gaussian membership functions for different σ:s (left) and different $\bar{x}$:s (right).

Some examples of the gaussian membership function are illustrated in Figure 3.4.

Note that $\bar{x}$ determines the center of the membership function, whereas σ controls the width of the function. □

EXAMPLE 3.2.5 (SIGMOID)
A sigmoid membership function is described by

$$g(x) = \frac{1}{1 + \exp\left(-\alpha\left(x - \bar{x}\right)\right)} \tag{3.4}$$

Some sigmoid functions are illustrated in Figure 3.5. Note that $\bar{x}$ determines

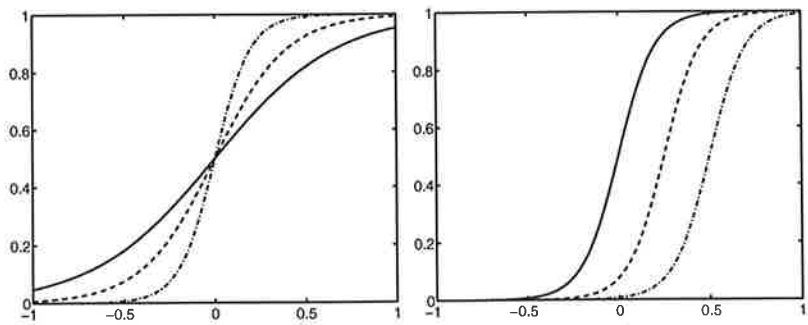


Figure 3.5 Gaussian membership functions for different α:s (left) and different $\bar{x}$:s (right).

the cross-over point of the membership function (the element whose member-

ship value equals 0.5), and α controls the slope of the function at this point. □

EXAMPLE 3.2.6 (SINGLETON)
The singleton membership function is described by

$$g(x) = \begin{cases} 1 & , x = \bar{x} \\ 0 & , \text{otherwise} \end{cases} \tag{3.5}$$

This function is illustrated in Figure 3.6. □

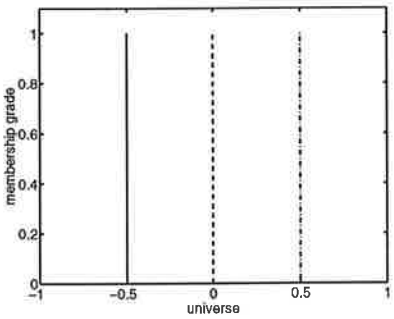


Figure 3.6 Membership functions for singleton fuzzy sets.

3.3. Fuzzy Predicates and Fuzzy Propositions

Similar to binary logic, we can talk about *fuzzy predicates* and *fuzzy propositions*. By fuzzy propositions, we mean vague statements to which we can assign a truth-value (now in the interval $[0, 1]$.) Vague statements for which we can not assign a particular truth-value are called fuzzy predicates. The statements

Mary is tall

X is close to zero

The car drives at high speed

are are all fuzzy predicates. However, if we measure Mary's height and evaluate the membership function describing our notion of "tall", we can assign a truth-value to the statement "Mary is tall". This statement can now be regarded as a fuzzy proposition.

Compound fuzzy propositions are formed by combining simple fuzzy propositions using the connectives "and", "or" and "not". Similar to binary logic, the fuzzy set representing a compound fuzzy proposition is formed using the fuzzy set operations intersection, union and complement respectively. How these fuzzy set operations are defined is the topic of the next section.

3.4. Fuzzy Set Operations

As we move from the classical sets to fuzzy sets, we must decide how to extend the set operations intersection, union and complement to fuzzy sets. The most stringent way to do this is to state a set of axioms that the set operation must satisfy in order to

(i) extend the classical set operations, and

(ii) produce an intuitively correct result on fuzzy sets.

This leads to the definition of the T-norm and S-norm.

It is important to notice that a compound set is formed by applying the set operations point-wise over all elements on the universe of discourse. To emphasize this, we will discuss the set-operations in terms of $a = \mu_A(x)$, $b = \mu_B(x)$ and $c = \mu_C(x)$. That is, for an arbitrary element x, a, b and c are its membership values in the fuzzy sets A, B and C respectively.

We are now ready for the following definitions:

DEFINITION 3.4.1 (T-NORM) *A T-norm, denoted $\star$, is an intersection operation on fuzzy sets that satisfies the following four axioms*

T.1	$a \star 0 = 0\,,\ a \star 1 = a$	*(Boundary Condition)*
T.2	$a \star b = b \star a$	*(Commutativity)*
T.3	$a \leq b \implies a \star c \leq b \star c$	*(Nondecreasing)*
T.4	$(a \star b) \star c = a \star (b \star c)$	*(Associativity)*

□

DEFINITION 3.4.2 (S-NORM) *An S-norm, denoted $\dot{+}$, is a union operation on fuzzy sets that satisfies the following four axioms*

S.1	$a\dot{+}1 = 1\,,\ a\dot{+}0 = a$	*(Boundary Condition)*
S.2	$a\dot{+}b = b\dot{+}a$	*(Commutativity)*
S.3	$a \leq b \implies a\dot{+}c \leq b\dot{+}\,c$	*(Nondecreasing)*
S.4	$(a\dot{+}b)\,\dot{+}c = a\dot{+}\,(b\dot{+}c)$	*(Associativity)*

□

DEFINITION 3.4.3 (COMPLEMENT) *The complement operation on a fuzzy set, denoted $\bar{}$, satisfies the following axioms*

C.1	$\bar{0} = 1\,,\ \bar{1} = 0$	*(Boundary Condition)*
C.2	$a \leq b \implies \bar{b} \leq \bar{a}$	*(Nondecreasing)*

□

Since the fuzzy set operations are not unique, we can create the union, intersection and complement of fuzzy sets in many ways. Some of the most common operations are defined in the following examples.

EXAMPLE 3.4.1 (INTERSECTION OPERATORS)
The intersection of two fuzzy sets is often formed using one of the following T-norm operations:

Algebraic Product:	$a \star b = ab$
Minimum:	$a \star b = \min(a, b)$
Drastic Product	$a \star b = \begin{cases} \text{a} & \text{if b} = 1 \\ \text{b} & \text{if a} = 1 \\ 0 & \text{otherwise} \end{cases}$
Einstein Product:	$a \star b = \dfrac{ab}{2 - (a + b - ab)}$

□

EXAMPLE 3.4.2 (UNION OPERATIONS)
The union of two fuzzy sets is often formed using one of the following S-norm operations

$$\begin{aligned}
&\text{Algebraic Sum:} && a\dot{+}b = a + b - ab \\
&\text{Maximum:} && a\dot{+}b = \max(a, b) \\
&\text{Drastic Sum:} && a\dot{+}b = \begin{cases} \text{a} & \text{if b} = 0 \\ \text{b} & \text{if a} = 0 \\ 1 & \text{otherwise} \end{cases} \\
&\text{Einstein Sum:} && a\dot{+}b = \frac{a+b}{1+ab} \\
&\text{Bounded Sum:} && a\dot{+}b = \min(1, a+b)
\end{aligned}$$

□

EXAMPLE 3.4.3 (THE COMPLEMENT OPERATION)
The complement of a fuzzy set is formed using the following operation

$$\text{Fuzzy Complement} \quad \bar{a} = 1 - b$$

□

We conclude this section by an example of fuzzy set operations.

EXAMPLE 3.4.4 (FUZZY SET OPERATIONS)
Returning to the car example, we can now define membership functions for the statements "*ok* OR *too high*", "*ok* AND *too high*" and "NOT *ok*". We have

$$\begin{aligned}
\mu_{ok \text{ OR } too\ high}(v) &= \mu_{ok}(v)\dot{+}\mu_{too\ high}(v) \\
\mu_{ok \text{ AND } too\ high}(v) &= \mu_{ok}(v) \star \mu_{too\ high}(v) \\
\mu_{\text{NOT } ok}(v) &= 1 - \mu_{ok}(v)
\end{aligned}$$

These membership functions are shown in Figure 3.7, using min as T-norm and max as S-norm. □

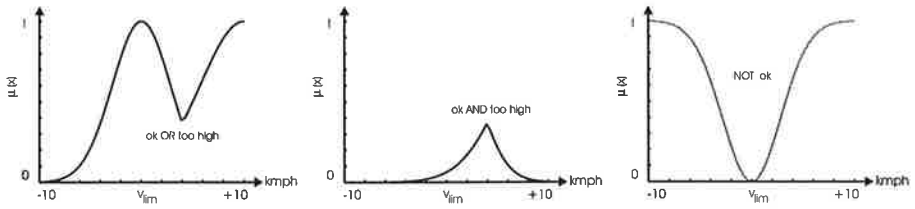


Figure 3.7 The fuzzy set operations intersection, union and complement demonstrated on the car example.

3.5. Linguistic Variables and Fuzzy IF-THEN Rules

Similar to the implications used in binary logic, we can represent vague knowledge as fuzzy IF-THEN rules. A fuzzy IF-THEN rule takes the form

$$\text{IF} < \textit{fuzzy proposition} > \text{THEN} < \textit{fuzzy proposition} >$$

The variables occurring in the fuzzy propositions are called *linguistic variables*. Introducing linguistic variables is a way to stress the conceptual difference between a numerical variable and a variable that takes words as its values. Formally, a linguistic variable is defined as follows:

DEFINITION 3.5.1 (LINGUISTIC VARIABLE) *If a variable takes words as its values, we call it a linguistic variable. A linguistic variable is characterized by* (N, V, U, S) *where*

- N *is the name of the variable*
- V *is the set of linguistic values the variable can take*
- U *is the universe of discourse for the actual physical variable*
- S *is a semantic rule assigning a fuzzy set in* U *to each linguistic value in* V

□

Let us apply this definition on the rules for driving a car at constant speed.

EXAMPLE 3.5.1 (CONSTANT SPEED, 3RD TRY)
To describe the speed of the car, we have used the linguistic variable "*speed*", which we can characterize as follows

- N The name of the variable is "*speed*"
- V The speed can take the linguistic values {"*too low*", "*ok*", "*too high*"}
- U The speed of the car is limited to $[0, 180]$ kmph.
- S The semantic meaning of "*too low*", "*ok*" and "*too high*" are given by the fuzzy sets $\mu_{too\ low}(v)$, $\mu_{ok}(v)$ and $\mu_{too\ high}(v)$ respectively.

□

Similar to classical logic rules, fuzzy IF-THEN statements are interpreted as implications. When defining the implication operation for fuzzy sets, two different solutions have been suggested in the literature. The most obvious one is simply to extend the material implication used in binary logic, i.e.

$$p \rightarrow q = q \cup \bar{p} \tag{3.6}$$

or equivalently

$$p \rightarrow q = (p \cap q) \cup \bar{p} \tag{3.7}$$

This is how the implication operation is usually defined in multi-valued logic, and depending on what S-norm and T-norm operations we choose we can define a broad range of fuzzy implication operations. We illustrate this by an example

EXAMPLE 3.5.2 (IMPLICATIONS)
Using max for fuzzy union in (3.6), we get the Dienes-Rescher implication

$$\mu_{p \rightarrow q}(x, y) = \max\left[\mu_q(y), (1 - \mu_p(x))\right]$$

Using max for union and min for intersection in (3.7), we get the Zadeh-implication

$$\mu_{p \rightarrow q}(x, y) = \max\left[\min\left(\mu_p(x), \mu_q(y)\right), 1 - \mu_p(x)\right]$$

□

In fuzzy control literature, however, the fuzzy IF-THEN rules are often interpreted as

$$\text{IF} < \textit{fuzzy proposition} > \text{THEN} < \textit{fuzzy proposition} > \text{ELSE} < \textit{void} >$$

where the consequent ELSE $< \textit{void} >$ means that if the fuzzy proposition of the antecedent evaluates to zero, this rule should be ignored. In symbolic logic, we can write

$$p \rightarrow q = p \cap q \tag{3.8}$$

In fuzzy logic, the AND operation $\cap$ is performed by a T-norm operation. This leads to the Mamdani implications, as illustrated by the next example.

EXAMPLE 3.5.3 (MAMDANI IMPLICATIONS)
By using algebraic product as T-norm in (3.8), we get the implication

$$\mu_{p \rightarrow q}(x, y) = \mu_p(x) \cdot \mu_q(y)$$

Choosing min for fuzzy intersection, we get the implication

$$\mu_{p \rightarrow q}(x, y) = \min\{\mu_p(x), \mu_q(y)\}$$

□

Although the approaches clearly lead to different implication operations, it can be verified that they produce the same result for binary logic when applied to the modus ponens inference rule.

In the next section, we will see how we can perform reasoning with fuzzy IF-THEN rules using a generalized version of modus ponens.

3.6. Approximate Reasoning

We now face the problem of extending the inference mechanisms of binary logic to fuzzy logic. As we have seen earlier, classical logic provides a large number of inference rules. What inference rule to use depends on how the rules are stated and what information is available.

In direct controllers, we use controller rules and measurements to infer a control action. For this purpose, modus ponens is clearly the most appropriate inference rule:

Observation	:	*input* is A
Knowledge	:	IF *input* is A THEN *output* is B
Conclusion	:	*output* is B

We will therefore focus on extending modus ponens to a *generalized modus ponens*. Intuitively, a generalized version of modus ponens would infer according to

Observation	:	*input* is A'
Knowledge	:	IF *input* is A THEN *output* is B
Conclusion	:	*output* is B'

where "the closer the A' to the A, the closer the B' to the B". Note that in this case, it is very natural to view the fuzzy IF-THEN rule as a fuzzy relation, relating the value of the output to the value of the input.

The generalized inference rules are all derived from the *compositional rule of inference.* The compositional rule of inference is a generalization of the following familiar procedure:

Suppose that we have a curve $y = f(x)$ describing the relation between x and y. When given $x = a$, we can infer that $y = f(a) = b$. This is illustrated in Figure 3.8.

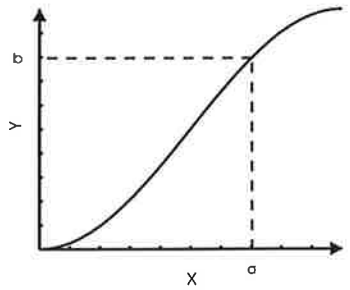


Figure 3.8 Graphical procedure for determining that $y = f(a) = b$.

This procedure can be generalized to the case where $f(x)$ is an interval-valued function. Given an interval a in X, we can infer the interval b to which y belongs using the following method, as illustrated in Figure 3.9.

(1) Construct the cyclic extension of a into $X \times Y$.

(2) Find the intersection, I, with the interval valued curve.

(3) Project I onto the y-axis to obtain $y = b$.

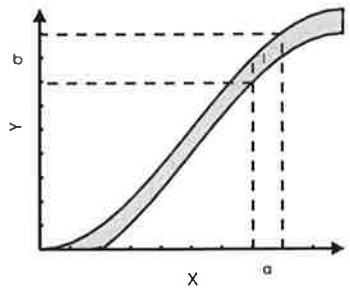


Figure 3.9 Graphical procedure for determine the interval to which y belongs when x and f(x) are interval valued.

Now assume that R is a fuzzy relation on $X \times Y$ and that A' is a fuzzy set in X. Obtaining a fuzzy set B' in Y is then straightforward:

(1) Construct the cyclic extension $c(A')$ of A' into $X \times Y$

$$\mu_{c(A')}(x, y) = \mu_{A'}(x)$$

(2) Find the intersection, I, with the relation R

$$\mu_I(x, y) = \mu_{c(A') \cap R}(x, y) = \mu_{A'}(x) \star \mu_R(x, y)$$

(3) Project I onto Y to obtain a fuzzy set B' in Y

$$\mu_{B'}(y) = \sup_{x \in X} [\mu_I(x, y)] = \sup_{x \in X} [\mu_{A'}(x) \star \mu_R(x, y)]$$

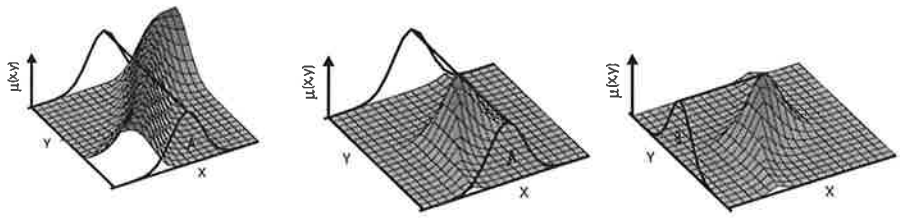


Figure 3.10 Graphical procedure for determining $\mu_{B'}(x)$ from the observation $\mu_{A'}(x)$ of x and the fuzzy relation $\mu_R(x, y)$: Form the extension of A, determine the intersection of A and R and project this intersection to obtain B.

We summarize this discussion in the following definition:

DEFINITION 3.6.1 (COMPOSITIONAL RULE OF INFERENCE) *Given a fuzzy set A' in X and a relation R in $X \times Y$, the compositional rule of inference infers a fuzzy set B' in Y as*

$$\mu_{B'}(y) = \sup_{x \in X} [\mu_{A'}(x) \star \mu_R(x, y)]$$

□

The generalized modus ponens is a special case of this procedure, where the fuzzy relation R is an implication $A \rightarrow B$.

EXAMPLE 3.6.1 (GENERALIZED MODUS PONENS)
Given a fuzzy set A' in X and the fuzzy IF-THEN rule

$$\text{IF } x \text{ IS } A \text{ THEN } y \text{ IS } B$$

the generalized modus ponens infers the fuzzy set B' given by

$$\mu_{B'}(y) = \sup_{x \in X} [\mu_{A'}(x) \star \mu_{A \rightarrow B}(x, y)]$$

□

Often, the fuzzy set A' is a fuzzy singleton. In this case, the generalized modus ponens simplifies considerably, as illustrated by the next example

EXAMPLE 3.6.2 (SIMPLIFIED GENERALIZED MODUS PONENS)
If the fuzzy set A' is a fuzzy singleton, we can simplify the generalized modus ponens. Since $\mu_{A'}(x) = 0$ if $x \neq \bar{x}$ and $1 \star a = a$, we have

$$\mu_{B'}(y) = \sup_{x \in X} [\mu_{A'}(x) \star \mu_{A \rightarrow B}(x, y)] = \mu_{A \rightarrow B}(\bar{x}, y)$$

If we use Mamdani implications, we get

$$\mu_{B'}(y) = \mu_A(\bar{x}) \star \mu_B(y)$$

□

3.7. Summary

This far we have seen how fuzzy set theory tries to capture the meaning of vague statements using the concept of a fuzzy set. Essentially, we give vague statements a precise meaning by stating their membership function. With fuzzy sets as starting point, fuzzy logic is simply an extension of logic to enable reasoning with information stated by fuzzy sets. Similar to logic, we can connect several fuzzy propositions into one compound proposition. Likewise, the membership function representing the compound fuzzy proposition is obtained through fuzzy set operations.
Reasoning with information stated as fuzzy sets is known as approximate reasoning. The compositional rule of inference, which we have motivated using a graphical technique, is a general rule from which we can derive fuzzy counterparts of the logic inference rules.

This far, we have only considered reasoning with one rule. We will now turn our attention to fuzzy systems, in which the rule base typically consists of several rules. For more material on fuzzy sets and fuzzy logic, the reader is referred to [Zadeh, 1965], [Lee, 1990], [Driankov and M.Reinfrank, 1993], [Dubois and Prade, 1980] and [Pedrycz, 1989].

4. From Fuzzy Logic to Fuzzy Systems

Aim: To describe the fuzzy systems found in most fuzzy controllers.

At this stage, we have developed a basic understanding of fuzzy logic and approximate reasoning. The essential difference between fuzzy logic and binary logic is that propositions no longer have to be completely true or false but can be fulfilled to any degree. This feature is captured by the introduction of fuzzy sets, whose main characteristic is a membership function which now can take values in the interval $[0, 1]$. Fuzzy logic and approximate reasoning is then concerned with extending classical sets and reasoning to apply for fuzzy sets in an intuitive manner.

In most applications, however, the inputs and outputs of a fuzzy logic system are not fuzzy sets, but rather numeric values. The inputs of a fuzzy controller, for instance, are numeric sensor measurements. Similarly, the output of a fuzzy controller is often an actuator command, and must therefore also be a numeric value. In order to apply fuzzy logic and approximate reasoning to these problems, we have to add interfaces between the fuzzy inference engine and the environment, as illustrated in Figure 4.1. The conversion from a numeric value to a fuzzy set is called *fuzzification*. The conversion from a fuzzy set to a numeric value is called *defuzzification*.

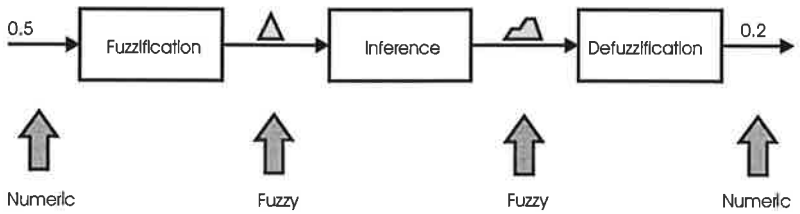


Figure 4.1 The fuzzification and defuzzification interfaces enable approximate reasoning to be applied to numeric data processing.

Without restriction, we will only consider fuzzy systems with one output.

In this chapter, we discuss the four components of a fuzzy system; the knowledge base, the fuzzifier, the inference engine and the defuzzifier. We put the pieces together to form a fuzzy system. This type of fuzzy system is called a Mamdani-type fuzzy system. We finally introduce a somewhat different type of fuzzy system called Takagi-Sugeno fuzzy system.

4.1. Describing Process Knowledge – The Knowledge Base

A fuzzy system knowledge base consists of fuzzy IF-THEN rules and membership functions characterizing the fuzzy sets. At this stage, we do not restrict the membership function shapes. However, we choose to limit our attention

to rules using only the AND connective. We thus define the fuzzy system rule base as follows.

DEFINITION 4.1.1 (FUZZY RULE BASE) *A fuzzy system rule base, $\mathcal{R}$, is a set of rules on the form*

$$\mathcal{R}^{(l)}: \text{IF } x_1 \text{ IS } A_1^{(l)} \text{ AND } \ldots \text{ AND } x_n \text{ IS } A_n^{(l)} \text{ THEN } y \text{ IS } B^{(l)} \tag{4.1}$$

where $A_i^{(l)}$ and $B^{(l)}$ are labels for the fuzzy sets characterized by the membership functions $\mu_{A_i^{(l)}}(x_i)$ and $\mu_{B^{(l)}}(y)$ respectively. □

In binary logic, logic formulas are sometimes illustrated using a Karnaugh map. Similarly, it is sometimes fruitful to view the fuzzy system rule base as a table, as illustrated in the next example.

EXAMPLE 4.1.1 (FUZZY PD CONTROLLER RULES)
Consider the following nine rules describing a fuzzy PD controller:

IF e IS NL AND $\dot{e}$ IS NL	THEN u IS	NL
IF e IS ZE AND $\dot{e}$ IS NL	THEN u IS	NS
IF e IS PL AND $\dot{e}$ IS NL	THEN u IS	ZE
IF e IS NL AND $\dot{e}$ IS ZE	THEN u IS	NS
IF e IS ZE AND $\dot{e}$ IS ZE	THEN u IS	ZE
IF e IS PL AND $\dot{e}$ IS ZE	THEN u IS	PS
IF e IS NL AND $\dot{e}$ IS PL	THEN u IS	ZE
IF e IS ZE AND $\dot{e}$ IS PL	THEN u IS	PS
IF e IS PL AND $\dot{e}$ IS PL	THEN u IS	PL

where we have introduced the labels NL, NS, ZE, PS and PL to mean *Negative Large*, *Negative Small*, *Zero*, *Positive Small* and *Positive Large* respectively.

Although there are only nine rules in this rule base, it is already hard to figure out the operation of this controller. In this case, we can enhance readability by presenting the rules in the table

		$\dot{e}$		
		NL	ZE	PL
e	PL	ZE	PS	PL
	ZE	NS	ZE	PS
	NL	NL	NS	ZE

In some literature, this kind of rule table is called a fuzzy associative memory (FAM). □

In the same way as we require the individual rules to be "well posed", it is often natural to put requirements on the knowledge base. For instance, we may require that at least one rule applies for each possible system input. It is also reasonable to require that there are no conflicting rules, in the sense that they have the same IF-part but different THEN-parts. This leads to the following definitions.

DEFINITION 4.1.2 (PROPERTIES OF THE KNOWLEDGE BASE) *A knowledge base is called complete if for each input, at least one rule applies, i.e.*

$$\forall \boldsymbol{x}\ \exists\, l : \mu_{A^{(l)}}(\boldsymbol{x}) \neq 0$$

A knowledge base is called an information system if the membership values of the rule antecedents always sum to one, i.e.

$$\sum_{l=1}^{M} \mu_{A^{(l)}}(\boldsymbol{x}) = 1,\ \forall \boldsymbol{x}$$

A rule base is called consistent if no two rules have the same IF-parts but different THEN-parts. □

We can illustrate these properties on the fuzzy PD-example.

EXAMPLE 4.1.2 (PROPERTIES OF THE FUZZY PD RULE BASE)
If we define NL, ZE and PL for e and $\dot{e}$ using the membership functions in Figure 4.2, we can discuss the properties of the fuzzy PD rule base.

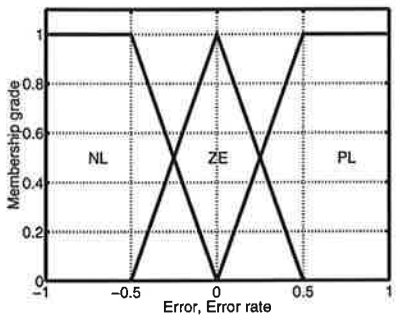


Figure 4.2 Membership functions for the fuzzy PD controller inputs.

Firstly, the rule base is consistent, since there is only one entry in each slot of the rule table. The knowledge base is also complete, since the membership functions cover all input values and there are entries in every slot of the rule table. If we use product T-norm, it can be verified that the knowledge base is also an information system. □

We are now ready to describe the three stages of the fuzzy system computations.

4.2. From Numeric to Fuzzy Set – Fuzzification

The first stage in the fuzzy system computations is to transform the numeric inputs into fuzzy sets. This operation is called fuzzification. Unfortunately, the theory of fuzzy sets and approximate reasoning does not aid us in the design of a fuzzifier. However, the following definition seems reasonable.

DEFINITION 4.2.1 (FUZZIFIER) *A Fuzzifier, $\mathcal{F}$, converts a numeric value, $\boldsymbol{x}^*$ into a fuzzy set $\mu_{A'}(\boldsymbol{x})$. We require that the fuzzy set $\mu_A(\boldsymbol{x})$ fulfills the following axioms*

F1. $\mu_A(\boldsymbol{x}^*) = 1$

F2. $\mu_A(\cdot)$ *is a decreasing function of* $||x - x^*||$.

□

The following fuzzifiers have been suggested in the literature.

EXAMPLE 4.2.1 (SINGLETON)
A *Singleton Fuzzifier* maps a numeric value x^* into the fuzzy singleton (3.5) with $\bar{x} = x^*$. □

EXAMPLE 4.2.2 (GAUSSIAN)
A *Gaussian Fuzzifier* maps a numeric value x^* into a the gaussian membership function (3.3) with $\bar{x} = x^*$. □

EXAMPLE 4.2.3 (TRIANGULAR)
A *Triangular Fuzzifier* maps a numeric value x^* into the triangular membership function (3.2) with $x_u^- = x_u^+ = x^*$. □

The fuzzifiers are illustrated in Figure 4.3.

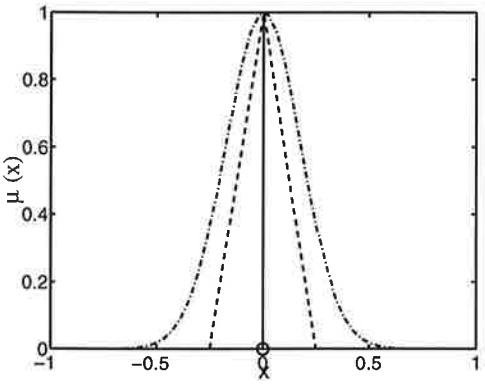


Figure 4.3 The suggested fuzzifiers for $x^* = 0$

The singleton fuzzifier is predominant in fuzzy control literature. The main reason for this is that this choice of fuzzifier simplifies the computations in the fuzzy system considerably. It is hard to motivate the use of the other fuzzifiers.

4.3. Computing a Fuzzy Output – Inference

When the numeric inputs of the fuzzy system have been transformed into appropriate fuzzy sets, the inference engine can be applied to infer the output of the system. From the point-of-view of fuzzy set theory, the inference engine is the heart of the fuzzy system. It is the inference engine that performs all fuzzy logic manipulations in a fuzzy system.

In the previous chapter, we explained how approximate reasoning could be used to infer using a single rule. Since a rule base generally consists of several rules, we must decide how to reason with multiple rules. There are two different approaches: composition based inference and individual rule based inference.

In *Composition Based Inference*, the complete rule base is treated as one fuzzy relation. This fuzzy relation is obtained as the combination of the relations describing the individual rules, i.e.

$$\mu_R(x, y) = \mu_{R(1)}(x, y) \dot{+} \ldots \dot{+} \mu_{R(M)}(x, y)$$

The relation describing the rule base is then treated as one single rule, exactly as in Chapter 2.

In *Individual Rule Based Inference*, as its name suggests, we treat each rule individually. From each rule, we infer an output fuzzy set, and the output fuzzy set resulting from all rules is a obtained by combining all the individual fuzzy sets.

$$\mu_{B'}(y) = \mu_{B'(1)}(y) \dot{+} \ldots \dot{+} \mu_{B'(M)}(y)$$

Incidentally, the two approaches coincide for some choices of inference engine parameters. We shall only be concerned with fuzzy systems that use individual rule based inference and Mamdani implication. This class of inference engines can be defined as follows.

DEFINITION 4.3.1 (FUZZY SYSTEM INFERENCE ENGINE) *A fuzzy system inference engine, $\mathcal{I}$, performs the following fuzzy system computations on the rule base $\mathcal{R}$ of Definition 4.1.1.*

1. *For the l^{th} rule, the fuzzy set describing the rule premise is formed using a T-norm operator*

$$\mu_{A^{(l)}}(x) = \mu_{A_1^{(l)}}(x_1) \star \cdots \star \mu_{A_n^{(l)}}(x_n)$$

2. *Using a (possibly different) T-norm as implication operation, the fuzzy relation defined by the rule is created*

$$\mu_{A^{(l)} \to B^{(l)}}(x, y) = \mu_{A^{(l)}}(x) \star \mu_{B^{(l)}}(y)$$

3. *The generalized modus ponens is applied to infer the output suggested by this rule*

$$\mu_{B'(l)}(y) = \sup_x \{\mu_{A'}(x) \star \mu_{A^{(l)} \to B^{(l)}}(x, y)\}$$

4. *The total output is obtained through aggregation of the individual rule outputs using an S-norm operation*

$$\mu_{B'}(y) = \mu_{B'(1)}(y) \dot{+} \ldots \dot{+} \mu_{B'(M)}(y)$$

□

Although there are a number of possible inference engines of this class, only two combinations are widely used in fuzzy control. These are the Product-Sum and Min-Max inference engines, as described in the examples below.

EXAMPLE 4.3.1 (PRODUCT-SUM INFERENCE ENGINE)
In the Product-Sum inference engine, we use individual rule based inference, algebraic product as T-norm, bounded sum as S-norm and algebraic product as implication. □

EXAMPLE 4.3.2 (MIN-MAX INFERENCE ENGINE)
In a Min-Max inference engine, we use individual rule based inference, product as T-norm, max as S-norm and min as implication. □

4.4. From Fuzzy Set to Numeric Value – Defuzzification

As we have seen from the previous section, the result of the inference process is an output represented by a fuzzy set (denoted $\mu_{B'}(y)$). As discussed initially in this chapter, the output of the fuzzy system should be a numeric value. The transformation of a fuzzy set into a numeric value is called *defuzzification*, which we choose to define in the following axiomatic way

DEFINITION 4.4.1 (DEFUZZIFIER) *A Defuzzifier, $\mathcal{D}$, maps a fuzzy set $\mu_{B'}(y)$ into a numeric value y^*. We require the defuzzification operation to fulfill the following axiom*

D1. y^* lies in the interval of support of $\mu_{B'}(y)$.

□

Several defuzzification operations have been suggested in the literature. We choose to give examples of the most common ones.

EXAMPLE 4.4.1 (CENTER OF GRAVITY)
The Center of Gravity (COG) defuzzifier specifies the numeric value to be the center of gravity of the fuzzy set, i.e.

$$y^* = \frac{\int_V y \cdot \mu_{B'}(y)\, dy}{\int_V \mu_{B'}(y)\, dy}$$

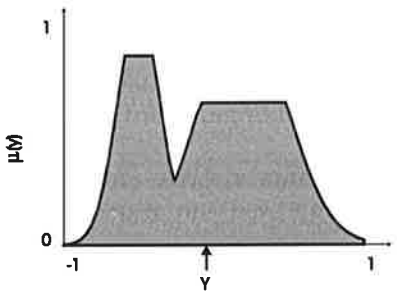


Figure 4.4 Center of Gravity defuzzifification of the fuzzy set B'.

□

EXAMPLE 4.4.2 (CENTER AVERAGE)
The Center Average (CA) defuzzifier specifies the numeric value to be the weighted average of the centers, $\bar{y}^{(l)}$ of the individual consequent fuzzy sets, i.e.

$$y^* = \frac{\sum_{l=1}^{M} \bar{y}^{(l)} \cdot w^{(l)}}{\sum_{l=1}^{M} w^{(l)}}$$

The weights, $w^{(l)}$ are equal to the height of the respective fuzzy sets.

□

EXAMPLE 4.4.3 (MAXIMUM)
Maximum defuzzifiers determines y^* as a point in the interval I_M where the fuzzy set has its maximum value

$$I_M = \{y \in V \mid \mu_{B'}(y) = \sup_{y \in V} \mu_{B'}(y)\}$$

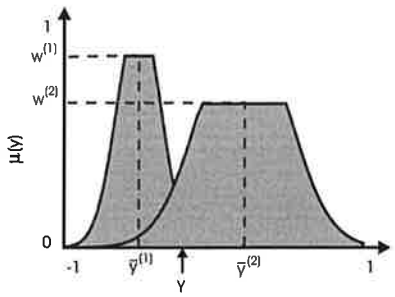


Figure 4.5 Center Average defuzzifification of the fuzzy set B'.

Since the defuzzifier should infer a unique output, there are three different Maximum defuzzifiers

1. Smallest of Maxima (SOM), for which y^* is the smallest value in I_M.
2. Mean of Maxima (MOM), for which y^* is the middle point of I_M.
3. Largest of Maxima (LOM), for which y^* is the largest value in I_M.

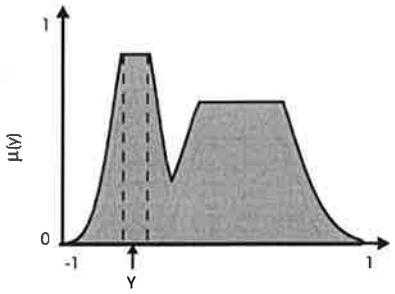


Figure 4.6 Maximum defuzzifification of the fuzzy set B'.

□

There are several factors which can influence the choice of defuzzifier. We may for instance require that the defuzzifier produces a plausible result in a computationally simple way. In the next chapter, we shall see that both the Center Average and the Center of Gravity defuzzifiers allow efficient computations.

4.5. Putting the Pieces Together – The Fuzzy System

Having defined all components of a fuzzy system, we can now put the pieces together to form the fuzzy system. The fuzzy system is illustrated in Figure 4.5.

For later reference, we make a more formal definition of the fuzzy systems under consideration.

DEFINITION 4.5.1 (FUZZY SYSTEM) *A fuzzy system of class* $\mathcal{S} = \mathcal{S}(\mathcal{R}, \mathcal{F}, \mathcal{I}, \mathcal{D})$, *consists of*

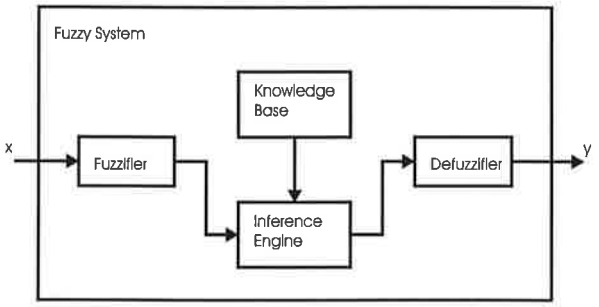


Figure 4.7 The fuzzy system.

$\mathcal{R}$ – *a rule base as defined in Definition 4.1.1*

$\mathcal{F}$ – *a fuzzifier as defined in Definition 4.2.1*

$\mathcal{I}$ – *an inference engine as defined in Definition 4.3.1*

$\mathcal{D}$ – *a defuzzifier as defined in Definition 4.4.1*

□

In Section 3.1, we briefly mentioned that the singleton fuzzifier is predominant in fuzzy control literature. The main reason for this can be found in Example 3.6.2. In this example, we showed that the generalized modus ponens simplifies considerably if the singleton fuzzifier is used. It is the simplified inference computations that are implemented in most fuzzy controllers. These computations also have a very intuitive interpretation, as illustrated in the next example.

EXAMPLE 4.5.1 (INFERENCE COMPUTATIONS)
Consider a fuzzy system of class $\mathcal{S}$ with a singleton fuzzifier. In Example 3.6.2, we showed that in the case of singleton fuzzifier and Mamdani implication, the generalized modus ponens infers according to

$$\mu_{B'}(y) = \mu_A(x^*) \star \mu_B(y)$$

where x^* denotes the input to the system.

Let us illustrate the inference computations on the fuzzy PD controller rule base of Example 3.1 with fuzzy sets defined in Example 3.2. In this case, we have $x^* = (e, \dot{e})$, and consequently

$$\mu_{B'}(y) = \mu_{A_1}(e) \star \mu_{A_2}(\dot{e}) \star \mu_B(y)$$

The inference computations can be recast into the following steps

1. The fuzzy sets of all inputs are evaluated.

$$\mu_{A_1^{(i)}}(e),\ \mu_{A_2^{(i)}}(\dot{e})$$

2. The degree of fulfilment of each rule is determined by applying the desired T-norm.

$$\mu_{A^{(i)}}(e, \dot{e}) = \mu_{A_1^{(i)}}(e) \star \mu_{A_2^{(i)}}(\dot{e})$$

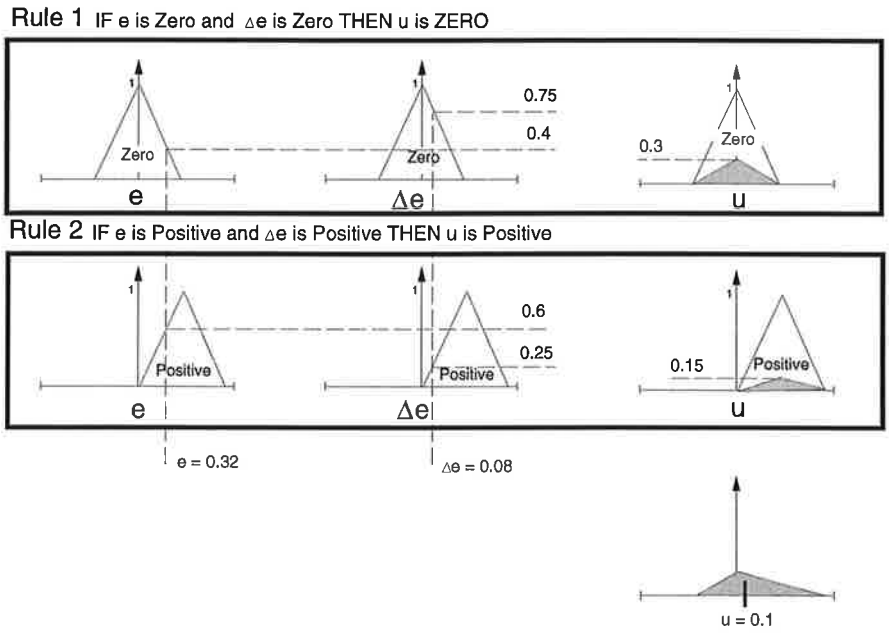


Figure 4.8 Graphical illustration of the simplified inference computations.

3. The contribution of each rule to the control signal is determined by the Mamdani implication

$$\mu_{B'^{(l)}}(y) = \mu_{A^{(l)}}(e, \dot{e}) \star \mu_{B^{(l)}}(y)$$

4. The output fuzzy set is formed by aggregating the individual rule contributions.

$$\mu_{B'}(y) = \mu_{B'^{(1)}}(y) \dot{+} \ldots \dot{+} \mu_{B'^{(M)}}(y)$$

For the case of Product-Sum inference, parts of the computations are illustrated in Figure 4.8.

□

In the next chapter, we shall see how the computations can be simplified even further in the case where the fuzzy sets of the consequents are singletons. Before moving on to the next chapter, however, we choose to make a small detour to describe a slightly different class of fuzzy systems.

4.6. Sugeno Type Fuzzy Systems – A Different Approach

A different class of fuzzy-like systems has been suggested by Sugeno and Takagi [Takagi and Sugeno, 1985]. This type of fuzzy system uses rules on the form

$$R^{(l)}: \text{IF } x_1 \text{ IS } A_1^{(l)} \text{ AND } \ldots \text{ AND } x_n \; IS \; A_n^{(l)} \text{ THEN } y = h^{(l)}(\boldsymbol{x}) \qquad (4.2)$$

The output function is an arbitrary function of the input $\boldsymbol{x}$. In many cases the output function is a constant or a linear combination of the input variables, i.e.

$$h^{(l)}(\boldsymbol{x}) = l_0 + l_1^{(l)} x_1 + \cdots + l_n^{(l)} x_n$$

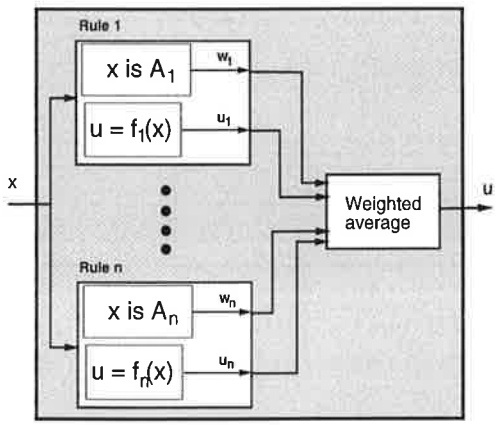


Figure 4.9 Graphical illustration of inference in Sugeno-type fuzzy systems.

The total system output is a weighted average of the individual rule outputs

$$y = \sum_{l=1}^{M} \frac{\mu_{A^{(l)}}(x)}{\sum_{k=1}^{M} \mu_{A^{(k)}}(x)} h^{(l)}(x)$$

where the weights $\mu_{A^{(l)}}(x)$ are computed according to

$$\mu_{A^{(l)}}(x) = \prod_{i=1}^{n} \mu_{A_i^{(l)}}(x_i)$$

The simple inference mechanism of a Sugeno-type fuzzy system is illustrated in Figure 4.9.

Although only a minimum of fuzzy logic is used in Sugeno-type fuzzy systems, it will become evident in the next section that these systems capture most of the attractive features of fuzzy systems.

4.7. Summary

In this chapter, we have developed the extensions of the basic theory of Chapter 2, necessary for fuzzy logic to be applicable to engineering problems. Firstly, we have moved from reasoning with one single rule to reasoning with fuzzy system rule bases that consist of several rules. Secondly, since engineering applications usually involve numeric manipulations, we have shown how interfaces must be added around the fuzzy logic inference engine.

It is natural to structure a fuzzy system into four subsystems; fuzzifier, rule base, inference engine and defuzzifier. In theory, any combination of fuzzifier, rule base, inference mechanism and defuzzifier can be used. In practice, however, only a limited number of combinations are widely acknowledged. The singleton fuzzifier, for instance, is predominant in fuzzy control, and we have shown how the rule evaluation in this case has a nice graphical interpretation. In the next chapter, we will see that if we limit the fuzzy system parameters further, we can derive closed form expressions of the nonlinear mappings performed by fuzzy systems. For more material on fuzzy systems, see [Wang, 1994] [Driankov and M.Reinfrank, 1993] and [Pedrycz, 1989].

5. From Fuzzy Systems to Nonlinear Mappings

Aim: To show how fuzzy systems perform nonlinear mappings

From the conceptual point of view, fuzzy systems are motivated by the theory of fuzzy sets and the attempt of approximate reasoning to mimic how human beings reason with vague terms. From the mathematical point of view, however, fuzzy systems are nothing but nonlinear mappings from the inputs to the outputs. When we want to express heuristic knowledge of an input-output relationship, it is often convenient to manipulate fuzzy IF-THEN rules. When we want to develop systematic analysis and synthesis methods for fuzzy control systems, however, it is instrumental that we view fuzzy systems as nonlinear mappings. This view of fuzzy systems is illustrated in Figure 5.1. The aim of

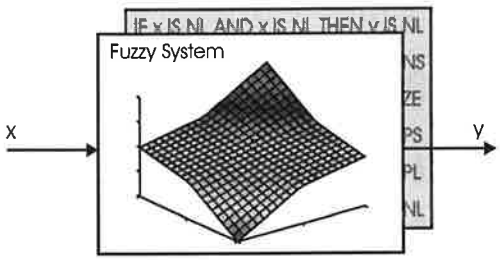


Figure 5.1 Shifting focus: From internal mechanisms to an input-output view.

this chapter is to give insight in how fuzzy systems synthesize nonlinearities and to derive compact analytic descriptions of fuzzy system nonlinearities. We will also provide measures on how well fuzzy systems can approximate continuous functions. Finally, we will relate fuzzy systems to other function approximation schemes, such as neural networks, splines and radial basis functions. This presentation of fuzzy system nonlinearities condenses and develops the ideas described in [Johansson, 1993], [Johansson, 1994a] and [Johansson, 1994b].

5.1. Crafting Nonlinearities with Fuzzy Systems

Fuzzy systems have a very large number of adjustable parameters. The parameters can be divided into two groups: one group of parameters specifies the knowledge base, and the other parameters determine how the rules should be interpreted.

Rule base	Fuzzifier	Inference Engine	Defuzzifier
Number of fuzzy sets	Fuzzification	Inference mehod	Defuzzification
Fuzzy set distributions		Implication	
Fuzzy set shapes		T-norm	
Fuzzy IF-THEN rules		S-norm	

The approach taken in this work is to fix the reasoning parameters, and investigate how the knowledge base parameters influence the nonlinearity. We will develop an intuition of how fuzzy systems synthesize nonlinearities based on the similarity between a fuzzy system and a look-up table. In fact, describing a nonlinearity by fuzzy IF-THEN rules is very similar to describing a nonlinearity by a look-up table with interpolation. Let us illustrate this analogy by an example.

EXAMPLE 5.1.1 (LOOK-UP TABLE)
Consider describing the nonlinear valve-characteristic

$$y = \sqrt{x}$$

with a look-up table. A look-up table consists of a set of nodal points, n_k, and the corresponding function values $\mathcal{L}(n_k)$. A coarse look-up table for this nonlinearity is given by

Nodes	$(x = n_k)$	0	0.5	1
Outputs	$(y = \mathcal{L}(n_k))$	0	0.7	1

We can describe the operation of this look-up by the following rules

$$\begin{aligned}
&\text{IF} \quad 0.00 \leq x < 0.25 \quad \text{THEN} \quad y = 0 \\
&\text{IF} \quad 0.25 \leq x < 0.75 \quad \text{THEN} \quad y = 0.7 \\
&\text{IF} \quad 0.75 \leq x \leq 1.00 \quad \text{THEN} \quad y = 1
\end{aligned}$$

The characteristic functions used in these rules are shown in Figure 5.2, along with the resulting nonlinearity.

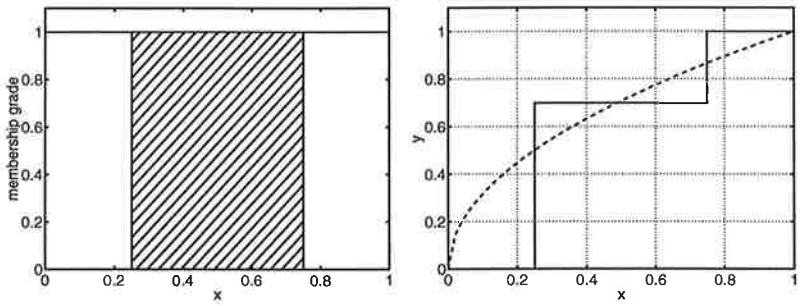


Figure 5.2 Left: Characteristic functions. Right: Binary logic approximation (full line) compared to actual nonlinearity (dashed).

If we choose to synthesize the same nonlinearity with a fuzzy system, we could use the rules

$$\begin{aligned}
&\text{IF} \quad x \text{ IS } \textit{Small} \quad \text{THEN} \quad y = 0 \\
&\text{IF} \quad x \text{ IS } \textit{Medium} \quad \text{THEN} \quad y = 0.7 \\
&\text{IF} \quad x \text{ IS } \textit{Large} \quad \text{THEN} \quad y = 1
\end{aligned}$$

The membership functions for *Small*, *Medium* and *Large* would in this case be continuous versions of the characteristic functions used in the logic rules.

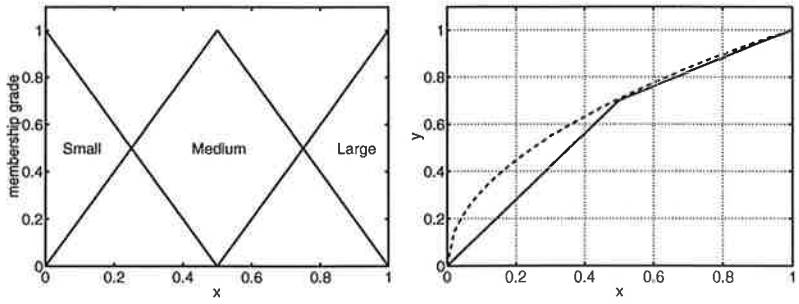


Figure 5.3 Left: Membership functions. Right: Fuzzy system nonlinearity (full line) compared to actual valve nonlinearity (dashed).

Typical membership functions and the fuzzy system nonlinearity are shown in Figure 5.3.

Note that this fuzzy system exactly reconstructs the table at the nodal points while for intermediate values, the reasoning mechanism results in an interpolation. This results in a continuous approximation of the valve nonlinearity. We could of course obtain a similar result by combining the original look-up table with some other interpolation method. □

We may say that in fuzzy systems, the fuzzy set distribution and rules play the role of a look-up table. This analogy is particularly transparent when we use triangular membership functions and an information system rule base (such as the fuzzy rule base of the previous example). In this case, the rule base parameters can be shown to have the following influence on the nonlinearity:

- The fuzzy set distributions used in the rule premises partition the input space into a set of intervals.
- The rules, in turn, assign one function value to each interval endpoint (or nodal point). The function values are determined by the location of the fuzzy sets of the consequent variable.
- For intermediate input values, the fuzzy system calculations result in an interpolation.

The analogy is a powerful tool for developing a qualitative understanding of how the parameters of the rule base influence the fuzzy system nonlinearity. In the next section, we will see how particular choices of

This analogy is a powerful tool for developing a qualitative understanding of how the parameters of the rule base influence the fuzzy system nonlinearity. It can also be useful when we want to find out the nonlinearity of a specific fuzzy system, as illustrated by the next example.

EXAMPLE 5.1.2 (INTERPOLATION IN 2-INPUT SYSTEMS)
Consider the Fuzzy PD controller of Example 4.1.1, with membership functions defined in Figure 5.4.

We can get an idea of what the controller nonlinearity looks like using the procedure outlined above:

- The vertices of the input fuzzy sets partition the input space into a Cartesian grid.

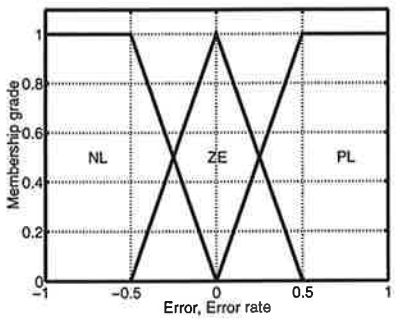
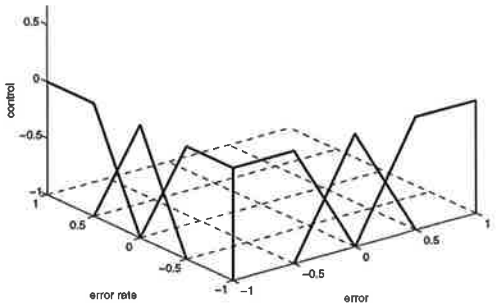
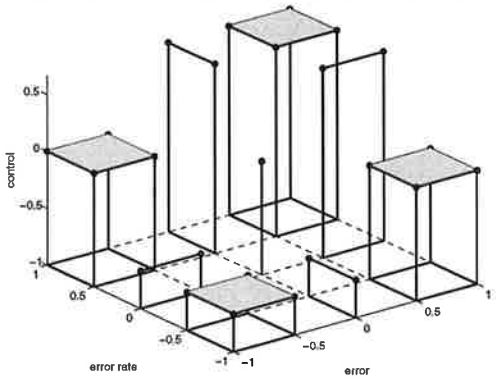


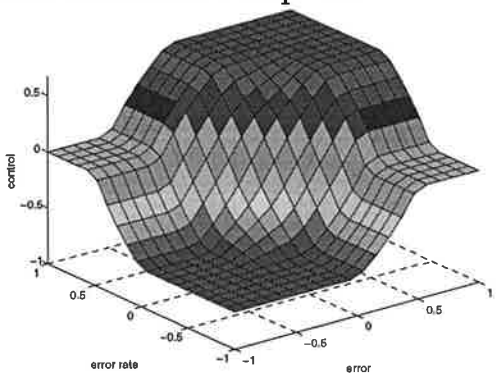
Figure 5.4 Fuzzy Sets for the PD controller.



- The rules order one function value to each node of the grid.



- The fuzzy system calculations interpolate



□

Although we have only considered triangular membership functions, the analogy holds with good approximation also for other classes of membership

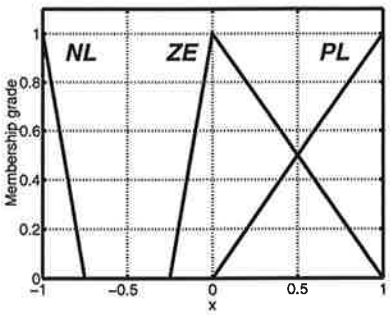


Figure 5.5 Fuzzy sets for the P-controller.

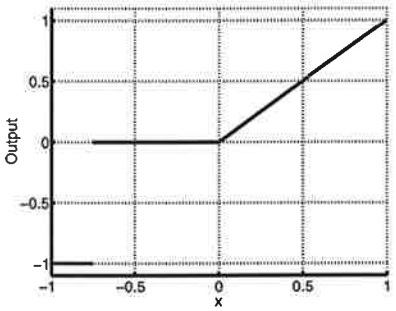


Figure 5.6 The resulting nonlinearity of the fuzzy P-controller.

functions. In a majority of cases, triangular membership functions are chosen to have full overlap. This is generally a good idea, as illustrated by the next example.

EXAMPLE 5.1.3 (WHY OVERLAPPING FUZZY SETS?)
Consider a simple P-controller described by the rules

$$\begin{aligned} &\text{IF } e \text{ IS } NL \quad \text{THEN } u \text{ IS } NL \\ &\text{IF } e \text{ IS } ZE \quad \text{THEN } u \text{ IS } ZE \\ &\text{IF } e \text{ IS } PL \quad \text{THEN } u \text{ IS } PL \end{aligned}$$

where the fuzzy sets for e are defined in Figure 5.5, and the fuzzy sets for u are singletons centred at $(-1,\ 0,\ 1)$ respectively. The resulting nonlinearity is shown in Figure 5.6. Note that this nonlinearity exhibits both plateus and discontinuities, which are often undesireable features.

For the inference parameters that we will use later in this chapter (see Lemma5.2.1–5.2.3 and Corollary5.2.1), the following rule-of-thumb can be shown to hold:

1. No active rule results in a zero output.
2. One active rule results in a constant output.
3. Several active rules result in an interpolation.

Compare with the P-controller nonlinearity. □

In the next section, we will see how particular choices of fuzzifier, inference engine parameters and defuzzifier, make it possible to derive closed form

expressions for the fuzzy system mappings. These expressions will be instrumental for mathematical analysis of fuzzy control systems.

5.2. The Nonlinear Mappings of Fuzzy Systems

We have seen that the fuzzification, inference and defuzzification can be performed in a number of ways; the only requirements we can pose are that a set of basic axioms are fulfilled for each operation. Further, we can conclude that there is no "best" way to perform a specific fuzzy set operation, it is a matter of preference what operations we choose in the implementation of our fuzzy system. In this section, we prefer fuzzy systems that

1. Can be described by compact mathematical formulas, and
2. Are computationally efficient

The first requirement makes mathematical analysis feasible, while the second requirement guides us to design control systems that we can implement without having to rely on expensive special-purpose hardware. We derive separate formulas for the mappings of Mamdani-type fuzzy systems and Sugeno-type systems.

Mamdani-type Fuzzy Systems

A very useful formula for fuzzy system mappings is the following:

LEMMA 5.2.1 (FUZZY SYSTEM MAPPINGS I) *The fuzzy system with rule base on the form $\mathcal{R}$, singleton fuzzifier, Product-Sum or Min-Max inference and CA defuzzification, performs the nonlinear mapping*

$$y = \sum_{l=1}^{M} \frac{\prod_{i=1}^{n} \mu_{A_i^{(l)}}(x_i)}{\sum_{k=1}^{M} \prod_{i=1}^{n} \mu_{A_i^{(k)}}(x_i)} \bar{y}^{(l)} \tag{5.1}$$

where $\bar{y}^{(l)}$ are the centres of the consequent fuzzy sets. □

This formula is the same, whatever membership functions are used in the rule-premises. It is therefore important to stress that the choice of function class of the membership functions $\mu_{A_i^l}(x_i)$ has a great impact on the fuzzy system nonlinearity. Triangular membership functions, for instance, are not continuously differentiable and have very local support. Gaussian membership functions, on the contrary, are infinitely smooth and are supported for all input values.

Since Gaussian membership functions have global support, they allow us to obtain a more transparent formula.

COROLLARY 5.2.1 (GAUSSIAN MEMBERSHIP FUNCTIONS IN THE PREMISES) *If the fuzzy system of Lemma 5.2.1 uses gaussian membership functions in the rule premises, the fuzzy system performs the nonlinear mapping*

$$y = \sum_{l=1}^{M} \frac{\exp\left(-\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}_i^{(l)}}{\sigma_i^{(l)}}\right)^2\right)}{\sum_{k=1}^{M} \exp\left(-\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}_i^{(k)}}{\sigma_i^{(k)}}\right)^2\right)} \bar{y}^{(l)} \tag{5.2}$$

□

Fuzzy systems with triangular membership functions in the rule premises have some nice properties, as we can see from the following results.

LEMMA 5.2.2 (FUZZY SYSTEM MAPPINGS II) *The fuzzy system with a consistent information system rule base, $\mathcal{R}$, triangular membership functions $\mu_{A_i^l}(x)$ in the premises, singleton fuzzifier, product-sum inference and COG defuzzifier, perform the nonlinear mapping*

$$y = \sum_{l=1}^{M} \frac{\prod_{i=1}^{n} \mu_{A_i^{(l)}}(x_i)}{\sum_{k=1}^{M} a^{(k)} \prod_{i=1}^{n} \mu_{A_i^{(k)}}(x_i)} m^{(l)} \tag{5.3}$$

where $a^{(l)}$ and $m^{(l)}$ are the area and the moment respectively of the fuzzy set $B^{(l)}$. □

Note that this fuzzy system allows separate manipulation of the numerator and denominator of the interpolation functions. If we do not need this extra degree of freedom, the simplification indicated by the next lemma can be an alternative.

LEMMA 5.2.3 (FUZZY SYSTEM MAPPINGS III) *If the consequent fuzzy sets are all singletons, the fuzzy system mapping (II) simplifies to*

$$y = \sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu_{A_i^{(l)}}(x_i) \right) \bar{y}^{(l)} \tag{5.4}$$

□

In this fuzzy system, we have fixed a majority of the fuzzy system parameters. We are only free to adjust the most important parameters of the rule base, namely the number of rules, M, the distribution of the fuzzy sets and the rules. Thanks to its simplicity, we can derive some more specific results for this class of fuzzy systems.

COROLLARY 5.2.2 (PIECEWISE MULTILINEAR MAPPINGS) *The fuzzy system (III) performs a piecewise multilinear mapping*

$$y = (\alpha_1 x_1 + \beta_1) \cdot (\alpha_2 x_2 + \beta_2) \cdot \cdots \cdot (\alpha_n x_n + \beta_n) \tag{5.5}$$

□

Corollary 5.2.2 indicates that a lot of precomputations can be made when implementing this class of fuzzy systems. Since linear systems is a special case of multilinear systems, we have the following constructive result.

COROLLARY 5.2.3 (LINEAR FUZZY MAPPINGS) *By choosing the consequent parameters*

$$\bar{y}^{(l)} = L^T \begin{bmatrix} x_{u,1}^{(l)} \\ x_{u,2}^{(l)} \\ \vdots \\ x_{u,n}^{(l)} \end{bmatrix} \tag{5.6}$$

the fuzzy system of Lemma 5.2.3 performs the linear mapping

$$y = L^T x \tag{5.7}$$

□

Note that Corollary 4.3 gives us one way to design a fuzzy controller. We can simply start out by a linear controller design

$$u = L^T x$$

and translate it into a fuzzy system as in Lemma 1.3. With this as a starting point, we can tune the fuzzy system parameters in order to introduce nonlinearities that improve performance.

Although the last results are stated for the COG defuzzifier, the results from Lemma 5.2.3 and on are valid also for the CA defuzzifier. Similarly, the formulas (5.2.1) and (5.2.1) are valid if we use COG defuzzifier, singleton consequents and summation as aggregation operation. More formally, we have the following result:

PROPOSITION 5.2.1 *Using CA defuzzification is functionally equivalent to using COG defuzzification with the consequent parameters constrained to be singletons, and allowing the aggregation operation be summation.* □

Roughly speaking, the derivation of the general formulas (5.2.1) and (5.2.1) requires that the rule aggregation step is somewhat violated. Either we disregard the rule aggregations step completely (by using the CA defuzzifer), or we violate the S-norm (by using summation rather that bounded sum). The trick of the system in Lemma 5.2.2 is that its rule base guarantees that the aggregation operation never "hits the bound", i.e., the bounded summations is in fact a standard summation.

Sugeno-type Fuzzy Systems

For Sugeno-type fuzzy systems, we limit our presentation to the following lemma:

LEMMA 5.2.4 (FUZZY SYSTEM MAPPINGS IV) *The Sugeno-type fuzzy system (4.2) performs the nonlinear mapping*

$$y = \sum_{l=1}^{M} \frac{\prod_{i=1}^{n} \mu_{A_i^{(l)}}(x_i)}{\sum_{k=1}^{M} \prod_{i=1}^{n} \mu_{A_i^{(k)}}(x_i)} h^{(l)}(x)$$

□

Here, we allow $h^l(x)$ to be an arbitrary function of the system input x. In most cases, however, the consequent function is a linear function

$$h^{(l)}(x) = l_1 x_1 + l_2 x_2 + \cdots + l_n x_n$$

Note that the Sugeno-type of fuzzy systems simplify to Mamdani-type fuzzy systems when the consequent functions are constant functions ("singletons").

$$h^{(l)}(x) = \bar{y}^{(l)}$$

5.3. Approximation Properties of Fuzzy Systems

Since we use fuzzy systems to synthesize nonlinearities, it is interesting to examine what functions we can approximate using fuzzy systems. As formulated by the next theorem, it turns out that a wide class of fuzzy systems are "universal approximators" [Castro, 1995].

THEOREM 5.3.1 (FUZZY SYSTEMS ARE UNIVERSAL APPROXIMATORS) *Let*

$$f : U \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$$

be a continuous function defined on a compact set U. Let $\mathcal{S}' = \mathcal{S}(F_\delta, I, D)$ be a fuzzy system defined in Definition 4.5.1, where F_δ denotes the singleton fuzzifier. Then for each ϵ, there exists an $\mathcal{S}'_\epsilon \in \mathcal{S}'$ such that

$$\sup_{x \in U} |f(x) - \mathcal{S}'_\epsilon(x)| \leq \epsilon$$

□

This theorem tells us that for every continuous function $f(x)$, we can find a fuzzy system $\mathcal{S}'_\epsilon$ that approximates this function with a maximum absolute error of ϵ. Based on the analogy of a look-up table, the above result is not surprising. A fuzzy system of class $\mathcal{S}'$, can represent the function $f(x)$ exactly at the nodal points. Since $f(x)$ is a continuous function , we can approximate $f(x)$ arbitrary well by simply increasing the number of rules (cf number of entries in a look-up tanle).

For practical use, we would like to know what worst-case approximation error is for a certain number of rules. This approximation bound is provided by the following theorem [Xiao-Jun Zeng, 1995]

THEOREM 5.3.2 (APPROXIMATION BOUNDS FOR FUZZY SYSTEMS) *Let*

$$f : U \subseteq \mathbb{R}^2 \rightarrow \mathbb{R}$$

be a twice continously differentiable function defined on a compact set U. Let $\mathcal{S}_\Delta$ be the fuzzy system of Lemma 1.3. Then

$$||f - \mathcal{S}_\Delta||_\infty \leq \frac{1}{8}\left(||\frac{\partial^2 f}{\partial x_1^2}||_\infty \delta_1^2 + ||\frac{\partial^2 f}{\partial x_2^2}||_\infty \delta_2^2\right)$$

where we have introduced

$$\begin{aligned} \delta_1 &= \max_{i=1\ldots m_1-1} |x_{u1}^{i+1} - x_{u1}^i| \\ \delta_2 &= \max_{i=1\ldots m_2-1} |x_{u2}^{i+1} - x_{u2}^i| \end{aligned}$$

□

The result of this theorem also matches the analogy of the look-up table very well. Recall that this fuzzy system can exactly represent the unknown function at the nodal points. Since we require $f(x)$ to be continuous, we can bound its behaviour inbetween the nodal points.

5.4. Fuzzy Systems and Function Approximation

In this chapter, we have seen how fuzzy systems perform nonlinear mappings. One of the key results has been that certain choices of inference parameters allow us to derive closed form expressions of the mappings performed by these fuzzy systems. In other words, we can get a one-to-one correspondance between the fuzzy system knowledge base (which may be given by a heuristic design) and a simple nonlinear function (which we hope to use for mathematical analysis).

The formulas for fuzzy system mappings that we have derived can all be written as a weighted sum of basis functions, i.e.

$$f(x;\theta) = \sum_{i=1}^{M} \overbrace{g_i(x)}^{\text{IF-part}} \underbrace{w_i}_{\text{THEN-part}} \tag{5.8}$$

To obtain correspondance to the notation used in Lemma 5.2.1 and Lemma 5.2.3, note that the weights are determined by

$$w_i = \bar{y}^{(i)} \tag{5.9}$$

and the "fuzzy basis functions"[Wang, 1994] are given by

$$g_i(x) = \frac{\prod_{k=1}^{n} \mu_{A_k^{(i)}}(x_k)}{\sum_{i=1}^{M} \prod_{k=1}^{n} \mu_{A_k^{(i)}}(x_k)} \tag{5.10}$$

The parametrization (5.8) of a function approximation scheme is by no means unique to fuzzy systems. Many conventional function approximation methods such as Splines and Radial Basis Functions (RBF) can be written in the same form. Quite surprisingly, it turns out that the fuzzy system mapping of Lemma 5.2.3 is functionally equivalent to a Linear B-Spline [deBoor, 1978], [Brown and Harris, 1995] and that it would be fair to refer to the fuzzy system mapping of Corollary 5.2.1 as a normalized RBF Altogether, this indicates that a basic understanding of function approximation can be very useful for understanding the problems and prospects of using fuzzy systems for control. The remainings of this chapter will be devoted to a brief description of how fuzzy systems relate to other function approximation methods.

The function approximation problem can be stated as follows: "Given a training set $(x(i), y(i))$ consisting of N pairs of inputs and the corresponding outputs, construct a map that for a new input x provides a reasonable prediction of the associated unobserved output y." [Wahba, 1995]. In machine learning, this ability to predict an unobserved output associated with a new input is often referred to as generalization.

Clearly, just storing away the training data is not likely to be sufficient. Rather, we would like to construct some function that matches the training data reasonably well. How the function approximation should be carried out in detail depends on what information is available: Is the training data noisy or exact? Is the function to be approximated known to be smooth, and in that case to what degree? In order to understand how fuzzy systems fit into the framework of function approximation, consider the following examples:

EXAMPLE 5.4.1 (RULE BASED SYSTEMS AND INTERPOLATION)
Fuzzy systems are often used for crafting controller nonlinearities based on the knowledge of an experienced process operator. In this case, it is reasonable to interpret the fuzzy rule

$$\text{IF } x \text{ IS } A^{(l)} \quad \text{THEN } y \text{ IS } B^{(l)} \tag{5.11}$$

as a description of the controller nonlinearity close to some operating condition $x^{(l)}$. In the same spirit, the complete rule base can be seen as a set of sparse data points $(x(i), F(x(i)))$, describing the unknown nonlinearity. Based on these data, we are now interested in recovering $F(x)$ by a fuzzy system $f(x)$.

If we believe that the operator has given us exact descripritions of his controller actions, it is natural to require that the fuzzy system mapping exactly matches the training data, i.e. that the interpolation conditions

$$f(x(i)) = F(x(i)) \quad \forall i = 1 \ldots M \tag{5.12}$$

are fulfilled. The fuzzy system used in Lemma 5.2.3 , for example, interpolates the points $\left(x_u^{(l)}, \bar{y}^{(l)}\right)$ as illustrated in Figure 5.7 □

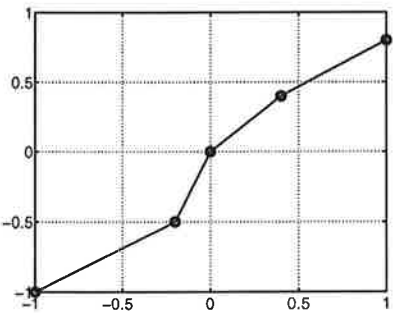


Figure 5.7 Fuzzy system nonlinearities – rules and interpolation.

EXAMPLE 5.4.2 (NONLINEAR SYSTEM IDENTIFICATION AND SMOOTHING)
The situation is quite different when we want to identify some static nonlinearity based on measurement of input-output data. Since the data may contain a considerable amount of noise, interpolation is now a dubious goal. Even if the noise level is low, we often choose to collect a large set of data, and constructing an interpolant to this data would require very many parameters (which in fuzzy systems translates to rules and membership functions).

In this case, it is more natural to try to construct a function that matches the training data closely under some constraints on the shape of the approximant. This is known as the smoothing problem, illustrated in Figure 5.8. □

The tradeoff between closeness to the training data and smoothness of the approximant is captured by the regularization problem: Find the approximant f that minimizes the functional

$$H[f] = \sum_{i=1}^{N} (f(x_i) - y_i)^2 + \lambda \Phi[f] \tag{5.13}$$

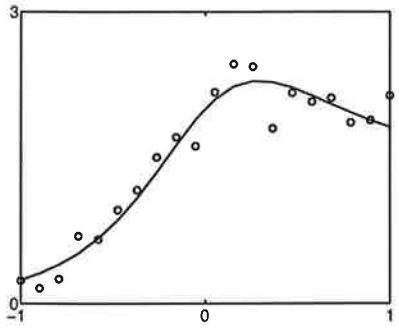


Figure 5.8 Smoothing – Construct a smoooth function that matches the data.

Here, the first term enforces closeness to the data, and $\Phi[f]$ is a functional that enforces smoothness of f. A typical smoothness functional is

$$\Phi[f] = \int \left(f^{(2)}(u)\right)^2 \mathrm{du} \tag{5.14}$$

which penalizes large second derivatives of the approximant. The positive parameter λ is known as the regularization parameter, and it is used to control the tradeoff between the two terms. The exterme case $\lambda \to 0$ corresponds to the interpolation problem. The other extreme, $\lambda \to \infty$, along with the smoothness functional (5.14) results in a solution that tends to the straight line that gives the best least squares fit of the data.

Several function approximation schemes, such as Splines and RBF:s, can be derived from regularization theory by posing the appropriate class of smoothness functional [Federico Girosi, 1993]. We may thus argue that there is some form of regularization inherent in our choice of function approximation method. Thus, in fuzzy systems the choice of membership function shape acts as some sort of "implicit regularization". Gaussian membership functions, for instance, can give fuzzy systems that are infinitely smooth, while triangular membership functions can at best result in a continuous approximant. For more specific results on function approximation in the context of system identification, see [Juditsky *et al.*, 1995] and [Sjöberg *et al.*, 1995].

Regularization ideas can also be used to understand some of the problems that are often encountered when we use optimization methods to adjust fuzzy system parameters to input-output data:

EXAMPLE 5.4.3 (FUZZY SYSTEMS AND THE BIAS-VARIANCE TRADEOFF)
When fuzzy systems are used to approximate functions, the parameters of the fuzzy system mappings are often optimized using some iterative descent algorithm that tries to minimize some cost functional, typically

$$J_{SS} = \sum_{i=1}^{N} (f(x_i) - y_i)^2 \tag{5.15}$$

Clearly, if the number of system parameters (rules) is large enough, and the descent algorithm is allowed to run long enough, this optimization would give a fuzzy system parameters that interpolate the data. From Example 5.4.2, we know that for noisy data, this is often not a very good idea. A "poor man's regularization" in this case, may be done by stopping the descent algorithm

early (in other words, not driving J_{SS} as far as it can go) or adjusting the number of parameters of the fuzzy system. Bear in mind though, that the fuzzy system should have enough parameters to give a fair reconstruction of the unknown function. This dilemma is known in statistical literature as the 'bias-variance tradeoff'. □

5.5. Fuzzy Systems and Neural Networks

A graphical illustration of the computations involved in the function approximation schemes on the form

$$f(x;\theta) = \sum_{l=1}^{M} g_i(x) w_i \tag{5.16}$$

takes the form of a feedforward network, as illustrated in Figure 5.5. Similarly,

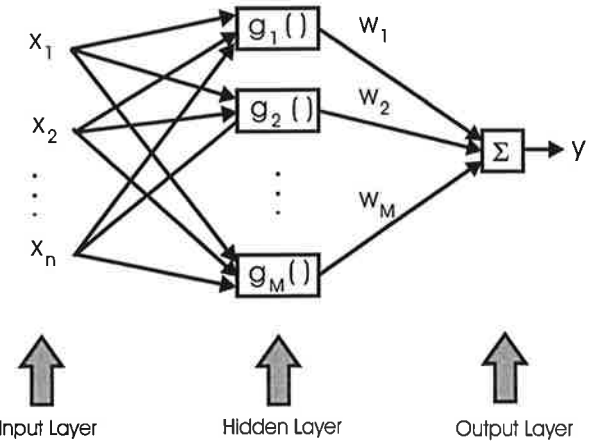


Figure 5.9 The evaluation of many function approximation schemes can be illustrated as a feedforward network.

most function approximation algorithms can be recast into the form of (feedforward) neural networks. Indeed, neural networks based on RBF:s, Splines and Fuzzy Systems have been suggested in the literature. Consequently, many results from function approximation theory and neural network control applies directly to fuzzy systems, and (naturally) the other way around.

It is the author's strong belief that automatic control research would benefit from a unification of these subtly different approaches to approximation based control instead of a allowing a continuous recasting of well-known results from one field into the other. We advocate viewing fuzzy systems as nonlinear mappings that can be interpreted as a set of linguistic rules. Under certain choices of fuzzy system parameters, we have seen how fuzzy systems are functionally equivalent to well-established function approximation schemes. A graphical illustration of the evaluation of these function approximation schemes takes the form of a network, and so, they qualify as neural networks.

5.6. Summary

This chapter has been concerned with describing how fuzzy systems perform nonlinear mappings. A qualitative understanding of fuzzy system mappings

has been developed based on the analogy of a look-up table. Loosely speaking, the rules determine the output for certain combinations of input values while the inference mechanism interpolates the output for other inputs. This intuitive idea of how fuzzy systems perform nonlinear mappings (using rules and interpolation) is instrumental for identifying the role of different fuzzy system parameters, and for heuristic tuning of fuzzy systems. In this context, fuzzy systems have the nice property of being nonlinear mappings that can be interpreted in terms of a set of IF-THEN rules.

We have also seen how quantitative descriptions of fuzzy system mappings can be derived for certain classes of fuzzy systems. Deriving these closed form expressions of fuzzy system nonlinearities is a way of bringing fuzzy systems "back home". These classes of fuzzy systems can now be analyzed in the (smooth) ODE framework using nonlinear control theory. Similarly, there are great possibilities for developing systematic design procedures for these classes of fuzzy controllers. Fuzzy systems for control applications will be treated in more depth in the next chapter.

6. Fuzzy Systems for Control

Aim: To show how fuzzy systems can be used for control

In the previous section we showed how fuzzy systems synthesize nonlinear mappings using rules and interpolation. We also derived compact closed form expressions of the nonlinear mappings performed by the most common fuzzy systems. Further, we have investigated to what accuracy fuzzy systems can approximate real-valued continuous functions.

In this section, we will discuss how fuzzy systems can be used in control. Since fuzzy systems have some nice properties in terms of function approximation, it is easy to be bold and exclaim that "every system can be fuzzified". In theory, this is indeed the case, but it is important to note the drawbacks of a representing a function by a fuzzy system; a fuzzy system requires a large number of parameters, and a significant amount of computations. Also, just noting that we can approximate every function by fuzzy systems does not mean that fuzzifying every control structure is always a good idea. Moving from linear to nonlinear control, we have to face issues such as filtering, estimation and controller design for nonlinear systems. Moreover, since fuzzy systems can only approximate the real system mapping, we have to take into consideration the effects of approximation errors. This section will give a flavor of possible applications of fuzzy control systems.

6.1. What is a Fuzzy Controller?

The early work in fuzzy control was motivated by a desire to express the control actions of an experienced human operator directly in an automatic controller, and to obtain a smooth interpolation between discrete controller outputs. Since that time, the application range of fuzzy control has widened substantially. As we will indicate in this chapter, fuzzy systems can be used in a variety of control problems. In most cases, a fuzzy controller is used for direct feedback control. However, a fuzzy controller can also be used on the supervisory level as e.g. a self-tuning device for a conventional PID controller.

As we have seen in the previous chapter, the most commonly used fuzzy systems are functionally equivalent to function approximation schemes such as B-splines or normalized Radial Basis Functions. If the parameters of a fuzzy system are adjusted on-line, the adaptive fuzzy system can be interpreted as a neural network. Consequently, the differences between "fuzzy control" and "approximation-based control" and between "adaptive fuzzy control" and "neural adaptive control" are in many cases philosophical rather than functional.

The variety of applications and functional similarity to other approaches makes it difficult to define what a fuzzy controller is. We will adopt the following general definition:

> A *Fuzzy Controller* is a controller that contains a possibly nonlinear mapping that has been defined using fuzzy logic-based rules.

The key-issue here is the presence of a nonlinear mapping that can be interpreted in terms of fuzzy logic-based rules.

6.2. The General Structure of a Fuzzy Controller

A fuzzy system mapping is just one part of a fuzzy controller. Often, signal processing is required both before and after the fuzzy system evaluation. A general structure that captures most applications of fuzzy systems to control is illustrated in Figure 6.1.

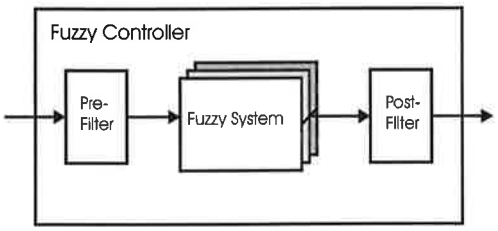


Figure 6.1 A general fuzzy controller structure, consisting of a prefiltering device, fuzzy system mappings and a postfiltering device.

This structure, which is a slight modification of the learning controller structure in [Sontag, 1993], consists of three parts:

1. A Prefiltering Device – for computing the fuzzy system inputs
2. One (or several) Fuzzy System Mapping(s)
3. A Postfiltering Device – for computing the actual control signal

Since the pre- and post-filtering devices are new components, we now discuss them in further detail.

Prefiltering Device

The prefiltering device represents the signal processing performed on the controller inputs in order to obtain the inputs of the fuzzy system. The prefiltering device may, for instance, perform some of the following operations on the input signals

Sampling. This includes time-sampling, quantization, and general A/D conversion.

Signal Conditioning. It is sometimes convenient to work with signals on a normalized domain. For instance, we may want to work with fuzzy sets on a normalized domain, typically $[-1, 1]$. This can be accomplished by the introduction of normalization gains. The normalization gain is a linear gain that scales the input into the normalized domain $[-1, 1]$. Values that fall outside the normalized domain are mapped onto the appropriate endpoint.

Dynamic Filtering. This includes both linear and nonlinear filters. In a fuzzy PID controller, for instance, linear filters are used to obtain the control error, the error derivative and the error integral. Nonlinear filters are found in nonlinear observers, and in adaptive fuzzy control where they are used to obtain the fuzzy system parameter estimates.

Feature Extraction. By feature extraction, we mean numeric transformations of the controller inputs. These transformations may be Fourier- or Wavelet-based transformations, coordinate transformations or other

basic operations performed on the fuzzy controller inputs. One interesting example is the linear transformation $W = V\chi$, with χ being the input vector and V a possibly rectangular matrix. Function approximation on linear transforms of input variables are found in so called "Ridge Approximation" schemes [Sjöberg *et al.*, 1995]. Another interesting example is to compute the pairwise products $x_{ij} = \chi_i \chi_j$, which allows correlations to be used as inputs to the subsequent parts of the controller.

Postfiltering Device

The postfiltering device represents the signal processing performed on the fuzzy system output to obtain the actual control signal. Operations that the postfiltering device may perform include

Precomputed Part of Control. In some fuzzy controllers, the purpose of the fuzzy system is to model the process dynamics. The "precomputed part of control" is then typically a model-based control scheme that uses the fuzzy model to compute the appropriate control action. Another example is when the fuzzy system is a supervisory tuning-device for a conventional PID controller. The "precomputed part of control" is then the PID algorithm, and the purpose of the fuzzy system is to select the appropriate PID parameters.

Signal Conditioning. This can be a denormalization gain that scales the output of the fuzzy system to the physical domain of the actuator signal.

Dynamic Filtering. In some cases, the output of the fuzzy system is the control increments. The actual control signal is then obtained by integrating the control increments. Of course, other forms of smoothing devices and even nonlinear filters may be considered.

Sampling. This is typically hold devices and more general D/A conversion.

6.3. Fuzzy Systems for Feedback Control

In most cases, fuzzy controllers are used for direct feedback control. In the literature, a considerable effort has been put into investigations of fuzzy controllers that are structurally equivalent to various conventional controllers. However, these investigations have often been limited to simulation studies – theoretical analysis and development of systematic design procedures have been largely ignored. Consequently, some nice indicative results exist, while many theoretical problems remain open. In the following, we will indicate how fuzzy systems can be used for control.

Nonlinear PID

Linear PID control is the most widely used control structure in process industry. In textbooks on automatic control, the PID controller is usually presented as

$$u(t) = K\left(e(t) + T_d \frac{d}{dt} e(t) + \frac{1}{T_i} \int_{\tau=0}^{t} e(\tau)\, d\tau\right) \tag{6.1}$$

An alternative parameterization of the PID controller is the velocity form

$$\frac{du}{dt} = K\left(\frac{d}{dt}e(t) + \frac{1}{T_i}e(t) + T_d\frac{d^2}{dt^2}e(t)\right) \tag{6.2}$$

The PID structure has also been favored by people working on fuzzy control. By replacing the mapping (6.2) by fuzzy rules, we obtain a fuzzy PID controller

$$\frac{du}{dt} = f\left(e(t), \frac{d}{dt}e(t), \frac{d^t}{dt^2}e(t)\right) \tag{6.3}$$

where $f(\cdot)$ is a possibly nonlinear fuzzy system mapping. For practical use, the textbook PID controller (6.1) is often refined by limiting the high-frequency gain in the derivative term and introducing setpoint-weighting along with a scheme for avoiding windup in the integral part. These issues should also be considered in a fuzzy PID controller

The fuzzy PID controller fits into the general fuzzy controller structure, as illustrated in Figure 6.2.

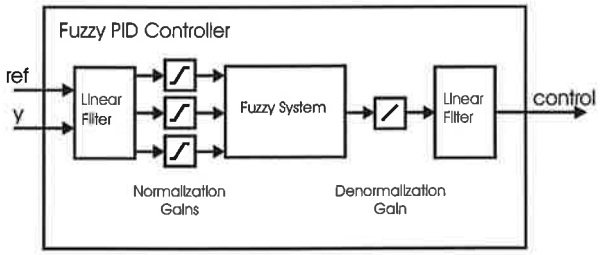


Figure 6.2 The Fuzzy PID Controller.

In this case, the prefilter contains of a linear filter that computes the control error, error derivative and error integral followed by three separate normalization gains. The fuzzy system uses the filtered control errors to compute the control increments. A subsequent integration is required to obtain the actual control. Thus, the postfiltering device consists of a linear filter integrating the control increments, and a denormalization gain scaling the filter output into the physical domain of the actuator.

There are potential advantages in making a PID controller nonlinear. If the plant is linear, we may for instance improve transient performance by crafting a controller nonlinearity that works as a time optimal controller when the control error is large and as a linear controller when the error is small. If the plant is nonlinear, the controller nonlinearity can be designed so as to compensate for plant nonlinearities.

However, it can be argued that the error-feedback framework is incorrect for nonlinear systems. Even if some nonlinear systems can be stabilized using error feedback, the dynamics in the error coordinates will change with the value of the reference signal. One way to circumvent this problem is to design a set of PID parameters for different operating conditions, and let the reference value influence what controller parameters are used. This is the idea behind gain scheduled controllers, described next.

Gain Scheduling using Fuzzy Systems

The dynamics of a nonlinear system varies with the operating condition. To design a fixed linear controller that works well for a broad range of operating

conditions is often a nontrivial task. In many cases, however, it is possible to find measurable variables that correlate well with the changes in process dynamics. The idea of a gain scheduled controller is to utilize this kind of variables to change, or schedule, the parameters of a linear controller so as to compensate for the changes in process dynamics.

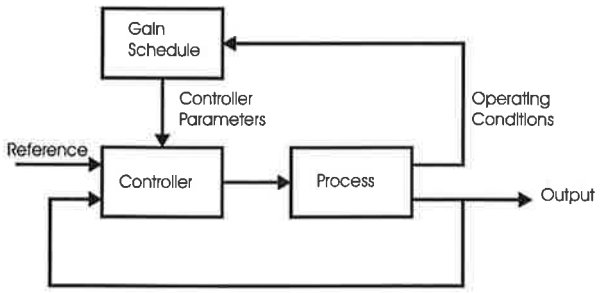


Figure 6.3 Gain scheduled control.

According to the discussion above, we can write a gain scheduled controller on the form

$$u = C(x; p)$$

where x is a vector of system state variables, and p is a vector of scheduling variables. Often, the scheduling is determined by logic rules. If the gain schedule uses an interpolation mechanism to guarantee smooth changes in controller parameters, a gain scheduled controller becomes very similar to a fuzzy controller.

One may indeed argue that all fuzzy controllers are gain scheduled controllers, but the relation is most clear in the case of Sugeno fuzzy systems. From Section 3.6, we know that Sugeno fuzzy systems uses rules on the form

$$R^{(l)} : \text{IF } p_1 \text{ IS } A_1 \text{ AND } \ldots \text{ AND } p_n \text{ IS } A_n \text{ THEN } y^{(l)} = L^{(l)T}x$$

A slight modification of Lemma 5.2.4 gives that this system performs the nonlinear mapping

$$y = \sum_{i=1}^{M} \frac{\prod_{i=1}^{n} \mu_{A_i^{(l)}}(p_i)}{\sum_{k=1}^{M} \prod_{i=1}^{n} \mu_{A_i^{(k)}}(p_i)} L^{(l)T}x = \sum_{l=1}^{M} \kappa^{(l)}(p) L^{(l)T}x = L^T(p)x$$

and the relation to gain scheduled controllers should be apparent.

The use of Sugeno fuzzy systems to design gain scheduled controllers is becoming increasingly popular. Somewhat confusing, this controller structure can (correctly) be called a Sugeno controller [Takagi and Sugeno, 1985], a gain scheduled controller [Åström and Wittenmark, 1995] or a regime based controller.

In gain scheduled controllers based on Mamdani-type fuzzy systems, the scheduling variables and process states both enter the rules antecedents:

$$R^{(l)} : \text{IF } p_1 \text{ IS } A_{p,1} \text{ AND } \ldots \text{ AND } x_1 \text{ IS } A_{x,1} \ldots \text{ THEN } y^{(l)} = B^{(l)}$$

Since the rules now have to describe both the scheduling space and the state space, the total number of rules may increase drastically compared to the Sugeno-type approach.

Identification of Fuzzy System Models

Many fuzzy control schemes are based on a fuzzy model of parts of the system to be controlled. It is therefore of interest to demonstrate briefly how the parameters of a fuzzy system model can be identified from input-output data.

From the discussion in the preceding chapter, we know that we can write fuzzy system models as a weighted sum of (fuzzy) basis functions

$$\hat{f}(x) = \sum_{i=1}^{M} \overbrace{g_i(x)}^{\text{IF-part}} \underbrace{c_i}_{\text{THEN-part}}$$

If we fix the shape parameters of the basis functions, the model is linear in the free parameters c_i. Identification of the weight parameters of a fuzzy system model is thus a linear regression problem, which can be addressed using standard least squares techniques.

Recall that the basis functions are determined by the rules premises, while the c_i:s correspond to the consequent parameters in the fuzzy system rules. In this setting, we thus fix the rule premises and adjust the consequent parameters to fit the fuzzy system mapping to a set of observational data. Off-line identification of a fuzzy system model is described in the example below.

EXAMPLE 6.3.1 (OFF-LINE IDENTIFICATION OF FUZZY SYSTEM MODELS)
Consider the static mapping

$$y = f(x) \tag{6.4}$$

for which it is hard to obtain a sufficiently good physical model.

Assume that we have some prior knowledge of $f(x)$, which we find convenient to decode into a fuzzy system rule base. Assume further that we can record N noisy input-output pairs $\{x^{(k)}, y^{(k)}\}$ of f, i.e.

$$\begin{aligned} y^{(1)} &= f(x^{(1)}) + e^{(1)} \\ y^{(2)} &= f(x^{(2)}) + e^{(2)} \\ &\vdots \\ y^{(N)} &= f(x^{(N)}) + e^{(N)} \end{aligned}$$

The first step of an identification procedure for fuzzy systems is to transform the initial rule base into a fuzzy system mapping

$$\hat{f}(x) = \sum_{i=1}^{M} g_i(x) c_i = \phi^T(x) C \tag{6.5}$$

where we have introduced the regressor vector

$$\phi(x) = \begin{bmatrix} g_1(x) & g_2(x) & \dots & g_M(x) \end{bmatrix}^T$$

and the parameter vector

$$C = \begin{bmatrix} c_1 & c_2 & \dots & c_M \end{bmatrix}^T$$

In matrix notation, we now have

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \phi^T(x_1) \\ \phi^T(x_2) \\ \vdots \\ \phi^T(x_N) \end{bmatrix} C + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix} = \Phi C + e$$

Assuming the noise to be white, the parameters of the fuzzy system model that minimizes the least squares criterion are obtained as

$$C = \Phi^+ Y$$

where $\Phi^+ = (\Phi^T\Phi)^{-1}\Phi^T$ denotes the Moore-Penrose inverse of the regressor matrix Φ. The extension to Mamdani-type models is straight-forward and omitted here for brevity. □

Adjusting the shape parameters of the basis functions can be posed as a nonlinear least squares problem, which can be approached using a number of "off the shelf" optimization routines such as Gauss-Newton or Conjugate Gradient [Fletcher, 1987]. Recursive identification of the linear parameters is also standard, see for instance [Johansson, 1993] for a comprehensive treatment of the recursive linear least squares method. For a nice treatment of identification of Mamdani-type models, the reader is referred to [Katayama *et al.*, 1994], while process identificaton using Mamdani-type models is treated in [Johansen, 1995].

Nonlinear State Feedback

Intuitively, a plant with significant nonlinear dynamics should be controlled using a nonlinear controller. However, control design for nonlinear systems is a hard problem, and at present date there are no general design methods available. Some promising progress have been made using techniques like feedback linearization and backstepping designs. Both methods can be applied to certain classes of systems. We choose to illustrate a special case of these design methods which is the nonlinear equivalent to pole placement.

EXAMPLE 6.3.2 (NONLINEAR POLE PLACEMENT)
Consider a nonlinear system on the form

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_{n-1} &= x_n \\ \dot{x}_n &= f(x) + g(x)u \end{aligned} \tag{6.6}$$

where $f(x)$ and $g(x)$ are nonlinear function of the process state vector. If we know $f(x)$ and $g(x) \neq 0$ perfectly, the feedback

$$u = \frac{1}{g(x)}\left(-f(x) + L^T x + r\right) \tag{6.7}$$

cancels the nonlinearity and gives the closed loop system the desired linear dynamics [Isidori, 1989]

$$\dot{x} = \begin{bmatrix} 0 & I_{n-1\times n-1} \\ & -L^T \end{bmatrix} x + \begin{bmatrix} 0_{n-1\times 1} \\ 1 \end{bmatrix} r := A_m x + b_m r \tag{6.8}$$

If the functions $f(x)$ and $g(x)$ are not known, one approach is to obtain approximate models $\hat{f}(x)$ and $\hat{g}(x)$ from an identification experiment. The approximations $\hat{f}(x)$ and $\hat{g}(x)$, can for instance be in terms of fuzzy systems whose parameters are optimized to fit the identification data. Using a certainty equivalence approach, we may then try the control

$$u = \frac{1}{\hat{g}(x)} \left(-\hat{f}(x) + L^T x + b_m r \right) \tag{6.9}$$

In this case the cancellation of the nonlinearity is only approximate. □

Present approaches to model based fuzzy control in the above spirit simply assumes that the approximation of $f(x)$ and $g(x)$ is "sufficiently good". A more satisfactory approach would be to develop design algorithms that are robust with respect to bounded approximation errors.

Sliding Mode Control

A simple and highly robust control structure for uncertain nonlinear systems is the so called sliding mode controller [Utkin, 1977] [Slotine and Li, 1991], [Hung *et al.*, 1993]. The main idea behind sliding mode control is to transform the problem of stabilizing an nth order system (which is hard) into the problem of stabilizing a 1st order system (which is easier).

Sliding mode control can be demonstrated on the problem of globally stabilizing the system (6.6). Let $x(t)$ be the n-dimensional system state vector and consider the scalar function $s(x,t) = c^T x(t)$. This function defines a surface S of dimension n-1 in the system state space

$$S : s(x,t) = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n = 0 \tag{6.10}$$

Substituting the n-1 first state equations of (6.6) into (6.10), we have

$$s(x,t) = c_1 x_1 + c_2 \dot{x}_1 + \cdots + c_n x^{(n-1)} = 0 \tag{6.11}$$

Consequently, S also defines an ordinary differential equation in x_1. If we define the surface S so that the equation (6.11) is exponentially stable, the system state is guaranteed to converge to the origin if it remains on the surface for all future times.

Thus, the n-dimensional problem of driving the state vector to origin has been transformed into the 1-dimensional problem to forcing the system state onto the surface S and keeping the state on the surface. Forcing the system state to the sliding surface is equivalent of forcing the scalar $s(x;t)$ to zero in finite time. This can be accomplished by finding a control u such that the Lyapunov-like "sliding condition"

$$\frac{1}{2}\frac{d}{dt} s^2(x;t) \leq -\eta |s(x;t)| \tag{6.12}$$

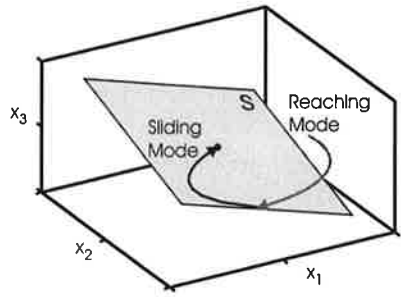


Figure 6.4 State trajectories for a system under sliding mode control

with $\eta > 0$ is fulfilled. Such a control forces the system state to the sliding surface S, and once on the surface the system state remains there, obeying the dynamics of (6.11). This is illustrated in Figure 6.4.

Next, we give an example of how sliding mode control can be used to guarantee stability for a nonlinear system in the presence of approximation errors.

EXAMPLE 6.3.3 (SLIDING CONTROL USING A FUZZY MODEL)
Consider the problem of stabilizing the following nonlinear system

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x) + u\end{aligned}$$

Assume that a mathematical model of $f(x)$ is unknown, but that we have some heuristic knowledge that we can formulate into a fuzzy model $\hat{f}(x)$. We can not expect this approximate model to be perfect, but assume that we can bound the approximation error by a function $F(x)$

$$|f(x) - \hat{f}(x)| \leq F(x)$$

A sliding mode controller can be designed for this system in the following steps. Define the sliding surface to be

$$S : s(x,t) = \begin{bmatrix} \lambda & 1 \end{bmatrix} x = \lambda x_1 + \dot{x}_1 = 0$$

with $\lambda > 0$. If the system state is on the sliding surface, it remains on the surface provided that $\dot{s} = 0$, i.e.

$$\dot{s} = (\lambda \dot{x}_1 + \dot{x}_2) = (\lambda x_2 + f(x) + u) = 0$$

Since we do not know $f(x)$, we use the control

$$u(x) = \underbrace{-\hat{f}(x) - \lambda x_2}_{\hat{u}} - \underbrace{k \operatorname{sgn}(s)}_{u^*} \tag{6.13}$$

The first part, $\hat{u}$, of the control is a certainty equivalence control (compare with (6.9)). To ensure that the state vector is forced onto the surface, we have added a second term, u^*, discontinuous over the sliding surface. If we set

$$k = F(x) + \eta$$

we can verify that this control satisfies the sliding condition (6.12):

$$\frac{1}{2}\frac{d}{dt}s^2 = \dot{s}s = \left(f(x) - \hat{f}(x)\right) s - k\,\mathrm{sgn}(s)s \leq -\eta|s|$$

□

From (6.13), we can see that the system state is directed towards the sliding surface "from both sides", by switching the control whenever the state trajectory crosses the sliding surface. In presence of noise or unmodelled dynamics, this results in a "chattering" control, inhibiting the direct implementation of sliding mode control. In practice, chattering can be eliminated by approximating the switching $\mathrm{sgn}(s)$ by some smooth function, resulting in a "boundary layer" around the sliding surface.

It has been suggested to design fuzzy controllers by approximating the function (6.13) by a fuzzy system. It is not clear, however, why one would like to approximate the compact formula (6.13) by a fuzzy system, increasing the number of system parameters and the computational requirements.

Compensation of Static Nonlinearities and Static Scheduling

In some cases, the main nonlinearities of a process are static nonlinearities on the input and output of the system, whereas the system dynamics is linear. In these situations, we can use the approximation capabilities of fuzzy systems to compensate for these nonlinearities.

EXAMPLE 6.3.4 (INPUT NONLINEARITIES)
Consider a SISO plant with linear dynamics and an input nonlinearity

$$\dot{x} = Ax + b \cdot g(u)$$

One approach to control this systems is to momentarily disregard the input nonlinearity and design a linear controller

$$v = L(x)$$

for the linear plant. We can then approximate the inverse (which may, or may not exist) of the input nonlinearity using a fuzzy system

$$f(v) \approx g^{-1}(v)$$

and apply the control

$$u = f(L(x))$$

□

In some cases, it can be motivated to introduce static nonlinearities at the input of a linear system, as described by the next example.

EXAMPLE 6.3.5 (STATIC SCHEDULING)
In climate control systems, the temperature dynamics of a room can be modeled as a linear system

$$\dot{x} = Ax + bq$$

where q is the heat delivered by the air conditioner (AC). It is straight forward to design a linear controller for this system. Typically, this controller measures the room temperature and computes the heat that the AC must provide. However, we can not control the heat of the delivered air stream directly. The heat contained in the air delivered by the AC is proportional to the product of the air mass flow, w, and the temperature, T, of the air stream, i.e.

$$q \propto w \cdot T$$

We may then use fuzzy logic rules to design a schedule

$$\begin{aligned} w &= f_1(q;p) \\ T &= f_2(q;p) \end{aligned}$$

that for a desired heat q and auxiliary operating conditions p suggests the most comfortable combination of air mass flow and air temperature. This control structure is illustrated in Figure 6.5.

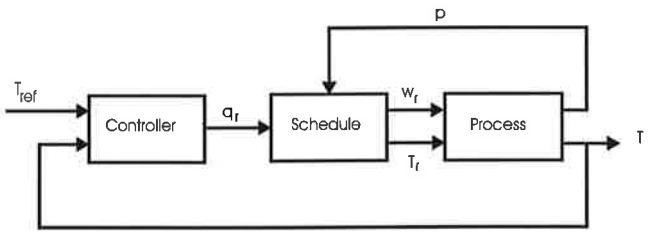


Figure 6.5 Static scheduling using fuzzy systems.

□

Adaptive Fuzzy Control

The main idea of an adaptive controller is to let some of the controller parameters be adjustable, and incorporate an adjustment mechanism into the control algorithm.

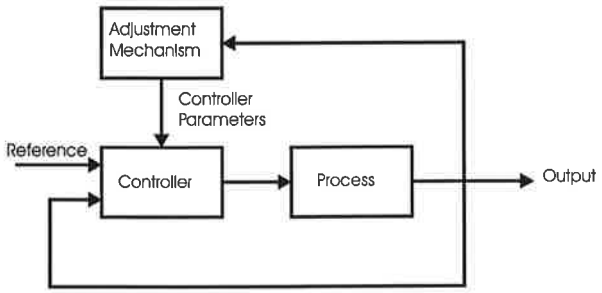


Figure 6.6 Adaptive control.

Adaptive control is useful when the process dynamics change with time. It is a tempting challenge to try to derive adaptive fuzzy control algorithms. However, moving from fixed control systems and time-invariant plants to adaptive control systems and time-varying plants is a major step. Further, if we use approximate system descriptions like fuzzy systems, we have to face the fact that we can never expect the fuzzy system to approximate the actual

system perfectly. Similar to linear adaptive control, we can expect the adaption mechanism to compensate for the approximation errors to some extent. Still, adaptive fuzzy control algorithms should be based on design methods that guarantee stability under bounded approximation errors in the fixed controller case. Up to this date, there are no completely satisfying solutions to the adaptive fuzzy control problem. The shortcomings of current adaptive fuzzy control approaches is demonstrated in the following example.

EXAMPLE 6.3.6 (ADAPTIVE FUZZY CONTROL)
Consider the problem of designing a Model Reference Adaptive System (MRAS) for the nonlinear system (6.6) based on the idea of nonlinear pole placement. For simplicity, assume the function $g(x)$ to be known, while $f(x)$ can only be described approximately by means of the fuzzy system $\hat{f}(x)$.

Recall that in the MRAS, the desired system dynamics is specified by a reference model

$$\dot{x}_m = A_m x_m + b_m r \tag{6.14}$$

and that the controller parameters are adjusted based on the errors between the outputs of this model and the closed loop system.

Let the prescribed model A_m and b_m have the structure defined in (6.8), and assume that we have access to the full state vector of (6.6). Then the control

$$u = \frac{1}{g(x)}\left(-\hat{f}(x) + y_m^{(n)} + L^T e\right)$$

ideally cancels the nonlinearity and assigns the desired linear dynamics to the closed loop.

Inspired by linear adaptive control [Åström and Wittenmark, 1989], an adaptive fuzzy controller can be derived in the following steps [Wang, 1994]. Using the same notation as in the identification section, we re-write the fuzzy system approximation $\hat{f}(x)$ of $f(x)$ in vector form

$$\hat{f}(x) = \phi^T(x)C$$

As a Lyapunov function candidate, we use

$$V(e, C) = \frac{1}{2}e^T P e + \frac{1}{2\gamma}(C - C^*)^T (C - C^*) \tag{6.15}$$

In this expression, C^* denotes the "optimal" fuzzy system parameters in some sense, P is the positive definite solution to the Lyapunov equation

$$A_m^T P + P A_m = -Q$$

and γ is a scalar constant. The time derivative of (6.15) along (6.6) is

$$\begin{aligned}\frac{d}{dt}V(e, C) = &-\frac{1}{2}e^T Q e + e^T P b_m \left(f(x) - \phi^T C^*\right) \\ &+ \frac{1}{2\gamma}(C - C^*)^T \left(\frac{dC}{dt} + \gamma x^T P b_m \phi^T(x)\right)\end{aligned} \tag{6.16}$$

If we choose the parameter update law

$$\frac{dC}{dt} = -\gamma e^T P b_m \phi^T(x)$$

the last term of (6.16) is cancelled. However, stability and convergence of the adaptive system can not be guaranteed without further investigations of the effects of the non-vanishing approximation error $f(x) - \phi^T(x)C^*$.

See [Wang, 1994], for the case when both $f(x)$ and $g(x)$ are approximated by fuzzy systems.

□

6.4. Supervisory Fuzzy Control

This far, we have only been concerned with fuzzy systems for direct feedback control. However, when the complexity of a control system increases, the need for supervision and planning becomes increasingly important. A controller incorporating supervisory and planning stages can be structured hierarchically as illustrated in Figure 6.7.

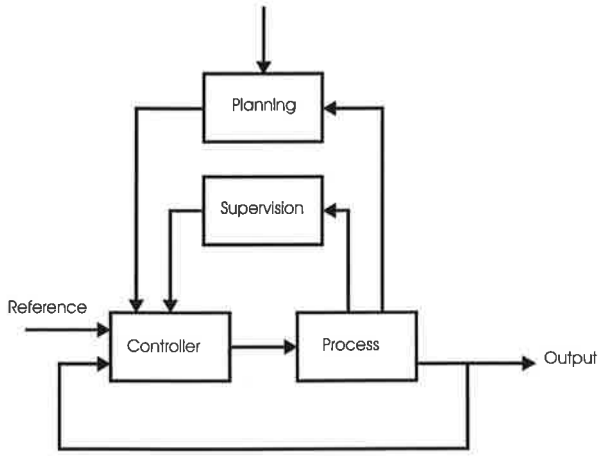


Figure 6.7 Supervisory control structure.

Loop level controllers are often designed based on a physical model of the process under control. When the process model is not fully known, we have seen how it is sometimes possible to use fuzzy systems to approximate parts of the model, and then use these approximate models in the control design. When we move up in the hierarchy, heuristics and human decision making becomes increasingly important. It is thus natural to describe the supervisory functions of a controller using logic rules.

Analysis of the combination of supervisory logic and direct controller is a complicated task, since the two hierarchical levels are hard to formulate in the same framework. One approach is to model the hierarchical system using a combination of differential equations and discrete events. Another approach, as suggested in [Wang, 1995], is to formulate the supervisory control actions using fuzzy IF-THEN rules.

$$R^{(l)}\text{IF } s_1 \text{ IS } A_1^{(l)} \text{ AND } \ldots \text{ AND } s_n \text{ IS } A_n^{(l)} \text{ THEN } u_s^{(l)} \text{ IS } B^{(l)}$$

Since a fuzzy system can be recast into a nonlinear mapping

$$u_s = f(s_1, \ldots, s_n)$$

the combination of the supervisory controller and loop level controller can be analyzed using standard mathematical tools for the differential equation framework.

6.5. Summary

In this chapter, we have indicated how fuzzy systems can be used for control. Although fuzzy logic has been used for a wide variety of control problems, we have presented a general fuzzy controller structure that encompasses most of the fuzzy controllers known to the authors of this report. It is important to stress that a fuzzy controller contains a fuzzy system mapping, but in many cases also dynamic components. To obtain good control performance, the filters and the fuzzy system mapping are equally important.

Understanding alternative control approaches is often a matter of "bringing it all back home". Fuzzy control can be related to other methods by noting that a fuzzy system mapping can in some cases be re-cast into a compact mathematical formula. In large, this property forms the basis for what we have presented in this chapter; fuzzy system models, model based fuzzy control schemes, adaptive fuzzy controllers and supervisory fuzzy controllers.

The one-to-one relation between a fuzzy system and the nonlinear formula is also the key to appreciating fuzzy control. Prior knowledge of control strategies can be expressed in terms of linguistic IF-THEN rules. These rules can be transformed into a nonlinear mapping for which some analysis and tuning methods from exisiting nonlinear control theory applies. After the parameters of the nonlinear formula have been tuned, the formula can be re-cast into a fuzzy system rule base. These altered linguistic rules can then be inspected by an expert.

For further material on fuzzy systems and neural networks for control, see [Zbikowski *et al.*, 1994], [Jang and Sun, 1995] and [Wang, 1994].

7. Bibliography

Åström, K. J. and B. Wittenmark (1989): *Adaptive Control.* Addison-Wesley, Reading, Massachusetts.

Åström, K. J. and B. Wittenmark (1995): *Adaptive Control.* Addison-Wesley, Reading, Massachusetts, second edition.

Brown, F. M. (1990): *Boolean Reasoning: The Logic of Boolean Equations.* Kluwer Academic Publishers.

Brown, M. and C. Harris (1995): *Neuro-Fuzzy Adaptive Modeling Control.* Prentice Hall.

Castro, J. (1995): "Fuzzy logic controllers are universal approximators." *IEEE Transactions on Systems, Man and Cybernetics*, April, pp. 629–635.

deBoor, C. (1978): *A practical guide to Splines.* Applied Math. Sciences Vol.27. Springer Verlag, New York.

Driankov, D., H. H. and M.Reinfrank (1993): *An Introduction to Fuzzy Control.* Springer Verlag, Berlin Heidelberg.

Dubois, D. and H. Prade (1980): *Fuzzy Sets and Systems: Theory and Applications.* Academic Press, Inc., Orlando, FL.

Federico Girosi, Michael Jones, T. P. (1993): "Priors, Stabilizers and Basis Functions: from regularization to radial, tensor and additive splines." Technical Report A.I. Memo No. 1430, M.I.T. Artificiall Intelligence Laboratory.

Fletcher, R. (1987): *Practical Methods of Optimization.* John Wiley & Sons, second edition.

Hung, J. Y., W. Gao, and J. C. Hung (1993): "Variable structure control: A survey." *IEEE Transactions on Industrial Electronics*, February, pp. 2–21.

Isidori, A. (1989): *Nonlinear Control Systems: an Introduction.* Springer-Verlag.

Jang, J.-S. R. and C.-T. Sun (1995): "Neuro-fuzzy modeling and control." *Proceedings of the IEEE*, March.

Johansen, T. (1995): *Operating Regime based Process Modeling and Identification.* PhD thesis ITK-Report-94-109-W, Norvegian Institute of Technology.

Johansson, M. (1993): "Nonlinearities and interpolation in fuzzy control." Master thesis ISRN LUTFD2/TFRT--5491--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Johansson, M. (1994a): "The nonlinear nature of fuzzy control." In *Proceedings of the 4.e Dortmunder Fuzzy Tage*, Dortmund, Germany.

Johansson, M. (1994b): "Rule based interpolating control—fuzzy and its alternatives." In *Preprints 2nd IFAC Workshop on Computer Software Structures Integrating AI/KBS Systems in Process Control*, pp. 205–210, Lund, Sweden.

Johansson, R. (1993): *System Modeling and Identification*. Prentice Hall, Englewood Cliffs, New Jersey.

Juditsky, A., H. Hjalmarsson, A. Benveniste, B. Deylon, and L.Ljung (1995): "Nonlinear black-box modeling in system identification: Mathematical foundations." *Automatica*, **31**.

Katayama, R., Y.Kajitani, and Y. Nishida (1994): "A self generating and tuning method for fuzzy modelling using interior penalty method and its application to knowledge acquistion of fuzzy controller." In Kandel and Langholz, Eds., *Fuzzy Control Systems*, pp. 197–224. CRC Press.

Klenk, V. (1989): *Understanding Symbolic Logic*. Prentice Hall, Englewood Cliffs, N.J.

Lee, C. (1990): "Fuzzy logic in control systems: Fuzzy logic controller part i and ii." *IEEE Transactions on Man, Systems and Cybernetics*, **20:2**, pp. 404–435.

Pedrycz, W. (1989): *Fuzzy Control and Fuzzy Systems*. John Wiley & Sons, New York.

Sjöberg, J., Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P. Glorennec, H. Hjalmarsson, and A. Juditsky (1995): "Nonlinear black-box modeling in system identification: A unified overview." *Automatica*, **31**.

Slotine, J.-J. and W. Li (1991): *Applied Nonlinear Control*. Prentice Hall International.

Sontag, E. (1993): "Some topics in neural networks and control." Technical Report, Siemens Coporate Research.

Takagi, T. and M. Sugeno (1985): "Fuzzy identification of systems and its applications to modeling and control." *IEEE Transactions on Systems, Man and Cybernetics*, **No 15**, pp. 116–132.

Utkin, V. I. (1977): "Variable structure systems with sliding modes: a survey." *IEEE Transactions on Automatic Control*, **No 22**, pp. 212–222.

Wahba, G. (1995): "Generalization and regularization in nonlinear learning systems." In Arbib, Ed., *Handbook of Brain Theory and Neural Networks*, pp. 426–430. MIT Press.

Wang, L.-X. (1994): *Adaptive Fuzzy Systems and Control – Design and Stability Analysis*. Prentice Hall, Englewood Cliffs, N.J.

Wang, L.-X. (1995): "Modeling and Control of Hierarchical Systems using Rule Based Systems." . Submitted for publication.

Xiao-Jun Zeng, M. G. S. (1995): "Approximation theory of fuzzy systems – MIMO case." *IEEE Transactions on Fuzzy Systems*, **3:2**, pp. 219–235.

Zadeh, L. (1965): "Fuzzy sets." *Information and Control*, **No 8**, pp. 338–353.

Zbikowski, R., K. Hunt, A. Dzieliński, R. Murray-Smith, and P. Gawthorp (1994): "A Review of Advances in Neural Adaptive Control Systems." Technical Report, Daimler-Benz AG and University of Glasgow.