Fuzzy Logic Control Systems

During the past decade, fuzzy logic control (FLC), initiated by the pioneering work of Mamdani and Assilian [1975], has emerged as one of the most active and fruitful areas for research in the application of fuzzy set theory, fuzzy logic, and fuzzy reasoning. Its application ranges from industrial process control to medical diagnosis and securities trading. Many industrial and consumer products using this technology have been built, especially in Japan where FLC has achieved considerable success. In contrast to conventional control techniques, FLC is best utilized in complex ill-defined processes that can be controlled by a skilled human operator without much knowledge of their underlying dynamics.

The basic idea behind FLC is to incorporate the "expert experience" of a human operator in the design of the controller in controlling a process whose input-output relationship is described by a collection of fuzzy control rules (e.g., IF-THEN rules) involving linguistic variables rather than a complicated dynamic model. This utilization of linguistic variables, fuzzy control rules, and approximate reasoning provides a means to incorporate human expert experience in designing the controller.

In this chapter, we shall introduce the basic architecture, the design methodology, and the stability analysis of fuzzy logic controllers. Some practical application examples will also be discussed. We will find that FLC is strongly based on the concepts of fuzzy sets and relations, linguistic variables, and approximate reasoning introduced in the previous chapters.

7.1 BASIC STRUCTURE AND OPERATION OF FUZZY LOGIC CONTROL SYSTEMS

The typical architecture of a FLC is shown in Fig. 7.1, which is comprised of four principal components: *a fuzzifier, a fuzzy rule base, an inference engine,* and *a defuzzifier.* If the out-

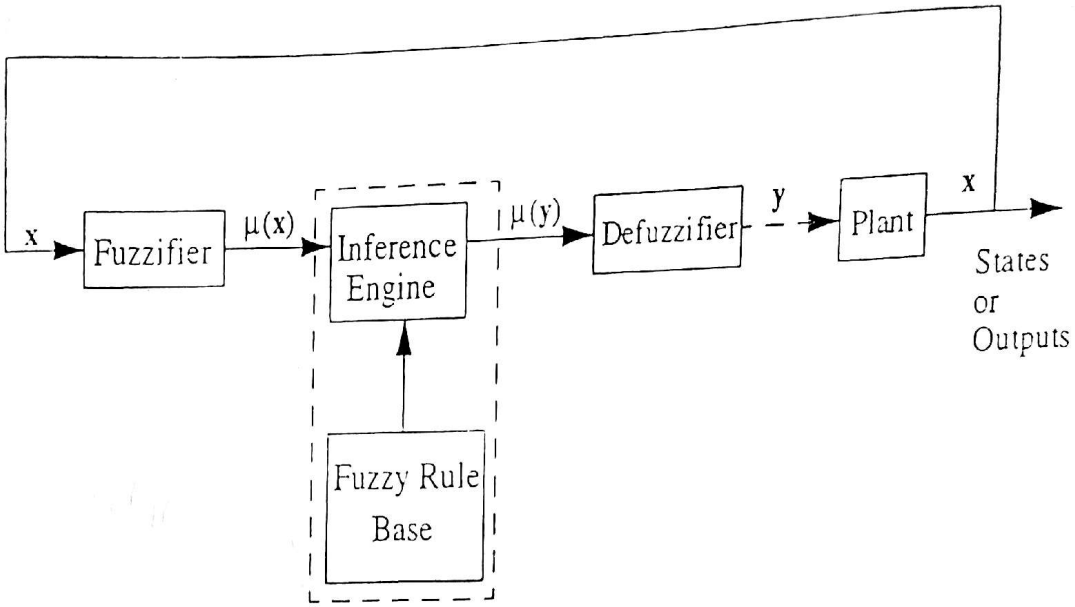

Figure 7.1 Basic architecture of a fuzzy logic controller (FLC).

put from the defuzzifier is not a control action for a plant, then the system is a fuzzy logic *decision* system. The fuzzifier has the effect of transforming crisp measured data (e.g., speed is 10 miles per hour) into suitable linguistic values (i.e., fuzzy sets, for example, speed is *too slow*). The fuzzy rule base stores the empirical knowledge of the operation of the process of the domain experts. The inference engine is the kernel of a FLC, and it has the capability of simulating human decision making by performing approximate reasoning to achieve a desired control strategy. The defuzzifier is utilized to yield a nonfuzzy decision or control action from an inferred fuzzy control action by the inference engine. More details about the operations of these components are described in the following sections.

7.1.1 Input-Output Spaces

The purpose of fuzzy logic controllers is to compute values of control (or action) variables from the observation or measurement of state variables of the controlled process such that a desired system performance is achieved. Thus, a proper choice of process state variables and control variables is essential to characterization of the operation of a fuzzy logic control system (FLCS) and has a substantial effect on the performance of a FLC. Expert experience and engineering knowledge play an important role during this state variables and control variables selection process. Typically, the input variables in a FLC are the state, state error, state error derivative, state error integral, and so on. Following the definition of linguistic variables, the input vector x which includes the input state linguistic variables x_i and the output state vector y which includes the output state (or control) linguistic variables y_i in Fig. 7.1 can be defined, respectively, as

$$\mathbf{x} = \{(x_i, U_i, \{T_{x_i}^1, T_{x_i}^2, \ldots, T_{x_i}^{k_i}\}, \{\mu_{x_i}^1, \mu_{x_i}^2, \ldots, \mu_{x_i}^{k_i}\})|_{i=1,\ldots,n}\}, \tag{7.1}$$

$$\mathbf{y} = \{(y_i, V_i, \{T_{y_i}^1, T_{y_i}^2, \ldots, T_{y_i}^{l_i}\}, \{\mu_{y_i}^1, \mu_{y_i}^2, \ldots, \mu_{y_i}^{l_i}\})|_{i=1,\ldots,m}\}, \tag{7.2}$$

where the input linguistic variables x_i form a fuzzy input space $U = U_1 \times U_2 \times \cdots \times U_n$ and the output linguistic variables y_i form a fuzzy output space $V = V_1 \times V_2 \times \cdots \times V_m$.

From Eqs. (7.1) and (7.2), we observe that an input linguistic variable x_i in a universe of discourse U_i is characterized by $T(x_i) = \{T_{x_i}^1, T_{x_i}^2, \ldots, T_{x_i}^{k_i}\}$ and $\mu(x_i) = \{\mu_{x_i}^1, \mu_{x_i}^2, \ldots, \mu_{x_i}^{k_i}\}$, where $T(x_i)$ is the *term set* of x_i, that is, the set of names of linguistic values of x_i with each value $T_{x_i}^{k_i}$ being a fuzzy number with membership function $\mu_{x_i}^{k_i}$ defined on U_i. So $\mu(x_i)$ is a semantic rule for associating each value with its meaning. For example, if x_i indicates speed, then $T(x_i) = \{T_{x_i}^1, T_{x_i}^2, T_{x_i}^3\}$ may be "Slow," "Medium," and "Fast." Similarly, an output linguistic variable y_i is associated with a term set $T(y_i) = \{T_{y_i}^1, T_{y_i}^2, \ldots, T_{y_i}^{l_i}\}$ and $\mu(y_i) = \{\mu_{y_i}^1, \mu_{y_i}^2, \ldots, \mu_{y_i}^{l_i}\}$. The size (or cardinality) of a term set $|T(x_i)| = k_i$ is called the *fuzzy partition* of x_i. The fuzzy partition determines the granularity of the control obtainable from a FLC. Figure 7.2(a) depicts two fuzzy partitions in the same normalized universe $[-1, +1]$. For a two-input FLC, the fuzzy input space is divided into many overlapping grids [see Fig. 7.2(b)]. Furthermore, the fuzzy partitions in a fuzzy input space determine the *maximum* number of fuzzy control rules in a FLCS. For example, in the case of a two-input–one-output fuzzy logic control system, if $|T(x_1)| = 3$ and $|T(x_2)| = 7$, then the maximum number of fuzzy control rules is $|T(x_1)| \times |T(x_2)| = 21$. The input membership functions $\mu_{x_i}^k$, $k = 1, 2, \ldots, k_i$, and the output membership functions $\mu_{y_i}^l$, $l = 1, 2, \ldots, l_i$, used in a FLC are usually parametric functions such as triangular functions, trapezoidal functions, and bell-shaped functions. Triangular functions and the trapezoidal functions can be represented by *L-R* fuzzy numbers, while bell-shaped membership functions can be defined as

$$\mu_{x_i}(x) = \exp\left(\frac{-(x - m_i)^2}{2\sigma_i^2}\right), \tag{7.3}$$

where m_i and σ_i specify the mean location and the width of the bell-shaped function, respectively. Proper fuzzy partitioning of input and output spaces and a correct choice of membership functions play an essential role in achieving a successful FLC design. Unfortunately, they are not deterministic and have no unique solutions. Traditionally, a heuristic trial-and-error procedure is usually used to determine an optimal fuzzy partition. Furthermore, the choice of input and output membership functions is based on subjective decision criteria and relies heavily on time-consuming trial and error. A promising approach to automating and speeding up these design choices is to provide a FLC with the ability to learn its input and output membership functions and the fuzzy control rules. This will be explored later in Part III of this book (see Chap. 19).

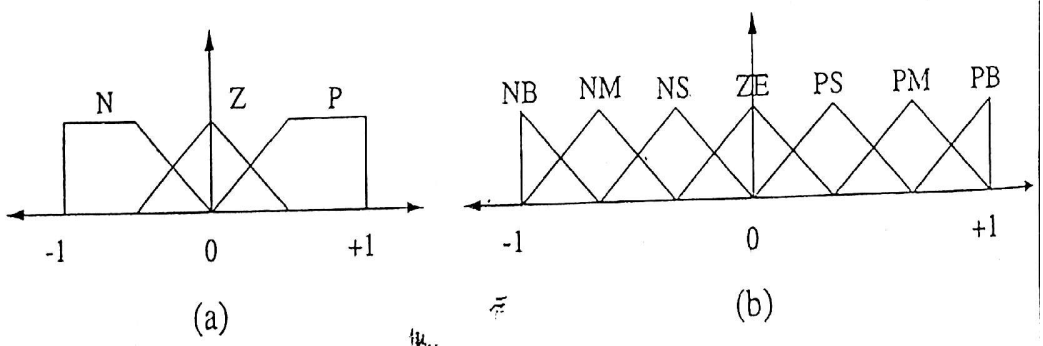

Figure 7.2 Diagrammatic representation of a fuzzy partition. (a) Coarse fuzzy partition with three terms: N, negative; ZE, zero; P, positive. (b) Finer fuzzy partition with seven terms: NB, negative big; NM, negative medium; NS, negative small; ZE, zero; PS, positive small; PM, positive medium; PB, positive big.

7.1.2 Fuzzifier

A fuzzifier performs the function of fuzzification which is a subjective valuation to transform measurement data into valuation of a subjective value. Hence, it can be defined as a mapping from an observed input space to labels of fuzzy sets in a specified input universe of discourse. Since the data manipulation in a FLC is based on fuzzy set theory, fuzzification is necessary and desirable at an early stage. In fuzzy control applications, the observed data are usually crisp (though they may be corrupted by noise). A natural and simple fuzzification approach is to convert a crisp value x_0 into a fuzzy singleton A within the specified universe of discourse. That is, the membership function of A, $\mu_A(x)$, is equal to 1 at the point x_0, and zero at other places. In this case, for a specific value $x_i(t)$ at time t, it is mapped to the fuzzy set $T_{x_i}^1$ with degree $\mu_{x_i}^1(x_i(t))$ and to the fuzzy set $T_{x_i}^2$ with degree $\mu_{x_i}^2(x_i(t))$, and so on. This approach is widely used in FLC applications because it greatly simplifies the fuzzy reasoning process. In a more complex case where observed data are disturbed by random noise, a fuzzifier should convert the probabilistic data into fuzzy numbers, that is, fuzzy (possibility) data. For this, Dubois and Prade [1985a] defined a bijective transformation which transforms a probability measure into a possibility measure by using the concept of the degree of necessity. In large-scale systems, some observations relating to the behavior of such systems are precise, others are measurable only in a statistical sense, and some, referred to as "hybrids," require both probabilistic and possibilistic modes of characterization.

7.1.3 Fuzzy Rule Base

Fuzzy control rules are characterized by a collection of fuzzy IF-THEN rules in which the preconditions and consequents involve linguistic variables. This collection of fuzzy control rules (or fuzzy control statements) characterizes the simple input-output relation of the system. The general form of the fuzzy control rules in the case of multi-input–single-output systems (MISO) is:

$$R^i: \text{IF } x \text{ is } A_i, \ldots, \text{ AND } y \text{ is } B_i, \text{ THEN } z = C_i, \qquad i = 1, 2, \ldots, n, \tag{7.4}$$

where $x, \ldots, y$, and z are linguistic variables representing the process state variables and the control variable, respectively, and $A_i, \ldots, B_i$, and C_i are the linguistic values of the linguistic variables $x, \ldots, y$, and z in the universes of discourse $U, \ldots, V$, and W, respectively. A variant of this type is that the consequent is represented as a function of the process state variables $x, \ldots, y$, that is

$$R^i: \text{IF } x \text{ is } A_i, \ldots, \text{ AND } y \text{ is } B_i, \text{ THEN } z = f_i(x, \ldots, y), \tag{7.5}$$

where $f_i(x, \ldots, y)$ is a function of the process state variables $x, \ldots, y$. The fuzzy control rules in Eqs. (7.4) and (7.5) evaluate the process state (i.e., state, state error, state error integral, and so on) at time t and compute and decide the control actions as a function of the state variables $(x, \ldots, y)$. It is worthwhile to point out that both fuzzy control rules have linguistic values as inputs and either linguistic values [as in Eq. (7.4)] or crisp values [as in Eq. (7.5)] as outputs.

7.1.4 Inference Engine

This is the kernel of the FLC in modeling human decision making within the conceptual framework of fuzzy logic and approximate reasoning. As mentioned in Section 6.3.1, the

generalized modus ponens (forward data-driven inference) in Eq. (6.27) plays an especially important role in this context. For application to fuzzy reasoning in FLCs, the generalized modus ponens in Eq. (6.27) can be rewritten as follows:

$$\begin{array}{ll} \text{Premise 1:} & \text{IF } x \text{ is } A, \text{ THEN } y \text{ is } B \\ \text{Premise 2:} & x \text{ is } A' \\ \hline \text{Conclusion:} & y \text{ is } B'. \end{array} \tag{7.6}$$

where A, A', B, and B' are fuzzy predicates (fuzzy sets or relations) in the universal sets U, U, V, and V, respectively. In general, a fuzzy control rule [e.g., premise 1 in Eq. (7.6)] is a fuzzy relation which is expressed as a fuzzy implication $R = A \rightarrow B$. According to the compositional rule of inference in Eq. (6.25), conclusion B' in Eq. (7.6) can be obtained by taking the composition of fuzzy set A' and the fuzzy relation (here the fuzzy relation is a fuzzy implication) $A \rightarrow B$:

$$B' = A' \circ R = A' \circ (A \rightarrow B). \tag{7.7}$$

In addition to the definitions of fuzzy composition and implication given in Eqs. (6.26) and (6.28), there are four types of compositional operators that can be used in the compositional rule of inference. These correspond to the four operations associated with the t-norms:

- Max-min operation [Zadeh, 1973],
- Max product operation [Kaufmann, 1975],
- Max bounded product (max $-\odot$) operation [Mizumoto, 1981],
- Max drastic product (max $-\hat{\wedge}$) operation [Mizumoto, 1981],

where bounded product and drastic product operations are stated in Eqs. (2.58) and (2.59), respectively. In FLC applications, max-min and max product compositional operators are the most commonly and frequently used because of their computational simplicity and efficiency. Let max $-\star$ represent any one of the above four composition operations. Then Eq. (7.7) becomes

$$\begin{aligned} B' &= A' \star R = A' \star (A \rightarrow B), \\ \mu_{B'}(v) &= \sup_u \{ \mu_{A'}(u) \star \mu_{A \rightarrow B}(u, v) \}, \end{aligned} \tag{7.8}$$

where $*$ denotes the t-norm operations such as min, product, bounded product, and drastic product operations. As for the fuzzy implication $A \rightarrow B$, there are nearly 40 distinct fuzzy implication functions described in the existing literature. Table 7.1 provides a list of several fuzzy implication rules commonly used in a FLC [Mizumoto, 1988].

It is observed that the fuzzy implication rules defined in Table 7.1 are generated from the *fuzzy conjunction, fuzzy disjunction,* or *fuzzy implication* by employing various t-norms or t-conorms. The first four fuzzy implications, R_c, R_p, R_{bp}, and R_{dp}, are all t-norms. For example, Mamdani's min fuzzy implication R_c is obtained if the intersection operator is used in the fuzzy conjunction. Larsen's product fuzzy implication R_p is obtained if the algebraic product is used in the fuzzy conjunction. R_{bp} and R_{dp} are obtained if the bounded product and the drastic product are used in the fuzzy conjunction,

TABLE 7.1 Various Fuzzy Implication Rules

Rule of Fuzzy Implication	Implication Formulas	Fuzzy Implication: $\mu_{A\to B}(u, v)$
R_m: min operation [Mamdani]	$a \to b = a \wedge b$	$= \mu_A(u) \wedge \mu_B(v)$
R_p: product operation [Larsen]	$a \to b = a \cdot b$	$= \mu_A(u) \cdot \mu_B(v)$
R_b: bounded product	$a \to b = 0 \vee (a + b - 1)$	$= 0 \vee [\mu_A(u) + \mu_B(v) - 1]$
R_{dp}: drastic product	$a \to b = \begin{cases} a, & b = 1 \\ b, & a = 1 \\ 0, & a, b < 1 \end{cases}$	$= \begin{cases} \mu_A(u), & \mu_B(v) = 1 \\ \mu_B(v), & \mu_A(u) = 1 \\ 0, & \mu_A(u), \mu_B(v) < 1 \end{cases}$
R_a: arithmetic rule [Zadeh]	$a \to b = 1 \wedge (1 - a + b)$	$= 1 \wedge (1 - \mu_A(u) + \mu_B(v))$
R_m: max-min rule [Zadeh]	$a \to b = (a \wedge b) \vee (1 - a)$	$= (\mu_A(u) \wedge \mu_B(v)) \vee (1 - \mu_A(u))$
R_s: standard sequence	$a \to b = \begin{cases} 1, & a \le b \\ 0, & a > b \end{cases}$	$= \begin{cases} 1, & \mu_A(u) \le \mu_B(v) \\ 0, & \mu_A(u) > \mu_B(v) \end{cases}$
R_c: Boolean fuzzy implication	$a \to b = (1 - a) \vee b$	$= (1 - \mu_A(u)) \vee \mu_B(v)$
R_g: Gödelian logic	$a \to b = \begin{cases} 1, & a \le b \\ b, & a > b \end{cases}$	$= \begin{cases} 1, & \mu_A(u) \le \mu_B(v) \\ \mu_B(v), & \mu_A(u) > \mu_B(v) \end{cases}$
R_3: Goguen's fuzzy implication	$a \to b = \begin{cases} 1, & a \le b \\ b/a, & a > b \end{cases}$	$= \begin{cases} 1, & \mu_A(u) \le \mu_B(v) \\ \mu_B(v)/\mu_A(u), & \mu_A(u) > \mu_B(v) \end{cases}$

respectively. Furthermore, we note that Zadeh's arithmetic rule R_a follows Eq. (2.78) by using the bounded sum operator, and Zadeh's max-min rule R_m follows Eq. (2.79) by using the intersection and union operators. Other fuzzy implication rules in Table 7.1 can be obtained employing fuzzy implication definitions in Eqs. (2.78)–(2.82) using various t-norms and t-conorms.

Example 7.1

This example illustrates the various fuzzy implications in Table 7.1 graphically. Assume fuzzy set A' is a singleton at u_0; that is, $\mu_{A'}(u_0) = 1$ and $\mu_{A'}(u) = 0$ for $u \neq u_0$. Let A and B be fuzzy sets in U and V, respectively, as in Fig. 7.3. Then the consequent B' of Eq. (7.6) at $A' = u_0$ under the fuzzy implications $A \to B$ in Table 7.1 is depicted in Fig. 7.4, where $\mu_A(u_0) = a$ with $a = 0.3$ (dotted line) and $a = 0.7$ (solid line). Figure 7.4 is obtained from the equations in the third column of Table 7.1 by setting $\mu_A(u) = \mu_A(u_0)$ in those equations.

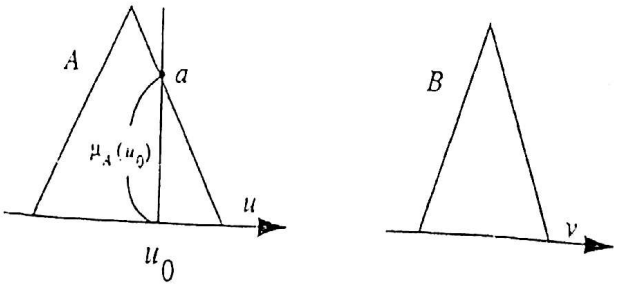

Figure 7.3 Fuzzy sets A and B in Example 7.1.

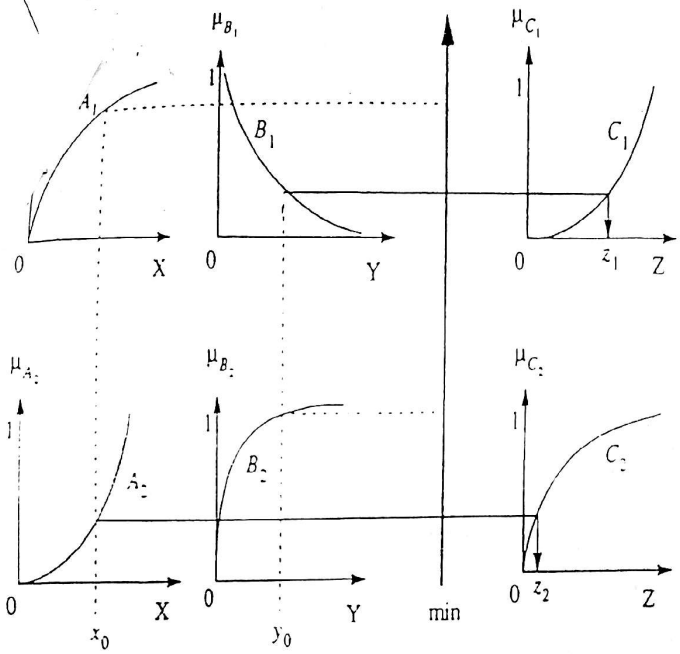

Figure 7.7 Diagrammatic representation of fuzzy reasoning of the second type.

4. *Fuzzy reasoning of the third type—The consequent of a rule is a function of input linguistic variables*: In this mode of reasoning, the ith fuzzy control rule is of the form of Eq. (7.5). For simplicity, assume that we have two fuzzy control rules as follows:

$$R^1\text{: IF } x \text{ is } A_1 \text{ AND } y \text{ is } B_1, \text{ THEN } z \text{ is } f_1(x, y),$$

$$R^2\text{: IF } x \text{ is } A_2 \text{ AND } y \text{ is } B_2, \text{ THEN } z \text{ is } f_2(x, y).$$

The inferred values of the control action from the first and second rules are $\alpha_1 f_1(x_0, y_0)$ and $\alpha_2 f_2(x_0, y_0)$, respectively. Consequently, a crisp control action is given by

$$z_0 = \frac{\alpha_1 f_1(x_0, y_0) + \alpha_2 f_2(x_0, y_0)}{\alpha_1 + \alpha_2}. \tag{7.29}$$

This method was proposed by Takagi and Sugeno [1983] and has been applied to guide a model car smoothly along a crank-shaped track [Sugeno and Nishida, 1985] and to park a model car in a garage [Sugeno and Murakami, 1985].

7.1.5 Defuzzifier

Defuzzification is a mapping from a space of fuzzy control actions defined over an output universe of discourse into a space of nonfuzzy (crisp) control actions. This process is necessary because in many practical applications crisp control action is required to actuate the control. Thus, a defuzzifier is necessary when fuzzy reasoning of the first type [Eqs. (7.24) and (7.26)] is used.

A defuzzification strategy is aimed at producing a nonfuzzy control action that best represents the possibility distribution of an inferred fuzzy control action. Unfortunately, there is no systematic procedure for choosing a defuzzification strategy. Two commonly used methods of defuzzification are the *center of area* (COA) method and the *mean of maximum* (MOM) method.

The widely used COA strategy generates the center of gravity of the possibility distribution of a control action. In the case of a discrete universe, this method yields

$$z^*_{\text{COA}} = \frac{\sum_{j=1}^{n} \mu_C(z_j)\, z_j}{\sum_{j=1}^{n} \mu_C(z_j)}, \tag{7.30}$$

where n is the number of quantization levels of the output, z_j is the amount of control output at the quantization level j, and $\mu_C(z_j)$ represents its membership value in the output fuzzy set C. If the universe of discourse is continuous, then the COA strategy generates an output control action of

$$z^*_{\text{COA}} = \frac{\int_z \mu_C(z)\, z\, dz}{\int_z \mu_C(z)\, dz}. \tag{7.31}$$

The MOM strategy generates a control action that represents the mean value of all local control actions whose membership functions reach the maximum. In the case of a discrete universe, the control action may be expressed as

$$z^*_{\text{MOM}} = \sum_{j=1}^{m} \frac{z_j}{m}, \tag{7.32}$$

where z_j is the support value at which the membership function reaches the maximum value $\mu_C(z_j)$ and m is the number of such support values.

Of these two commonly used defuzzification strategies, the COA strategy has been shown to yield superior results [Braae and Rutherford, 1978]. Furthermore, the MOM strategy yields a better transient performance, while the COA strategy yields a better steady-state performance (lower mean square error) [Lee, 1990].

Yager and Filev [Yager, 1992c; Filev and Yager, 1991] proposed a generalized approach to defuzzification based on the basic defuzzification distribution (BADD) transform. They view the defuzzification process as first converting the inferred fuzzy control actions on z into a probability distribution, BADD, on z and then taking the expected value. More specifically, in the case of a discrete universe, the control action is expressed as

$$z_0 = \sum_{j=1}^{n} z_i p_i, \tag{7.33}$$

where p_i is the BADD defined as

$$p_i = \frac{\mu_C^{\alpha}(z_i)}{\sum_{j=1}^{n} \mu_C^{\alpha}(z_j)} \tag{7.34}$$

and is parameterized by a parameter $\alpha \in [0, \infty]$. For different values of α, one obtains different defuzzification strategies: (1) If $\alpha = 1$, then the defuzzification strategy is reduced to the COA method. (2) If $\alpha \to \infty$, then the defuzzification strategy is reduced to the MOM method. (3) If $\alpha = 0$, then $p_i = 1/n$; where n is the cardinality of z.

One immediate implication of the introduction of BADD is that we can provide an adaptive learning scheme to obtain the optimal defuzzification parameter α. The value of α can be interpreted as some kind of measure of confidence in the controller rule base output. We see that $\alpha = 0$ corresponds to no confidence, and the information supplied by the rule base is completely discounted. If $\alpha = 1$, we take the information supplied by the controller at face value. This can be interpreted as normal confidence. If $\alpha = \infty$, then we place extremely high confidence in the information supplied by the controller.

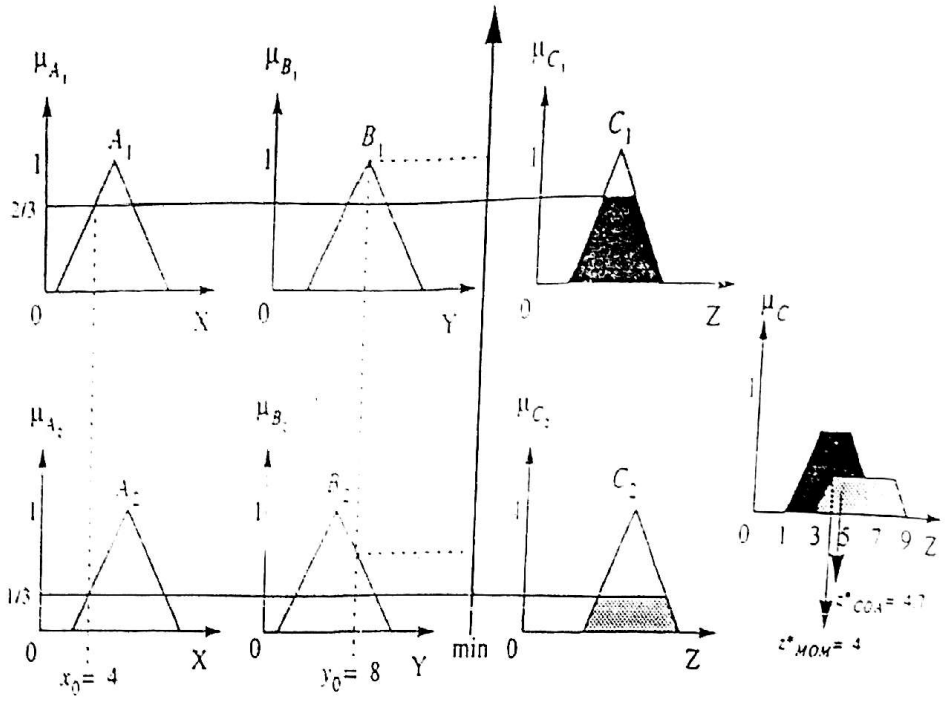

Figure 7.8 Fuzzy reasoning process in Example 7.3.

Using the MOM defuzzification strategy, three quantized values reach their maximum memberships in the combined membership function (i.e., 3, 4, and 5 with membership values of $\frac{2}{3}$). Therefore,

$$z^*_{\text{MOM}} = (3 + 4 + 5)/3 = 4.0.$$

7.2 DESIGN METHODOLOGY OF FUZZY CONTROL SYSTEMS

From the previous discussions, we can see that the principal elements of designing a FLC include (1) defining input and output variables, (2) deciding on the fuzzy partition of the input and output spaces and choosing the membership functions for the input and output linguistic variables, (3) deciding on the types and the derivation of fuzzy control rules, (4) designing the inference mechanism, which includes choosing a fuzzy implication and a compositional operator, and the interpretation of sentence connectives AND and ALSO, and (5) choosing a defuzzification operator.

The first two design principles indicate that, in the design of a FLC, one must identify the main process state variables and control variables and determine a term set that is at the right level of granularity for describing the values of each (linguistic) variable. For example, a three-term set such as {Small, Medium, Large} may not be satisfactory in some domains, and the use of a finer five-term set such as {VERY Small, Small, Medium, Large, and VERY Large} may be required instead. Hence, the number of fuzzy partitions of the input-output spaces should be large enough to provide an adequate approximation and yet be small enough to save memory space. This number has an essential effect on how fine a control can be obtained. Moreover, different types of fuzzy membership functions, for example, monotonic, triangular, trapezoidal, and bell-shaped functions, may be used for the linguistic values of each linguistic variable. There are two methods for making these choices. First, we can use experience and engineering (domain) knowledge to select the possible and proper input-output variables and then use a heuristic cut-and-try procedure to find a proper fuzzy partition and a trial-and-error approach to find suitable membership

functions. Although this method is rather time-consuming and nontrivial, it has been widely employed and has been used in many successful industrial applications. A rule of thumb of this method is concerned with the ϵ completeness of input-output membership functions; that is, the union of the supports of the fuzzy sets in a term set should cover the related universe of discourse in relation to some level set ϵ. In general, we choose the level ϵ at the crossover point as shown in Fig. 7.2, implying that a dominant rule always exists and is associated with a degree of belief greater than 0.5. In the extreme case, two dominant rules are activated with an equal belief of 0.5. Second, we can use learning or self-organization techniques. The idea is to decide on and adjust the fuzzy partitions and membership functions and to select important, useful input-output variables from a group of candidates automatically and systematically, and/or dynamically through learning techniques. In this methodology, the use of neural network–based learning techniques and genetic learning algorithms has proved to be a very promising approach. They will be discussed further in detail in Part III of this book.

For the fourth and fifth design principles of a FLC, there is no systematic methodology for realizing the design of an inference engine and the choice of a defuzzification operator. Most practitioners use empirical studies and results to provide guidelines for these choices.

The third design principle in determining fuzzy control rules depends heavily on the nature of the controlled plant. In general, there are four methods for the derivation of fuzzy control rules [Sugeno, 1985a; Lee, 1990], and these methods are not mutually exclusive. A combination of them may be necessary to construct an effective method for the derivation of fuzzy control rules.

1. **Expert experience and control engineering knowledge:** Fuzzy control rules are designed by referring to a human operator's and/or a control engineer's knowledge. More specifically, we can ask a human expert to express his or her knowledge in terms of fuzzy implications, that is, to express this know-how in fuzzy IF-THEN rules. We can also ask a control engineer to list a number of protocols based on his or her knowledge about the process to be controlled. Finally, a heuristic cut-and-try procedure is used to fine-tune the fuzzy control rules. This method is the least structured of the four methods, and yet it is the most widely used. A typical example is the operating manual for a cement kiln [King and Karonis, 1988; Zimmermann, 1991]. The kiln production process is complex and nonlinear, it contains time lags and interrelationships, and the kiln's response to control inputs depends on the prevailing kiln conditions. These factors are certainly the reason why a fuzzy controller was designed and used. The aim is to automate the routine control strategy of an experienced kiln operator. The strategies are based on detailed studies of the process operator experiences which include a qualitative model of influence of the control variables on the measured variables, for example, "If the air flow is increased, then the temperature in the smoke chamber will increase, while the kiln drive load and the oxygen percentage will decrease." From this kind of verbal statement, we can derive the following rule:

   IF drive load gradient is *Normal*

   AND drive load is *SLIGHTLY High*

   AND smoke chamber temperature is *Low*,

THEN change oxygen percentage is *Positive*

AND change airflow is *Positive*.

More details on this example can be found in [Holmblad and Ostergaard, 1982].

The disadvantages of this mode of derivation of fuzzy control rules are that (i) an operator may not be able to verbalize his or her knowledge, and (ii) it may be difficult for a control engineer to write down control rules because the controlled process is too complex.

2. Modeling an operator's control actions: We can model an operator's skilled actions or control behavior in terms of fuzzy implications using the input-output data connected with his control actions. Then we can use the obtained "input-output model" as a fuzzy controller. The idea behind this mode of derivation is that it is easier to model an operator's actions than to model a process since the input variables of the model are likely found by asking the operator what kind of information he uses in his control actions or by watching these actions.

A typical and interesting example is Sugeno's fuzzy car [Sugeno and Murakami, 1985; Sugeno and Nishida, 1985]. His model car has successfully followed a crank-shaped track and parked itself in a garage. The training process involves a skilled operator guiding the fuzzy model car under different driving conditions. The control policy incorporated is represented by a set of state-evaluation fuzzy control rules:

$$R^i\text{: IF } x \text{ is } A_i \text{ AND } \ldots \text{ AND } y \text{ is } B_i, \text{ THEN } z = a_0^i + a_1^i x + \cdots + a_n^i y, \tag{7.35}$$

where $x,\ldots,$ and y are linguistic variables representing the distance and orientation in relation to the boundaries of the track, z is the next steering angle decided by the ith control rule, and $a_0^i,\ldots,a_n^i$ are the parameters entering in the identification process of a skilled driver's actions. This identification is made by optimizing a least squares performance index via a weighted linear regression method (a weighted recursive least squares algorithm) [Tong, 1978a; Takagi and Sugeno, 1983, 1985; Sugeno and Kang, 1986, 1988]. In addition to modeling an operator's actions, this method is also used to model (identify) controlled processes according to their input-output data, which involves parameter learning as well as structure learning [Sugeno and Tanaka, 1991]. These are called *linguistic control rule* approaches to fuzzy modeling or fuzzy identification. The Takagi and Sugeno fuzzy model has the following form:

$$\begin{aligned} L^i\text{: IF } & x(k) \text{ is } A_1^i \text{ AND } \ldots \text{ AND } x(k-n+1) \text{ is } A_n^i \\ & \text{AND } u(k) \text{ is } B_1^i \text{ AND } \cdots \text{ AND } u(k-m+1) \text{ is } B_m^i, \\ \text{THEN } & x^i(k+1) = a_0^i + a_1^i x(k) + \cdots + a_n^i x(k-n+1) \\ & + b_1^i u(k) + \cdots + b_m^i u(k-m+1), \end{aligned} \tag{7.36}$$

where $x(\cdot)$ is the state variable, $x^i(k+1)$ is the output of rule L^i, and $u(\cdot)$ is the input variable. The output of the fuzzy model can be obtained by fuzzy reasoning of the third type [Eq. (7.29)]. The design of a FLC based on the derived fuzzy model is called a "model-based" FLC design and is representative of the third mode of design methodology described below.

3. **Based on a fuzzy model or behavior analysis of a controlled process:** In this mode, fuzzy control rules are derived or justified based on either the fuzzy model or the behavior analysis of a controlled process. If we have a fuzzy model of the process or if we know some useful properties of the process, we can design or generate a set of fuzzy control rules for attaining optimal performance. By fuzzy modeling, we mean representation of the dynamic characteristics of the process by a set of fuzzy implications with inputs, state variables, and outputs. There are two methods for designing fuzzy control rules in this mode.

   (a) *Heuristic method*: We set a fuzzy control rule to compensate for an undesirable system behavior by considering the control objective. This is done by analyzing the behavior of a controlled process. Such behavior analysis techniques include the phase-plane approach [King and Mamdani, 1975], the linguistic phase-plane approach [Braae and Rutherford, 1979a], the pole placement approach [Braae and Rutherford, 1979b], a fuzzy Proportional-Integral-Derivative (PID) control approach [Peng et al., 1988; Abdelnour et al., 1991], and other approaches [Baaklini and Mamdani, 1975; Mamdani and Assilian, 1975]. These techniques usually have their counterparts in conventional control theory.

   (b) *Optimal control method*: This is basically a deterministic method which can systematically determine the linguistic structure and/or parameters of fuzzy control rules that satisfy the control objectives and constraints (of minimizing a performance index) based on the fuzzy model of a process. Such systematic methods are usually studied by means of *fuzzy relational equations* and *linguistic control rules* for fuzzy modeling which are comprised of two phases, namely, structure identification and parameter estimation. The linguistic control rule approaches were mentioned above. For the fuzzy relational equation approach, structure identification requires determination of the system order and time delays of discrete-time fuzzy models, while parameter estimation reduces to determination of the overall fuzzy relation matrix from the input-output data of the system [Czogala and Pedrycz, 1981, 1982; Pedrycz, 1981, 1984, 1989; Togai and Wang, 1985; Xu and Zailu, 1987; Sugeno and Tanaka, 1991]. Let us use an example to illustrate the fuzzy relational equation approach to the fuzzy controller design.

Example 7.4

[Togai and Wang, 1985] Consider the fuzzy dynamical system in Fig. 7.9, where X and Y are fuzzy subsets on U and V, respectively, R is a ternary relation on $V \times U \times V$, and D is a delay unit. The fuzzy relation R represents the transition relation for the closed-loop system with a first-order delay unit D. Thus, we have

$$Y_{t+1} = (Y_t \cap X_t) \circ R = X_t \circ (Y_t \circ R), \tag{7.37}$$

where the second equality holds for the min operation ($\cap$) and the max-min composition ($\circ$). Then it is possible to derive the input X_t, which drives the state from Y_t to Y_{t+1}, by solving the fuzzy relational equation in Eq. (7.37) (see Sect. 3.4) as shown below:

$$X_t = (Y_t \circ R) \overset{\alpha}{\leftarrow} Y_{t+1}, \tag{7.38}$$

where $\overset{\alpha}{\leftarrow}$ indicates the α operator and is defined in Eq. (3.52).

4. **Based on learning (or self-organizing).** Many FLCs have been built to emulate human decision-making behavior. Currently, many research efforts are focused on

includes aspects of approximately achieving the previous goal, that is, "IF g_{i-1} is approximately achieved AND x_i is A_i, THEN z is C_i." We can see that the interactions between the goals g_i and g_{i-1} are handled by forming rules that include more preconditions on the left-hand side. For example, assume that we have acquired a set of rules for keeping a pole vertical. In deriving the second rule set R_2 for moving to a prespecified location, a precondition such as *the pole is almost balanced* can be added. The linguistic hedge, *concentration* [Eq. (6.11)], can be used here to systematically obtain a more focused membership function for the parameters representing the achievement of previous goals.

We have discussed general methodologies for the design of FLCs with four modes of deriving fuzzy control rules and two hierarchical design approaches. These design approaches and the derivation of fuzzy control rules are used in many industrial and consumer product applications. A few of these applications will be discussed in Sect. 7.4, and further applications of fuzzy sets and fuzzy logic will be considered in Chap. 8.

7.3 STABILITY ANALYSIS OF FUZZY CONTROL SYSTEMS

One of the most important concepts concerning the properties of control systems is stability. This is also true for fuzzy control systems. As fuzzy control has been successfully applied to many practical industrial applications, stability analysis of fuzzy control systems is gaining much attention. Of various existing methodologies for stability analysis of fuzzy systems [Mamdani, 1976; Kickert and Mamdani, 1978; Tong, 1978b, 1980; Braae and Rutherford, 1979a,b; Kania et al., 1980; Pedrycz, 1981; Kiszka et al., 1985; Chen, 1989; Yamashita, 1991; Langari and Tomizuka 1990b, 1991], the one proposed by Tanaka et al. is introduced in this section.

Tanaka and Sugeno [1992] and Tanaka and Sano [1992] used Lyapunov's direct method to perform stability analysis of fuzzy control systems where the fuzzy rules are in the form of Eqs. (7.35) and (7.36). They used Takagi and Sugeno's fuzzy control rule [Eq. (7.35)] and fuzzy model [Eq. (7.36)] and assumed that the membership functions of the fuzzy sets in these rules (e.g., A_i, B_i, A_k^i, B_k^i) are *continuous piecewise polynomial functions* (e.g., triangular type or trapezoidal type). Let us consider the following fuzzy system with zero input:

$$\begin{aligned} L^i\text{: IF } x(k) \text{ is } A_1^i \text{ AND } \cdots \text{ AND } x(k-n+1) \text{ is } A_n^i, \\ \text{THEN } x^i(k+1) = a_1^i x(k) + \cdots + a_n^i x(k-n+1), \end{aligned} \tag{7.40}$$

where $i = 1, 2, \ldots, l$. The consequent of Eq. (7.40) can be written in matrix form as

$$x^i(k+1) = \mathrm{A}_i \mathrm{x}(k), \tag{7.41}$$

where $\mathrm{x}(k) = [x(k), x(k-1), \ldots, x(k-n+1)]^T$ and

$$A_i = \begin{bmatrix} a_1^i & a_2^i & \cdots & a_{n-1}^i & a_n^i \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{7.42}$$

The output of the fuzzy system is inferred by fuzzy reasoning of the third type [Eq. (7.29)] as follows:

$$\mathbf{x}(k+1) = \frac{\sum_{i=1}^{l} \alpha_i A_i x(k)}{\sum_{i=1}^{l} \alpha_i}, \tag{7.43}$$

where α_i is the firing strength, which is defined as [the min operator in Eq. (7.23) is replaced by the product operation]

$$\alpha_i = \mu_{A_1^i}(x_0(k)) \cdot \mu_{A_2^i}(x_0(k+1)) \cdots \mu_{A_n^i}(x_0(k-n+1)), \tag{7.44}$$

where $x_0(\cdot)$ denotes the crisp value of $x(\cdot)$ at a specific instance. With this formulation, we have the following theorems of stability analysis. The first one is the well-known Lyapunov's stability theorem [Kuo, 1980].

Theorem 7.4

Consider a discrete system

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)), \tag{7.45}$$

where $\mathbf{x}(k) \in \mathcal{R}^n$, $\mathbf{f}(\mathbf{x}(k))$ is an $n \times 1$ function vector with the property that $\mathbf{f}(0) = 0$ for all k. Suppose that there exists a scalar function $V(\mathbf{x}(k))$ continuous in $\mathbf{x}(k)$ such that

(a) $V(0) = 0$,
(b) $V(\mathbf{x}(k)) > 0$ for $\mathbf{x}(k) \neq 0$,
(c) $V(\mathbf{x}(k))$ approaches infinity as $\|\mathbf{x}(k)\| \to \infty$,
(d) $\Delta V(\mathbf{x}(k)) < 0$ for $\mathbf{x}(k) \neq 0$.

Then the equilibrium state $\mathbf{x}(k) = 0$ for all k is asymptotically stable in the large, and $V(\mathbf{x}(k))$ is a Lyapunov function.

Theorem 7.5

If P is a positive-definite matrix such that

$$A^T P A - P < 0 \quad \text{and} \quad B^T P B - P < 0, \tag{7.46}$$

where $A, B, P \in \mathcal{R}^{n \times n}$, then

$$A^T P B + B^T P A - 2P < 0. \tag{7.47}$$

Proof: We have

$$\begin{aligned} A^T P B + B^T P A - 2P &= -(A-B)^T P (A-B) + A^T P A + B^T P B - 2P \\ &= -(A-B)^T P (A-B) + A^T P A - P + B^T P B - P. \end{aligned}$$

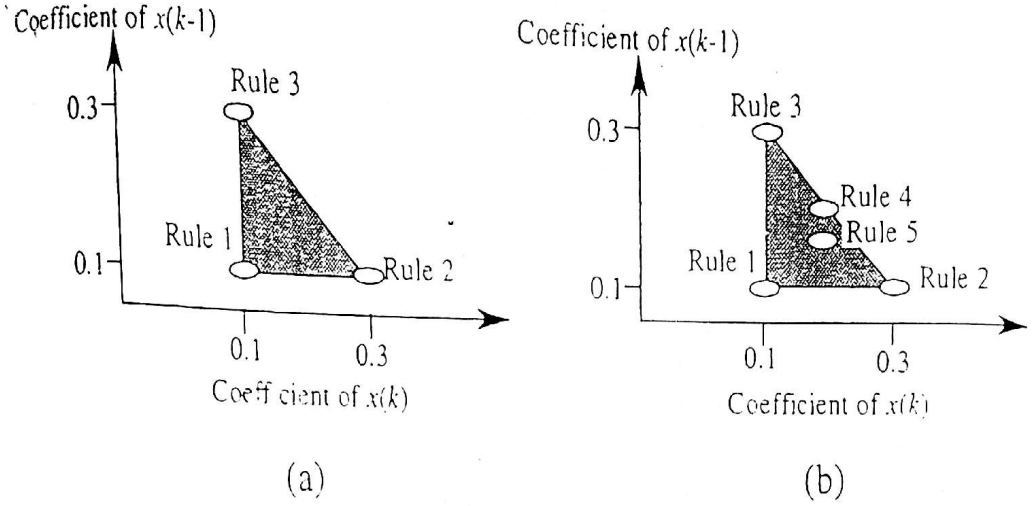

Figure 7.14 Parameter region (PR) representations of the fuzzy systems in Example 7.7. (a) PR of fuzzy system 1. (b) PR of fuzzy system 2.

We notice a difference between the PR of fuzzy system 1 and that of fuzzy system 2. In the former [Fig. 7.14(a)], each plotted point corresponds to each edge of the parameter region. Conversely, in the latter [Fig. 7.14(b)], the parameter region constructed using the plotted points of rules 1–3 includes the plotted points of rules 4 and 5. This observation implies that rules 1–3 of fuzzy system 1 or fuzzy system 2 are edges of the PR; they are said to be *edge rules*. The consequent matrices $\mathbf{A}_1$, $\mathbf{A}_2$, and $\mathbf{A}_3$ in *edge rules* are said to be *edge matrices*. A fuzzy system that consists only of *edge rules* is said to be a *minimum representation*. Obviously, fuzzy system 1 in Example 7.7 is a minimum representation, while fuzzy system 2 is not a minimum representation. The following theorem is important for checking stability in the case of nonminimum representation.

Theorem 7.8

[Tanaka and Sano, 1993]. Assume that $\mathbf{P}$ is a positive-definite matrix. If $\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i - \mathbf{P} < 0$ for $i = 1, 2, \ldots, l$, then $\mathbf{A}^{*T}\mathbf{P}\mathbf{A}^* - \mathbf{P} < \mathbf{0}$, where $\mathbf{A}^*$ is a nonedge matrix such that

$$\mathbf{A}^* = \sum_{i=1}^{l} s_i \mathbf{A}_i, \qquad \text{where } \sum_{i=1}^{l} s_i = 1 \text{ and } s_i \geq 0. \tag{7.58}$$

The above theorem indicates that the stability of a fuzzy system can be checked by applying the Tanaka-Sugeno theorem (Theorem 7.6) to a minimum representation of the fuzzy system. For example, in fuzzy system 2 in Example 7.7, $\mathbf{A}_4 = 0.5\mathbf{A}_2 + 0.5\mathbf{A}_3$ and $\mathbf{A}_5 = 0.5\mathbf{A}_1 + 0.25\mathbf{A}_2 + 0.25\mathbf{A}_3$. Therefore, a minimum representation of fuzzy system 2 is equivalent to fuzzy system 1. Hence, it is found from Theorem 7.8 that fuzzy system 2 is stable if fuzzy system 1 is stable.

APPLICATIONS OF FUZZY CONTROLLERS

Over the past decade, we have witnessed a very significant increase in the number of applications of fuzzy logic–based techniques to various commercial and industrial products and systems. In many applications, especially in controlling nonlinear, time-varying, ill-defined systems and in managing complex systems with multiple independent decision-making

processes, FLC-based systems have proved to be superior in performance when compared to conventional control systems.

Notable applications of FLC include a steam engine [Mamdani and Assilian, 1975; Ray and Majumder, 1985]; a warm water process [Kickert and Van Nauta Lemke, 1976]; heat exchange [Ostergaard, 1977]; activated sludge wastewater treatment [Tong et al., 1980; Itoh et al., 1987; Yu et al., 1990]; traffic junction control [Pappis and Mamdani, 1977]; a cement kiln [Larsen, 1980; Umbers and King, 1980]; aircraft flight control [Larkin, 1985; Chaudhary, 1990; Chiu et al., 1991]; autonomous orbital operations [Lea and Jani, 1992]; a turning process [Sakai, 1985]; robot control [Uragami et al., 1976; Scharf and Mandic, 1985; Tanscheit and Scharf, 1988; Ciliz et al., 1987; Isik, 1987; Palm 1989]; model car parking and turning [Sugeno and Murakami, 1984, 1985; Sugeno and Nishida, 1985; Sugeno et al., 1989]; automobile speed control [Murakami, 1983; Murakami and Maeda, 1985]; a water purification process [Yagishita et al., 1985]; elevator control [Fujitec, 1988]; automobile transmission and braking control [Kasai and Morimoto, 1988]; power systems and nuclear reactor control [Bernard, 1988; Kinoshita et al., 1988]; arc welding [Murakami et al., 1989; Langari and Tomizuka, 1990a]; refuse incineration [Ono et al., 1989]; process control [Efstathiou, 1987]; adaptive control [Graham and Newell, 1989]; automatic tuning [Ollero and Garcia-Cerezo, 1989]; control of a liquid level rig [Graham and Newell, 1988]; gasoline refinery catalytic reformer control [Bare et al., 1990]; a ping-pong game [Hirota et al., 1989]; biological processes [Czogala and Rawlik, 1989]; knowledge structure [Van Der Rhee et al., 1990]; a model helicopter [Sugeno, 1990]; a walking machine [DeYoung et al., 1992]; a rigid disk drive [Yoshida and Wakabayashi, 1992]; highway incident detection [Hsiao et al., 1993]; gas cooling plant control [Tobi et al., 1989]; control theory [Tang and Mulholland, 1987; Berenji et al., 1989; Li and Lan, 1989]; fuzzy hardware devices [Togai and Watanabe, 1986; Togai and Chiu, 1987; Watanabe and Dettloff, 1988; Yamakawa and Miki, 1986; Yamakawa and Sasaki, 1987; Yamakawa, 1988a,b, 1989; Yamakawa and Kabuo, 1988; Hirota and Ozawa, 1988, 1989]; and fuzzy computers [Yamakawa, 1987].

Among these applications, the cement kiln control system was the first successful industrial application of a FLC. In contrast to previous analog fuzzy logic controllers which were designed based on a continuous state-space model, a discrete-event fuzzy controller was intended for airport control [Clymer et al., 1992]. Fuzzy control has also been successfully applied to automatic train operation systems and automatic container crane operation systems [Yasunobu and Miyamoto, 1985; Yasunobu and Hasegawa, 1986, 1987; Yasunobu et al., 1987]. Fuzzy logic control systems have also found application in household appliances such as air conditioners (Mitsubishi); washing machines (Matsushita, Hitachi); video recorders (Sanyo, Matsushita); television autocontrast and brightness control cameras (Canon), autofocusing and jitter control [Shingu and Nishimori, 1989; Egusa et al., 1992]; vacuum cleaners (Matsushita); microwave ovens (Toshiba); palmtop computers (Sony); and many others. In the remainder of this section, the application of fuzzy logic in camera tracking control is discussed in more detail as an illustration of fuzzy control.

An interesting application of fuzzy control is the camera tracking control system at the Software Technology Laboratory, NASA/Johnson Space Center, used to investigate fuzzy logic approaches in autonomous orbital operations [Lea and Jani, 1992]. The camera tracking control system utilizes the tracked object's pixel position on the image as input and

controls the gimbal drives to keep the object in the field of view (FOV) of the camera as shown in Fig. 7.15(a). Thus, tracking an object means aligning the pointing axis of a camera along the object's line of sight (LOS). The LOS vector is estimated from the sensory measurements.

In this camera tracking control system, the monitoring camera is mounted on the pan and tilt gimbal drives, which can rotate the camera or, equivalently, the pointing axis of the camera within a certain range. The *camera frame* (viewing plane) consists of three axes: vertical, horizontal, and pointing vectors. This plane is a Cartesian coordinate plane of 170×170 pixels with the origin at the upper left corner as shown in Fig. 7.15(a). When an image is received, it is processed to determine the location of the object in the camera frame. Using an appropriate image processing technique, the centroid of the image is computed and used as the current location of the object in the viewing plane.

As shown in Fig. 7.15(b), the inputs to a fuzzy logic–based tracking controller are the LOS vector and the range of the object (the distance of the object from the camera), and the outputs are the command pan and tilt rates. The LOS vector is expressed in terms of pixel position (x, y) in the camera's FOV. The range of the object is received from the laser range finder as a measurement. With these three inputs, the task of the FLC is to determine the proper pan and tilt rates for the gimbal drives so that the pointing axis of the camera is along the LOS vector of the object and the image location is at the center of the viewing plane [i.e., at (85, 84)]. In the camera's FOV, tilt upward is negative and pan right is positive.

Membership functions of the input variables (i.e., range, horizontal, and vertical positions) are shown in Fig. 7.16(a). Membership functions of the scale-factor and the output variables (i.e., pan rate and tilt rate) are shown in Fig. 7.16(b). The scale-factor parameter is used as an intermediate step to indicate the degree that a control action should propose to reflect the distance of the tracked object from the camera. The basic concept is that the movement of the object in the FOV caused by the movement of the camera is greater when the object is closer to the camera than when it is farther from the camera.

Three sets of fuzzy control rules are used [see Table (7.3)]. The first set of rules is for finding the scale factor, which will be used in the next two sets of rules. For example, one rule in the second rule set may read: "If the horizontal position of the object is to the far left

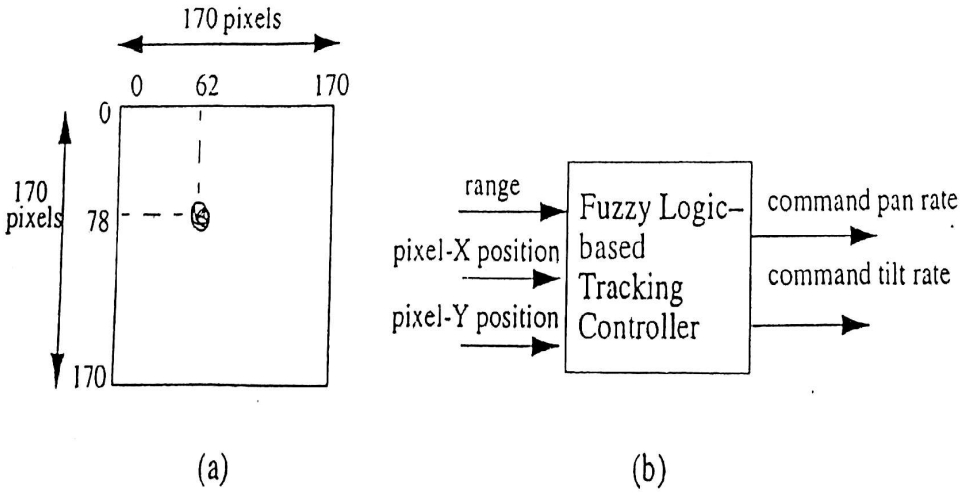

Figure 7.15 Camera tracking system. (a) Camera field of view. (b) Input-output of fuzzy logic–based tracking controller.

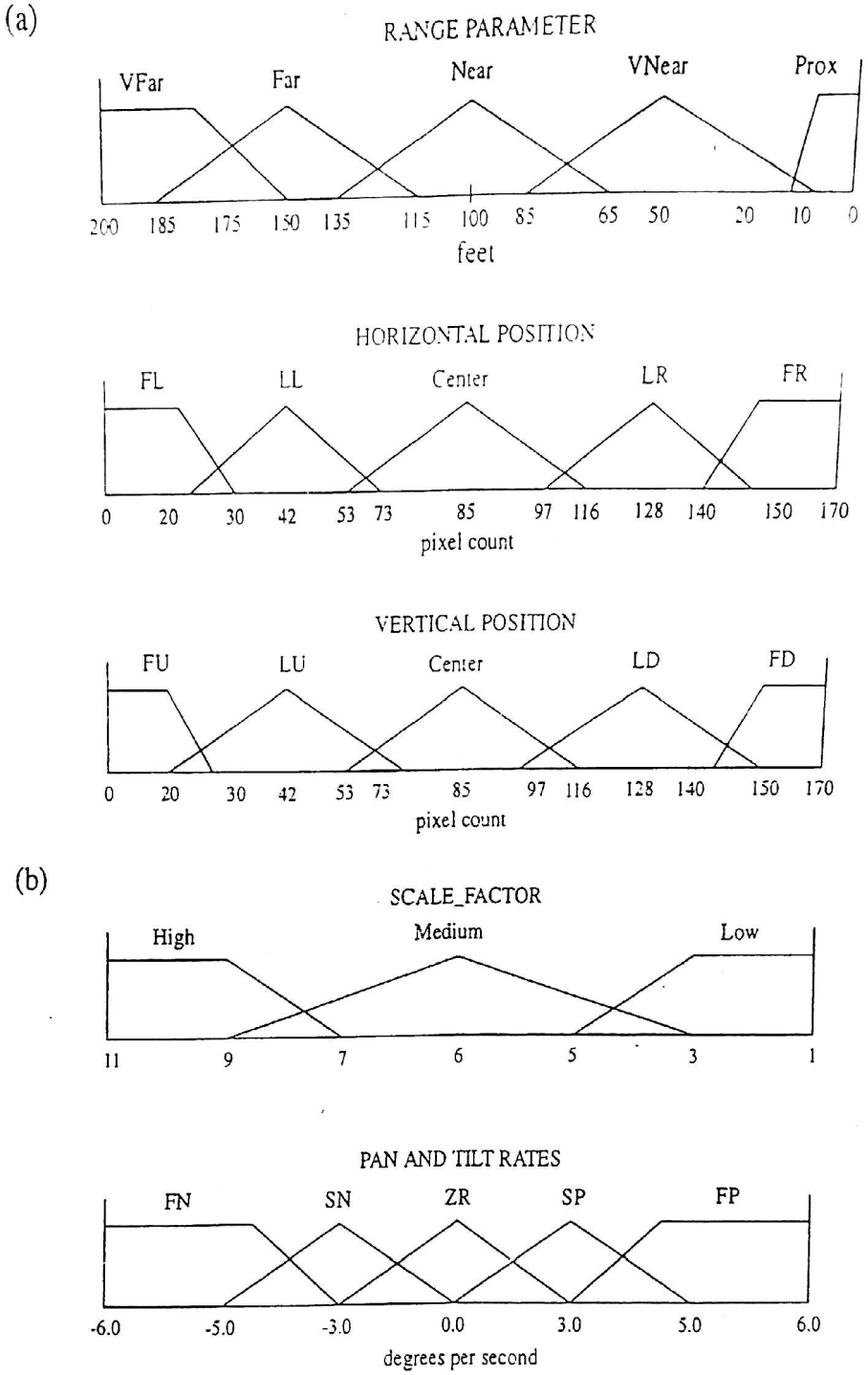

Figure 7.16 (a) Membership functions for input parameters. VFar, Very far; VNear, Very near; Prox, Proximity zone; FL, far left; LL, little left; LR, little right; FR, far right; FU far up; LU, little up; LD, little down; FD, far down. (b) Membership functions for Scale_Factor and Output parameters for camera tracking system. FN, Fast negative; SN, slow negative; ZR, zero; FP, fast positive; SP, slow positive. (Reprinted by permission of the publisher from "Fuzzy Logic in Autonomous Orbital Operations," by R. N. Lea and Y. Jani, *International Journal of Approximate Reasoning*, Vol. 6, No. 2, pages 151–184. Copyright 1992 by Elsevier Science Inc.)

of the center of the viewing plane and the scale factor is low (i.e., the distance of the object is far), then set the pan rate fast negative (left)." This example also illustrates application of the *chain rule* technique of expert systems in FLC design; that is, the firing of a rule causes the firing of another rule. Normally, a FLC has only single-layer-rule firing.

Applications of Fuzzy Theory

This chapter explores the application of fuzzy set theory to various application domains such as pattern recognition, optimization, mathematical programming, database systems, and human-machine interactions. In some applications, the incorporation of fuzzy set concepts into existing methodologies or techniques broadens their flexibility and robustness. Examples are worked out to illustrate the improved performance. Because of space limitations, this chapter samples only some important applications.

8.1 FUZZY PATTERN RECOGNITION

Much of the information that we have to deal with in real life is in the form of complex patterns. Pattern recognition involves the search for structure in these complex patterns. The methodologies used for recognition schemes include linear classification, statistical (probabilistic) approaches, fuzzy set theory (possibility approaches), perceptrons (neural networks), knowledge-based classification based on artificial intelligence techniques, and many others. Among these, fuzzy set theory has long been considered a suitable framework for pattern recognition, especially classification procedures, because of the inherent fuzziness involved in the definition of a class or a cluster. Indeed, fuzzy set theory has introduced several new methods of pattern recognition which have led to successful realizations in various areas including speech recognition, intelligent robots, image processing, character recognition, scene analysis, recognition of geometric objects, signal classification, and medical applications. Major references include [Bezdek, 1981; Kandel, 1982; Pal and Dutta Majumder, 1986; Pedrycz, 1990a; Bezdek and Pal, 1992].

As shown in Fig. 8.1, pattern recognition usually consists of three steps: (1) data acquisition, (2) feature selection, and (3) classification. First, the data for classification are gathered from the environment via a set of sensors. They can be numerical, linguistic, or both. Afterward, feature selection is usually performed to search for internal structure in the data. It is desirable that the dimensions of the feature space be much smaller than those of the data space so that classification techniques can be efficiently applied. Finally, classification is performed via a classifier and is actually a transformation between classes and features.

The concept of fuzzy set theory can be introduced into the pattern recognition process in Fig. 8.1 to cope with uncertainty in several different ways. Two of them are in evidence: (i) fuzziness involving the feature space, and (ii) fuzziness involving the classification space. It is understood that most of the information gathered in the recognition processes of a human being is of a nonnumerical type. Even if numerical data are available, the process is worked out by the human mind at a level of nonnumerical labels. This indicates that the classification is performed not on the basis of a mass of numbers but by elicitation relationships between the classes and linguistic labels attached to the object for recognition. These linguistic labels can be represented by fuzzy sets specified in appropriate spaces. Thus, we may have, for example, a classification rule like, "If an object is *heavy* and *small* and it moves *fast*, then it belongs to the class ω_i." A second important way of introducing fuzziness is about class assignment ("labeling") in the classification space. Unlike "hard" labeling in which an object is classified as belonging to only one class crisply, "fuzzy" labeling allows an object to be identified as belonging to different classes to different degrees. That is, the boundaries of the classes are vague. For example, we can say, "If an object is black and cubic, then it *possibly* belongs to the class ω_i." Obviously, the above two approaches can be merged into a classification rule like, "If an object is heavy and small and it moves fast, then it very possibly belongs to the class ω_i."

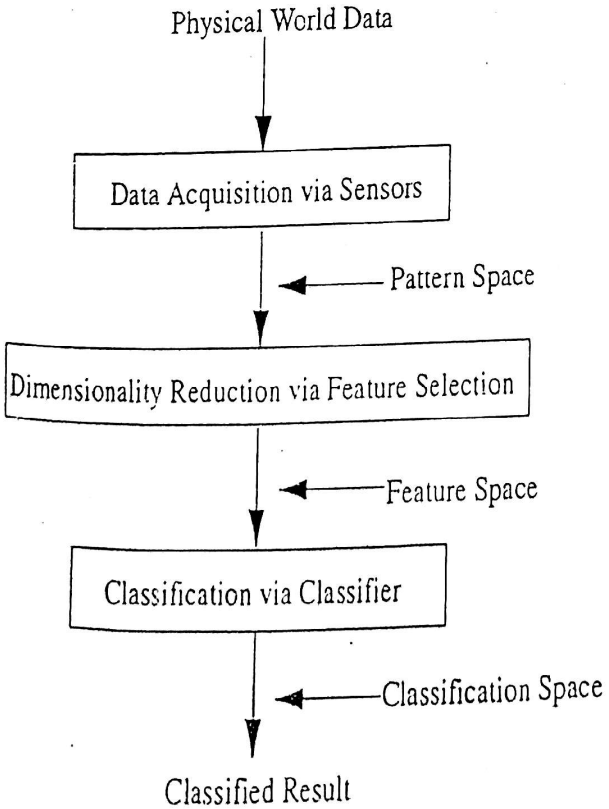

Figure 8.1 General scheme of pattern recognition.

2 FUZZY MATHEMATICAL PROGRAMMING

In mathematical programming, a real problem is described in terms of a mathematical model and then an optimal solution is found from the model. Specifically, mathematical programming is an algorithmic-based approach to solving the following type of optimization problem:

$$\begin{aligned} &\text{Maximize } f(x) \\ &\text{subject to } g_i(x) = 0 \qquad \text{for } i = 1, 2, \ldots, m, \end{aligned} \tag{8.31}$$

where $f(x)$ and $g_i(x)$ are, respectively, the *objective function* (goal) and the *constraints* of the problem to be solved. One of the most simple and most commonly used types of mathematical programming of Eq. (8.31) is *linear programming*. It focuses on the following problem

$$\begin{aligned} &\text{Maximize } f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}, \qquad \mathbf{c}, \mathbf{x} \in \mathcal{R}^n \\ &\text{such that } \mathbf{A}\mathbf{x} \le \mathbf{b}, \qquad \mathbf{b} \in \mathcal{R}^m, \mathbf{A} \in \mathcal{R}^{m \times n}, \mathbf{x} \ge 0. \end{aligned} \tag{8.32}$$

Traditionally, it is assumed that each element and operation in Eqs. (8.31) and (8.32) is crisp, that all the constraints are of equal importance, and that the violation of any single constraint renders the solution infeasible. These crisp assumptions and constraints are usually nonexistent in real-world problems. For example, the stated goal may be to maximize our gain to about \$1 million with the constraint to keep investments to about \$10,000 or less. This problem would be difficult to formulate using crisp mathematical programming. Fuzzy linear programming, which expresses a kind of ambiguity in terms of fuzzy sets, better describes our understanding about the problems that we want to optimize than crisp mathematical programming. Several types of fuzzy linear programming problems can be defined by partly relaxing the crisp assumptions in classical linear programming problems [Zimmermann, 1992]. For example, one might just want to improve one's present income considerably instead of maximizing it, a decision maker might accept small violations of different constraints, or one may only have fuzzy data (fuzzy numbers) instead of precise numerical data in this optimization problem. In this section, we shall discuss a simple type of fuzzy linear programming problem.

The problem to be considered is a type of symmetric fuzzy linear programming problem in the form of Eq. (8.32), called a linear programming problem with fuzzy inequality [Zimmermann, 1991, 1992]. The objectives and constraints are given by the following fuzzy inequalities:

$$\begin{aligned} \mathbf{c}^T\mathbf{x} &\succeq z, \\ \mathbf{A}\mathbf{x} &\preceq \mathbf{b}, \\ \mathbf{x} &\ge 0, \end{aligned} \tag{8.33}$$

where $\succeq$ means "essentially greater than or equal" and $\preceq$ means "essentially smaller than or equal." In Eq. (8.33), it is assumed that the decision maker can give an aspiration level z for the objective function, which he or she wants to achieve as far as possible, and that the constraints can be slightly violated.

Since Eq. (8.33) is fully symmetric with respect to the objective function and constraints, we can consolidate and write

$$\begin{aligned} \mathbf{B}\mathbf{x} &\preceq \mathbf{d}, \\ \mathbf{x} &\ge 0, \end{aligned} \tag{8.34}$$

where

$$\mathbf{B} = \begin{bmatrix} -\mathbf{c}^T & 0 \\ 0 & \mathbf{A} \end{bmatrix} \quad \text{and} \quad \mathbf{d} = \begin{bmatrix} -z \\ \mathbf{b} \end{bmatrix}. \tag{8.35}$$

ith inequality [i.e., the ith row of Eq. (8.34)] is defined by the following membership functions

$$\mu_i([\mathbf{Bx}]_i) = \begin{cases} 1 & \text{if } [\mathbf{Bx}]_i \le d_i \\ \in [0, 1] & \text{if } d_i < [\mathbf{Bx}]_i \le d_i + p_i, \quad i = 1, 2, \ldots, m+1 \\ 0 & \text{if } [\mathbf{Bx}]_i > d_i + p_i, \end{cases} \tag{8.36}$$

where $[\mathbf{Bx}]_i$ is the ith element of the vector $\mathbf{Bx}$, $\mu_i(\cdot)$ is the membership function of the ith inequality, d_i is the ith element of the vector $\mathbf{d}$, and p_i is the maximum possible value of the right-hand side of the ith inequality. Using the simplest type of membership function, we assume it to be linearly increasing over the *tolerance interval* p_i (see Fig. 8.6):

$$\mu_i([\mathbf{Bx}]_i) = \begin{cases} 1 & \text{if } [\mathbf{Bx}]_i \le d_i \\ 1 - [\mathbf{Bx}]_{i-d_i}/p_i & \text{if } d_i < [\mathbf{Bx}]_i \le d_i + p_i, \quad i = 1, 2, \ldots, m+1 \\ 0 & \text{if } [\mathbf{Bx}]_i > d_i + p_i. \end{cases} \tag{8.37}$$

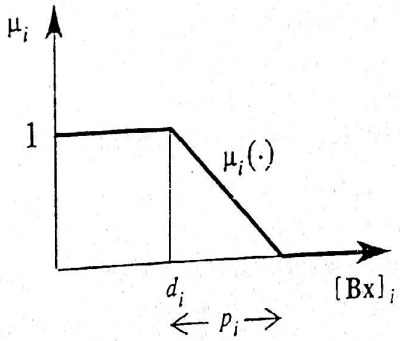

Figure 8.6 Example of the fuzzy set "Essentially smaller than or equal."

Then the maximizing decision is the $\mathbf{x}$ that satisfies

$$\max_{\mathbf{x} \ge 0} \min_i \{\mu_i([\mathbf{Bx}]_i)\} = \max_{\mathbf{x} \ge 0} \min_i \left(1 - \frac{[\mathbf{Bx}]_i - d_i}{p_i}\right). \tag{8.38}$$

By introducing a new variable λ and performing normalization, we can transform Eq. (8.38) into the following standard linear programming problem:

$$\begin{aligned} &\text{Maximize } \lambda \\ &\text{such that } \lambda p_i + [\mathbf{Bx}]_i \le d_i + p_i, \qquad i = 1, 2, \ldots, m+1, \\ &\qquad \mathbf{x} \ge 0, \end{aligned} \tag{8.39}$$

which can be solved using conventional approaches.

Example 8.5

[Zimmermann, 1976]. A company wants to decide on the size and structure of its truck fleet. Four trucks, x_1, x_2, x_3, x_4, of different sizes are considered. The objective is to minimize cost, and the constraints are to supply all customers who have a strong seasonally fluctuating demand. At first, the standard linear programming problem is formulated as follows:

$$\text{Minimize } f(\mathbf{x}) = 41{,}400x_1 + 44{,}300x_2 + 48{,}100x_3 + 49{,}100x_4$$

$$\text{such that } 0.84x_1 + 1.44x_2 + 2.16x_3 + 2.4x_4 \geq 170$$

$$16x_1 + 16x_2 + 16x_3 + 16x_4 \geq 1{,}300$$

$$x_1 \geq 6,\ x_2, x_3, x_4 \geq 0.$$

Then, the corresponding fuzzy linear programming problem is formulated based on the aspiration level $z = 4.2$ million and the following parameters:

Low bounds of the tolerance intervals: $d_1 = 3{,}700{,}000$, $d_2 = 170$, $d_3 = 1{,}300$, and $d_4 = 6$.
Spreads of tolerance intervals: $p_1 = 500{,}000$, $p_2 = 10$, $p_3 = 100$, and $p_4 = 6$.

After dividing all the rows in Eq. (8.39) by their respective p_i values and rearranging the position λ, the fuzzy linear programming problem becomes

$$\text{Maximize } \lambda$$

$$\text{such that } 0.083x_1 + 0.089x_2 + 0.096x_3 + 0.098x_4 + \lambda \leq 8.4$$

$$0.084x_1 + 0.144x_2 + 0.216x_3 + 0.240x_4 - \lambda \geq 17$$

$$0.160x_1 + 0.160x_2 + 0.160x_3 + 0.160x_4 - \lambda \geq 13$$

$$0.167x_1 - \lambda \geq 1$$

$$\lambda, x_1, x_2, x_3, x_4 \geq 0.$$

The solutions for the above nonfuzzy and fuzzy linear programming problems are

Nonfuzzy	Fuzzy
$x_1 = 6$	$x_1 = 17.414$
$x_2 = 16.29$	$x_2 = 0$
$x_3 = 0$	$x_3 = 0$
$x_4 = 58.96$	$x_4 = 66.54$
$z = 3{,}864{,}975$	$z = 3{,}988{,}250$

The values for the constraints are

	Nonfuzzy	Fuzzy
1.	170	174.33
2.	1,300	1,343.328
3.	6	17.414

From the solution, we find that "leeway" has been provided to all constraints at an additional cost of 3.2% using fuzzy linear programming.

8.3 FUZZY DATABASES

In real-world applications, because of the need for data integrity and independence, data from measurement devices or sensors must be efficiently stored and then processed. Because of these requirements, research on database systems has been very active since the

Database systems have traditionally modeled a precise universe where all values are known. However, in many real-world situations, especially in fields directly involving people, such as human-machine systems, decision making, and natural language processing, there is a great deal of ambiguous data whose values are imprecise with fuzzy connotations and are sometimes even missing. To use this ambiguous data constructively, people have tried to incorporate fuzzy set theory into standard databases and have built *fuzzy databases.*

There are important benefits in extending data models to incorporate fuzzy and imprecise information. First, it provides a more accurate representation of the database universe. Second, it allows for data retrieval based on *similarity* of values and thus provides the user with considerably more flexibility in data manipulation. Third, it provides much help in the coupling of artificial intelligence and databases, which has attracted growing research interest in improving the functionality and applicability of database systems.

The first generation of databases consisted of *network* and *hierarchical* data models. Because of the lack of physical independence of these models, *relational* databases, developed in the early 1970s, dominated research efforts in the 1980s and became the second generation of databases. The growing complexity of data modeling requirements, especially those involving complex objects and large amounts of data in scientific databases, led to the development of *extended relational models, semantic database models,* and *object-oriented databases.* Among these, object-oriented database systems mark the genesis of the third generation.

In general, there are two approaches to incorporating fuzzy information into databases. The first is to maintain the standard data model and allows fuzzy queries, and the second is to retain the standard database language (e.g., SQL in a relational database) and extends the data model. First-generation databases, network data models, have not received much attention in fuzzy database research because of the obstacle caused by the functionality condition of network databases indicating that the same record cannot appear in more than one set. In this section, we shall introduce some examples of fuzzy relational databases and fuzzy object-oriented databases [Petry et al., 1992].

8.3.1 Fuzzy Relational Databases

In a relational model, the database is a group of *relations.* The relations are essentially the same as the relations of set theory and are expressed in the form of two-dimensional tables. The columns of a table (relation) are called *attributes;* thus each row (called a *tuple*) is a sequence of attribute values. For each attribute, there is a prescribed set of values, called the *domain,* from which values may be selected. Each element of the domain set has the same structure, for example, integers, real numbers, or character strings.

Almost all fuzzy databases are extensions of relational models. Approaches to the representation of inexact information in relational data models include simply adding a membership attribute value to each relation by substituting similarity for equality in the application of query terms and relational calculus and by allowing data values to be possibility distributions. More than one of these approaches can be applied at the same time. Let us consider an example of a similarity-relational model [Petry et al., 1992].