6

FUZZY CONTROL

6.1 INTRODUCTION

Fuzzy control primarily refers to the control of processes through fuzzy linguistic descriptions. Since 1974, when E. H. Mamdani and S. Assilian (Mamdani, 1974) demonstrated that fuzzy *if/then* rules could regulate a model steam engine, a great number of fuzzy control applications have been successfully deployed. The list is very long and growing and includes cement kilns, subway trains, unmanned helicopters, autonomous mobile robots, process heat exchangers, and blast furnaces (Mamdani, 1977; Ostergaard, 1982; Yasunobu and Miyamoto, 1985; King and Karonis, 1988).[1] In the 1970s and early 1980s most applications were minicomputer-based, often found in the process industry in areas where automatic control was rather difficult to realize and hence left in the hands of human operators. More recently, with the advent of fuzzy microprocessors, a growing number of fuzzy control applications have emerged in consumer electronics and home appliances such as hand-held cameras, vacuum cleaners, air conditioners, and washing machines (Hirota, 1993; Yamakawa, 1989; Schwartz, 1992; Terano et al., 1992).

In this chapter we begin by reviewing conventional process control in order to establish the relevant context and proceed to fuzzy control, a subject we view primarily as an application of fuzzy linguistic descriptions (Chapter 5). Of course, the appropriate choice of controller in engineering applications

[1]There are a number of excellent books available on fuzzy control. The interested reader may want to consult, for example, Driankov et al., 1993; Pedrycz, 1993; Harris et al., 1993; Yager and Filev, 1994; and Wang, 1994.

is made not as much by a commitment to a particular methodology or technology as by careful examination of the needs and features of a given application. In fact, some of the most successful applications of fuzzy control have been in conjunction with conventional controllers such as the *proportional integral derivative* (PID) controller (Lee, 1990a, b). In fuzzy control we are concerned with two broad questions: How can we implement a control strategy as a fuzzy linguistic description? and What are the crucial factors involved in fuzzy algorithmic synthesis and analysis? Although fuzzy linguistic descriptions are a subject of wider interest than the replacement or enhancement of PID controllers, their application to control serves to illustrate some of the basic ideas we encountered in earlier chapters.

Consider the simple process system shown in Figure 6.1. Here, a tank is filled with liquid flowing from a pipe at the top (inlet flow). Liquid leaves the tank through a pipe at the bottom (outlet flow). The upper pipe is fitted with control valve A, used to adjust inlet flow, and the bottom pipe with valve B is assumed to remain at a preset position. A *controller* maintains the liquid in the tank at the desired level. By *process* here we mean the tank, the liquid, the pipes, and the valves. The term *process control system* refers to the

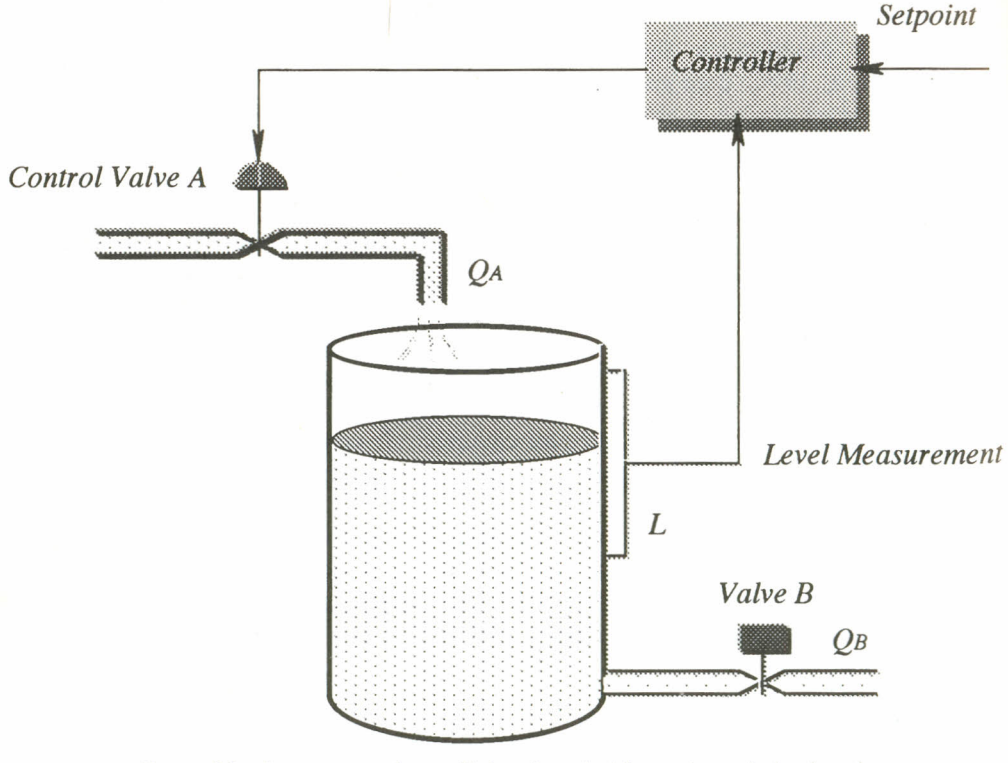


Figure 6.1 A process system with level control through control valve A.

process plus the controller and any required components for measurement and actuation.

The purpose of any process control system is to regulate some *dynamic variable* or *variables* of the process. In the liquid level process control system shown in Figure 6.1, the dynamic variable is the liquid level L, a process parameter that depends on other parameters and thus suffers changes from many different inputs. We select one of these other parameters to be our *controlling parameter*—in this case control valve A, the adjustment of which leads to control of flow rate, Q_A. Liquid level depends on flow rates via control valve A and valve B, ambient temperature T_A (not shown), liquid temperature T_l (also not shown), and the physical condition of valves A and B. This dependence may be described by a process relation of the form

$$L = f(Q_A, Q_B, T_a, T_l) \tag{6.1-1}$$

where Q_A is the flow rate through control valve A, Q_B is the flow rate through valve B, T_a is the ambient temperature, and T_l is the liquid temperature. In many cases the relationship of equation (6.1-1) is not analytically known and actually may not be a function (a *many-to-one mapping*) but instead a more general relation (a *many-to-many mapping*) as we discussed in Chapter 5.

The input to the controller is usually not L itself but instead the error e between a measured indication of L, denoted as y, and a *setpoint* or *reference* value r representing the desired value of the dynamic variable. The controller's *output* or *manipulated variable* is denoted by u and is a signal representing action to be taken when the measured value of the dynamic variable y deviates from reference r. Thus, the output of the controller u serves as input to the process. The error $e = r - y$ is actually *smoothed* and *scaled* before input to the controller. Smoothing is performed in sampled systems in order to avoid the instantaneous changes during sampling that misled the general direction of change for the variable. Such a smoothing function may be defined recursively as $(e_k) \equiv 0.9e_{k-1} + 0.1e_k$, where e_k is the error value at time $t = k$. Scaling is required in order to transform instrument values to a predetermined interval or transform them to a range of numbers that correspond to natural magnitudes.

The most common controller in the process industry is the *PID controller*, where the control relation associated with equation (6.1-1) takes the form

$$u(t) = K_P e(t) + K_P K_I \int_{t=0}^{t} e(t)\, dt + K_P K_D \frac{de(t)}{dt} + u(0) \tag{6.1-2}$$

where K_P is the *controller gain* representing a proportionality constant between error and controller output (dimensionless), K_I is the *reset constant* relating the rate to the error in units of [%/(% − sec)], K_D is the *rate constant* (or *derivative gain constant*) in units of [(% − sec)/%], and $u(0)$ is the controller output at $t = 0$ (when a deviation from setpoint starts).

The first term in equation (6.1-2) is called the *proportional term*, and if it was the only term in the equation it would represent a mode of control where the output of the controller $u(t)$ is changed in proportion to the error $e(t)$, which is the percent deviation from the setpoint. The second term is called the *integral term* and represents a mode of control where the present controller output depends on the history of errors from when observations started at $t = 0$. The amount of corrective action due to integral mode is directly proportional to the length of time that the error has existed. The reset constant K_I expresses the scaling between error and controller output. A large value of K_I means that a small error produces a large rate of change of u and vice versa. If this term alone was used in equation (6.1-2), in addition to the constant $u(0)$, then we would have a mode of control called *integral mode*. The third term in equation (6.1-2) represents the *derivative mode of control*. This mode provides that the controller output depends on the rate of change of error. Derivative mode tends to minimize oscillation of the system and prevent overshooting. Since derivative control is based solely on the rate of change of error, the controlled variable can stabilize at a value different from r, a condition termed "offset." In pure derivative mode the output depends upon the rate at which the error is changed and not on the value of the error. Integral control is used to address situations when permanent offset or slow returns to desired values cannot be tolerated. The combination of these three modes is called *proportional integral derivative* or (PID) control. PID is a powerful composite mode of control that has been used for virtually any linear process condition.

The process of adjusting the coefficients of each mode of control in equation (6.1-2) is called *tuning*. There are several methods for determining the optimum value of these gains such as *frequency response methods* and the *Ziegler–Nichols method* (Johnson, 1977). Fuzzy and neural approaches with adaptive characteristics have also been used for PID tuning and more generally for emulating and enhancing PID controllers (Matia et al., 1992; Maeda and Murakami, 1992; Shoureshi and Rahmani, 1992; He et al., 1993).

Example 6.1 PID Level Control. Consider the process control system shown in Figure 6.1. Suppose that we control the liquid level in the tank by adjusting control valve A (inlet flow) through a PID controller. The output of the controller $u(t)$ is based on the error $e(t)$—that is, the difference between a reference value r and the measured value of level y. The output of the controller is given by equation (6.1-2) as

$$u(t) = K_P e(t) + K_P K_I \int_{t=0}^{t} e(t)\,dt + K_P K_D \frac{de(t)}{dt} + u(0) \quad \text{(E6.1-1)}$$

with the following values for the various gains and initial controller output: $K_P = -1.3$, $K_I = 0.5[\%/(\% - \text{min})]$, $K_D = 1.9[(\% - \text{min})/\%]$, and $u(0) = 50\%$.

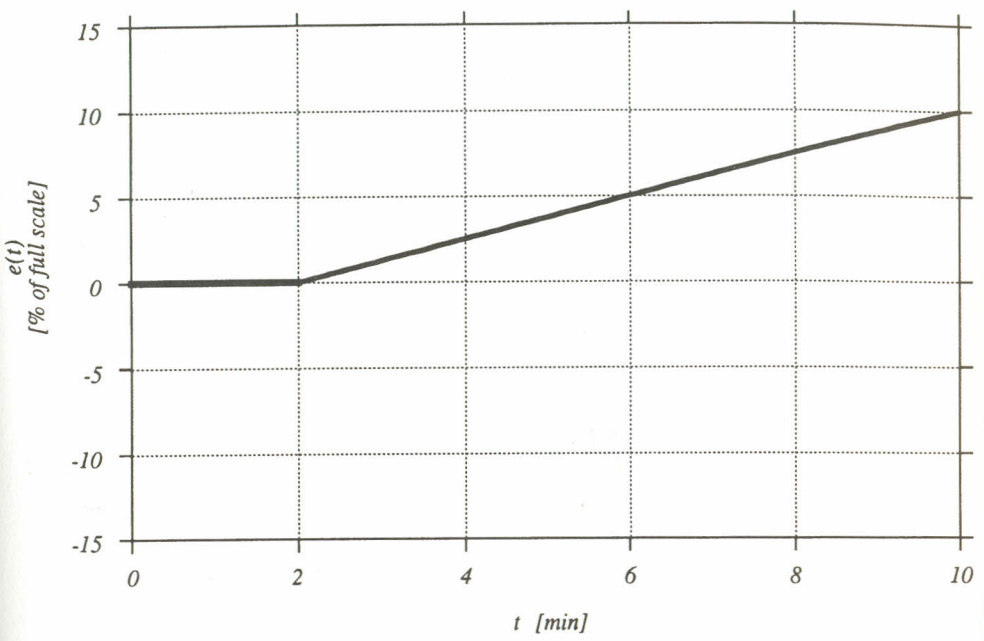


Figure 6.2 Error introduced to the PID level controller of Example 6.1.

Suppose that the error shown in Figure 6.2 is introduced to the system at $t = 2$ min. Such an error may be due to any change in process parameters—for example, an unforeseen change in the position of valve B—since valve B is not under control. The equation of error as function of time is

$$e(t) = 1.25t - 2.5 \quad \text{(E6.1-2)}$$

Using equation (E6.1-2) in equation (E6.1-1) the output of the controller after $t = 2$ is given by

$$u(t) = -1.3[1.25t - 2.5] - 1.3(0.5\ \text{min}^{-1}) \int_{t=2}^{t} [1.25t - 2.5]\,dt$$

$$- 1.3(1.9\ \text{min}) \frac{d}{dt}[1.25t - 2.5] + 50\% \quad \text{(E6.1-3)}$$

The first term in equation (6.1-3) represents the *proportional mode* of the controller, the second term the *integral mode*, and the third term the *derivative mode*. Let us call them $u_P(t)$, $u_I(t)$, and $u_I(t)$, respectively. Figure 6.3 shows the response due to each mode and the total response of the controller $u(t)$, which is the sum of the three terms plus the initial output of the controller, in this case 50%. Looking at Figure 6.3 we note that at the

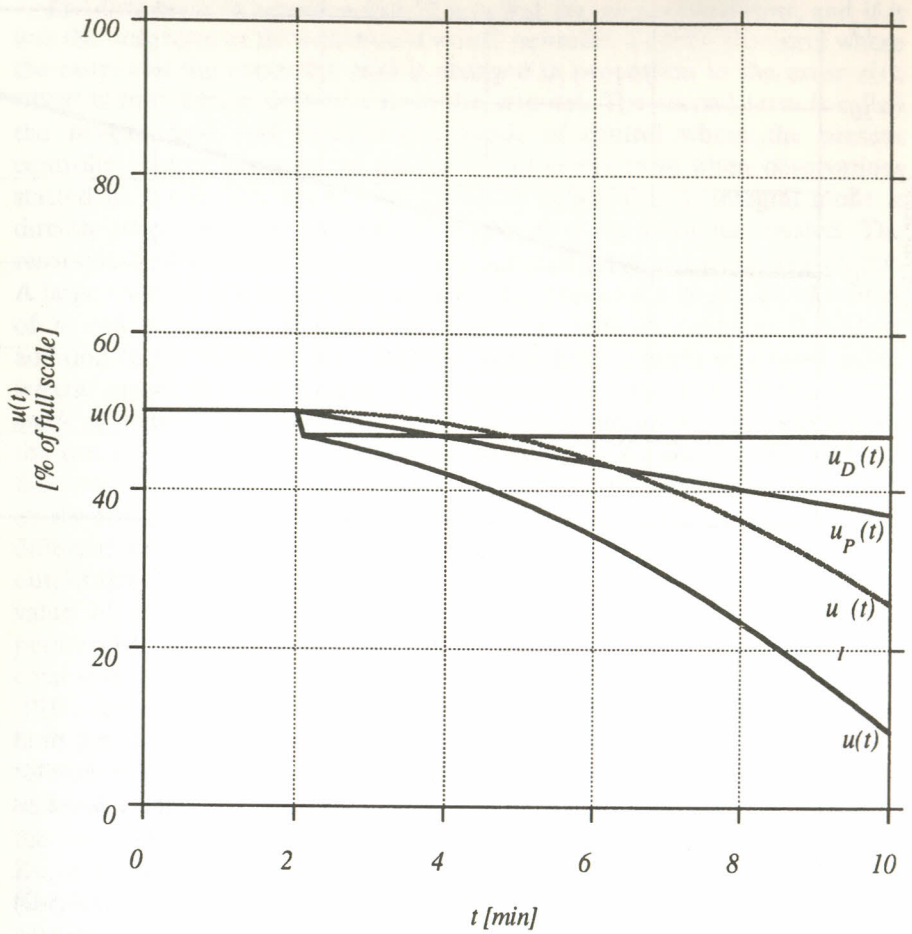


Figure 6.3 Proportional, integral, derivative, and total response of the PID level controller in Example 6.1.

end of the 10-min interval the controller sends a signal to control valve A, which is about 10% of its full scale. This does not necessarily mean that the valve itself allows at that time 10% of full flow to the tank. Different valves have different characteristics, often nonlinear. The relation between a dynamic variable and its transduced equivalent, although desired to be linear for many transducers, is almost always actually nonlinear. As an example, suppose that control valve A is an *equal-percentage* valve. In such valves a given percent change in the valve's stem position (which is what actually the controller controls) produces an equivalent change in flow, hence the name *equal-percentage*. Generally this type of valve does not shut off flow completely in its limit of stem travel. Let Q denote the flow rate through the

valve (in m^3/sec), Q_{min} the minimum flow when the stem of the valve is at the lowest limit of its travel, and Q_{max} the flow rate when the valve is fully open. The ratio $R = (Q_{max}/Q_{min})$ is called the *rangeability* of the valve; it is a parameter specific to a given valve. The actual flow at any given time varies nonlinearly with rangeability and is often given by an exponential expression of the form

$$Q = Q_{min} R^{u/u_{max}} \qquad \text{(E6.1-4)}$$

where u/u_{max} is the ratio of the actual to the maximum control signal sent to the actuator (actually the valve stem position at any given time divided by the maximum position of valve stem). Suppose that control valve A has rangeability $R = 30$. Thus when the control signal is 10% of full range—that is, $(u/u_{max}) = 0.1$—the flow rate according to equation (E6.1-4) becomes

$$Q = Q_{min}(30)^{0.1} = 1.4Q_{min} \qquad \text{(E6.1-5)}$$

We can see from equation (E6.1-5) that at the end of the 10-min interval the controller output is 10% of its maximum value even though the flow through control valve A is 1.4 times the minimum flow through the valve. Actuators in general have such nonlinear characteristics, and furthermore their characteristics change due to aging or other environmental factors. Some of the difficulties in the field utilization of control algorithms, such as the PID level controller here, arise from the collective impact of such changes. Their extent and nature may not be fully known when the controller is designed, tested, and initially deployed. In the course of time the control engineer has to make various judgments about the overall performance of the process control system and, in collaboration with operations and maintenance personnel, intervene to retune gains, repair or replace equipment, revise procedures for operation, and so on. An objective of linguistic control is to make this entire process somewhat easier. It may therefore be seen as irony in the choice of words, but indeed a benefit of *fuzzy* control is introducing even more *clarity* to the development, evaluation, and maintenance of control systems. □

6.2 FUZZY LINGUISTIC CONTROLLERS

The core of a fuzzy controller is a linguistic description prescribing appropriate action for a given state. As we saw in Chapter 5, fuzzy linguistic descriptions involve associations of fuzzy variables and procedures for inferencing. Whereas in a conventional PID controller what is modeled is the physical system or process being controlled, in fuzzy controllers the aim is to incorporate expert human knowledge in the control algorithm. In this sense, a fuzzy controller may be viewed as a real-time expert system—that is, a model of the thinking processes an expert might go through in the course of manipulating the process.

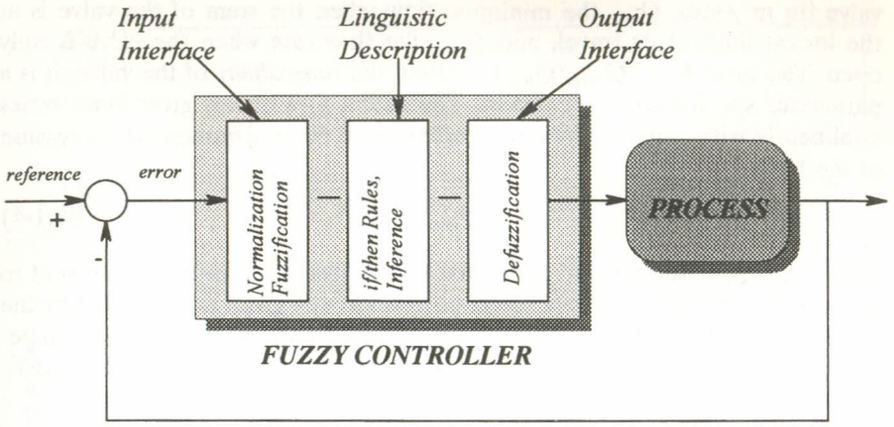


Figure 6.4 Block diagram of fuzzy process control system.

The basic structure of a fuzzy controller is outlined in Figure 6.4. The fact that measuring devices give crisp measurements and that actuators require crisp inputs calls for two additional considerations when linguistic descriptions are employed for control purposes: *fuzzifying* the input of the controller and *defuzzifying* its output. *Fuzzification* can be achieved through a fuzzifier kernel as we saw in Section 2.3, and *defuzzification* can be achieved through special procedures that select a crisp value representative of the fuzzy output (see Section 6.3). Many controllers, however, use directly crisp inputs. Figure 6.4 shows that in addition to a set of *if/then* rules,[2] a fuzzy controller has an *input interface* and an *output interface* handling fuzzification and defuzzification as well as various signal manipulations such as *normalization*, *scaling*, *smoothing*, and *quantization*. Scaling maps the range of values of the controlled variables into predefined universes of discourse, and quantization procedures assist in the mapping when discrete membership functions are used (Larkin, 1985; Efstathiou, 1987; Yager and Filev, 1994).

Fuzzy controllers operate in discrete time intervals. The rules are evaluated at regular intervals in the same way as in conventional digital control, with several rules being executed together (in *parallel*) within the same time interval. This parallel feature makes it possible to develop highly dispersed fuzzy algorithms as we will see later on. We use the subscript k to indicate a specific moment in time—that is, when $t = t_k$. The choice of sampling interval depends on the process being controlled and is usually selected so that at least several significant control actions are made during the process settling time (King and Mamdani, 1977).

Let us look at typical *input* or *left-hand side* (LHS) and *output* or *right-hand side* (RHS) fuzzy variables used in the knowledge base of fuzzy

[2]The set of *if/then* rules is also referred to as the controller's *knowledge base*.

controllers. Many fuzzy controllers use *error*, *change of error*, and *sum of errors* in the LHS,[3] based on measured process variables and setpoint values, and any process variable that can be manipulated directly in the RHS.

Input Variables

The most common LHS variable in fuzzy control is the *error*, or e. It is usually defined on the universe of discourse of crisp error e, which is the deviation of some measured variable y from a setpoint or reference r. At any time $t = k$ crisp error is defined as

$$e(k) \equiv r - y(k) \tag{6.2-1}$$

The change in error, Δe or $\Delta error$, between two successive time steps is also commonly used as an LHS variable. It is defined on the universe of discourse of crisp changes in error. At time $t = k$ the crisp change in error is the difference between present error and error in the previous time step $t = k - 1$, namely,

$$\Delta e(k) \equiv e(k) - e(k-1) \tag{6.2-2}$$

Fuzzy variables can also be defined for the *rate of change in error* $\Delta^2 e(k) \equiv \Delta e(k) - \Delta e(k-1)$, and so on. The *sum of errors* $\bar{e}(k)$ may be used as an LHS fuzzy variable also. It takes into account the integrated effect of all past errors and is defined as

$$\bar{e}(k) \equiv \sum_{i=1}^{k} e_i \tag{6.2-3}$$

In some cases, actual state variables may be used (instead of error, etc.) depending on the availability of parameter and structure estimation knowledge. It is even possible to use variables not directly measurable, such as *performance* or *reliability*, provided that they can be estimated in a timely and reliable manner (Tsoukalas, 1991).

Output Variables

RHS variables may be any directly manipulated variable. An RHS fuzzy variable u can be defined on the universe of discourse of a crisp manipulated variable. Actually the change in output Δu is more often used as the RHS variable. Δu indicates the extent of change of the control variable u at time $t = k$—that is, the *change in action*. Hence, if the defuzzified output at

[3]Using these variables, one can write *if/then* rules emulating PID modes of control.

time k is $\Delta u^*(k)$, the overall crisp output of the controller will be

$$u(k) = u(k-1) + \Delta u^*(k) \tag{6.2-4}$$

Using Δu is preferable, since it requires a smaller number of data points in the output universe of discourse in order for the controller to operate with reasonable accuracy.

if / then Rules and Inference

Often, but not always, LHS and RHS variables are scaled to the same universe of discourse and possess fuzzy values that have the same form. Scaling to a common universe of discourse with a common set of values for all variables may offer considerable savings in memory and speed as far as the computer implementation of a fuzzy algorithm is concerned. In addition, it may be helpful in analyzing the behavior of the controller itself, as we will see later in this chapter. With the advent of fuzzy microprocessors and fuzzy development shells, it is no longer necessary for a user to do scaling because it is done by the system automatically. Nonetheless, scaling helps to simplify algorithmic development and investigate factors involved in synthesis and analysis. As an example, consider the fuzzy values for the variables *error*, Δ*error*, and Δu shown in Figure 6.5 in connection with a fuzzy controller that emulates the derivative mode of a conventional controller (Sugeno, 1985;

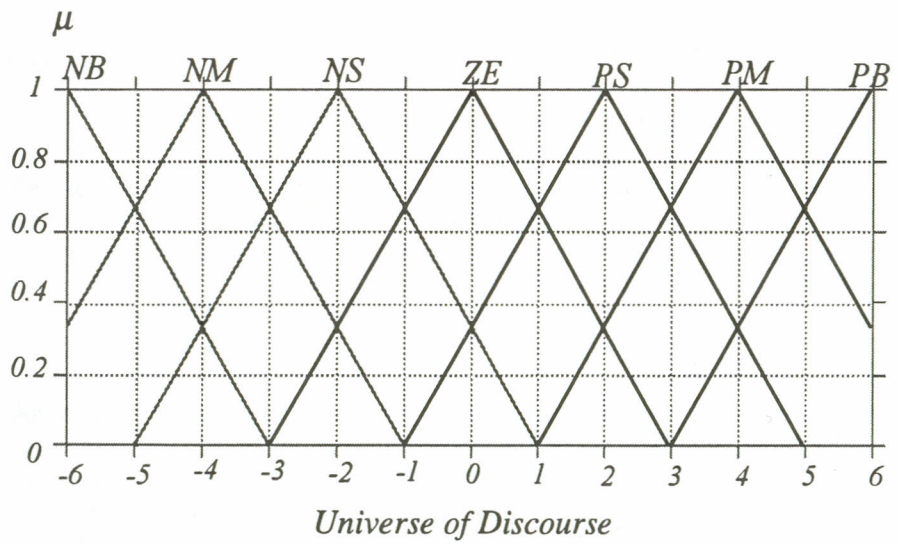


Figure 6.5 Common fuzzy values for the *error*, Δ*error*, and Δu variables, scaled to the same universe of discourse.

Mizumoto, 1988). The common fuzzy values are as follows:

$NB \equiv$ *negative big*,	$PS \equiv$ *positive small*
$NM \equiv$ *negative medium*,	$PM \equiv$ *positive medium*
$NS \equiv$ *negative small*,	$PB \equiv$ *positive big*
$ZE \equiv$ *zero*	

All variables share the same universe of discourse ranging between -6 and $+6$ as shown in Figure 6.5. In computer implementations, fuzzy values are usually quantized and stored in memory in the form of a look-up table as shown in Table 6.1. In this case the fuzzy values are stored in a 7×13 table, with every row in the table representing a quantized fuzzy value. The fuzzy algorithm of a controller that emulates a derivative mode is comprised of the following *if/then* rules:

R_1: *if error is NB AND* Δ*error is ZE then* Δu *is PB ELSE*

R_2: *if error is NM AND* Δ*error is ZE then* Δu *is PM ELSE*

R_3: *if error is NS AND* Δ*error is ZE then* Δu *is PS ELSE*

R_4: *if error is ZE AND* Δ*error is ZE then* Δu *is ZE ELSE*

R_5: *if error is PS AND* Δ*error is ZE then* Δu *is NS ELSE*

R_6: *if error is PM AND* Δ*error is ZE then* Δu *is NM ELSE*

R_7: *if error is PB AND* Δ*error is ZE then* Δu *is NB ELSE* (6.2-5)

R_8: *if error is ZE AND* Δ*error is NB then* Δu *is PB ELSE*

R_9: *if error is ZE AND* Δ*error is NM then* Δu *is PM ELSE*

R_{10}: *if error is ZE AND* Δ*error is NS then* Δu *is PS ELSE*

R_{11}: *if error is ZE AND* Δ*error is PS then* Δu *is NS ELSE*

R_{12}: *if error is ZE AND* Δ*error is PM then* Δu *is NM ELSE*

R_{13}: *if error is ZE AND* Δ*error is PB then* Δu *is NB*

When two LHS and one RHS variables are used as in (6.2-5), the algorithm can be visualized in the form of a table as shown in Table 6.2. Such an arrangement is sometimes called a "fuzzy associative memory (FAM) matrix." Blank items in the table indicate that there is no rule present for the particular combination of LHS variables. Obviously for algorithms with more than two LHS variables a tabular representation requires additional dimensions.

Table 6.1 Table of fuzzy values

	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
NB	1	0.67	0.33	0	0	0	0	0	0	0	0	0	0
NM	0.33	0.67	1	0.67	0.33	0	0	0	0	0	0	0	0
NS	0	0	0.33	0.67	1	0.67	0.33	0	0	0	0	0	0
ZE	0	0	0	0	0.33	0.67	1	0.67	0.33	0	0	0	0
PS	0	0	0	0	0	0	0.33	0.67	1	0.67	0.33	0	0
PM	0	0	0	0	0	0	0	0	0.33	0.67	1	0.67	0.33
PB	0	0	0	0	0	0	0	0	0	0	0.33	0.67	1

Table 6.2 A fuzzy algorithm in tabular form

Δerror / error	NB	NM	NS	ZE	PS	PM	PB
NB				PB			
NM				PM			
NS				PS			
ZE	PB	PM	PS	ZE	NS	NM	NB
PS				NS			
PM				NM			
PB				NB			

Fuzzy control algorithms are evaluated using *generalized modus ponens* (GMP). We recall from Chapter 5 that GMP is a data-driven inferencing procedure that analytically involves the composition of fuzzy relations, usually *max–min composition*. We also saw that max–min composition under a given implication operator affects the RHS in a specific manner—for example, by *clipping* (when *Mamdani min*, ϕ_c, is used) or *scaling* (when *Larsen product*, ϕ_p, is used). In general, GMP can be thought of as a transformation

of the RHS by a degree commensurate with the degree of fulfillment (DOF) of the rule and in a manner dictated by the implication operator chosen (see Examples 5.1 and 5.2). In this chapter, instead of explicitly using composition operations, we will mostly focus on such transformations as is often done, for the sake of convenience, in many fuzzy control applications. As far as the entire algorithm is concerned, the connective *ELSE* is analytically modeled as either *OR* ($\vee$) or *AND* ($\wedge$), again depending on the implication operator used for the individual *if/then* rules. For example, when the Mamdani min implication is used, the connective *ELSE* is interpreted as *OR* (see Table 5.5).

Fuzzy controller inputs are usually crisp numbers. Fuzzy inputs may also be considered in the case of uncertain or noisy measurements and crisp numbers may be fuzzified (see Section 2.3). Consider the situation shown in Figure 6.6 involving rules R_3, R_4, and R_{11} of (6.2-5). When at time $t = k$ crisp error e' and crisp change in error $\Delta e'$ as shown in Figure 6.6 are given to these rules we say that the rules have "fired," provided that their DOF is not zero. For example, in rule R_3 the crisp error e' shown has a 0.8 degree of membership to *NS* while the crisp change in error $\Delta e'$ has a 0.6 degree of membership to *ZE*. Thus the degree of fulfillment of rule R_3 at this particular time is

$$\text{DOF}_3 = \mu_{NS}(e') \wedge \mu_{ZE}(\Delta e') = 0.8 \wedge 0.6 = 0.6 \qquad (6.2\text{-}6)$$

Provided that we interpreted the LHS connective *AND* as min ($\wedge$) [a common alternative is *product* ($\cdot$)], the RHS value *PS* will be transformed in accordance with DOF_3 in equation (6.2-6). The nature of the transformation depends on the implication used as we saw in Chapter 5. When Mamdani min is used the transformation amounts to clipping *PS* at the height of DOF_3 as shown in Figure 6.6. Thus R_3 contributes $\mu_{PS'}(\Delta u)$, the shaded part of the RHS value, to the total fuzzy output. Similarly rules R_4 and R_{11} have degrees of fulfillment

$$\text{DOF}_4 = \mu_{ZE}(e') \wedge \mu_{ZE}(\Delta e') = 0.4 \wedge 0.6 = 0.4 \qquad (6.2\text{-}7)$$

$$\text{DOF}_{11} = \mu_{ZE}(e') \wedge \mu_{PS}(\Delta e') = 0.4 \wedge 1.0 = 0.4 \qquad (6.2\text{-}8)$$

and they contribute $\mu_{ZE'}(\Delta u)$ and $\mu_{NS'}(\Delta u)$, shown as shaded parts of the RHS values in Figure 6.6. The rest of the rules of algorithm (6.2-5) do not fire, that is, they contribute a zero output. The total fuzzy output is the *union* of the three outputs since we interpret the connective *ELSE* in (6.2-5) as *OR* ($\vee$)—that is,

$$\mu_{OUT}(\Delta u) = \mu_{PS'}(\Delta u) \vee \mu_{ZE'}(\Delta u) \vee \mu_{NS'}(\Delta u) \qquad (6.2\text{-}9)$$

$\mu_{OUT}(\Delta u)$ is shown at the lower part of Figure 6.6. At this point we need to defuzzify $\mu_{OUT}(\Delta u)$ and obtain a crisp value Δu_k^* representative of $\mu_{OUT}(\Delta u)$ to be used as input to the process. In the next section we will look at different methods for defuzzification.

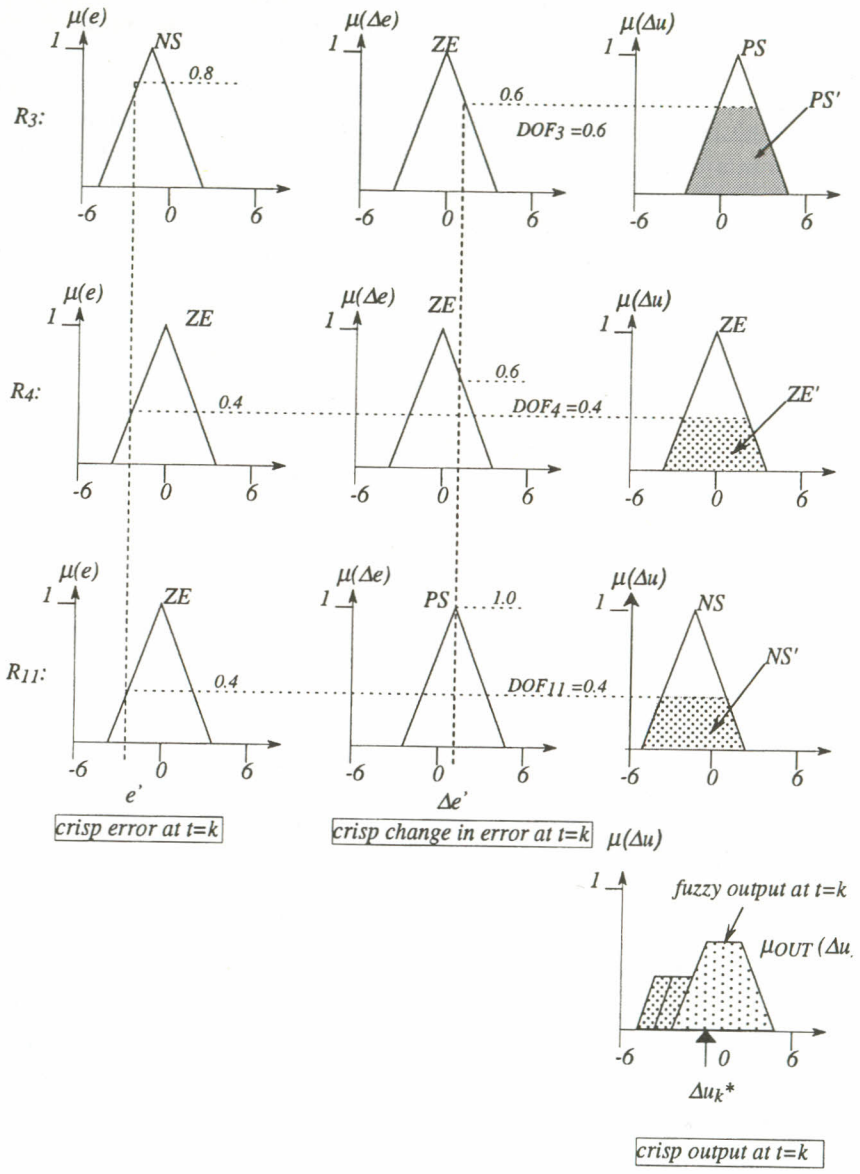


Figure 6.6 Evaluation of three control rules at time $t = k$ using Mamdani min implication and min interpretation of *AND* (DOF).

If *Larsen product* is used as the fuzzy implication operator for the individual rules of (6.2-5), the membership function of the RHS value is *scaled* by the degree of fulfillment of each rule as shown in Figure 6.7 (see Example 5.2). Since the connective *ELSE* is interpreted as *OR* ($\vee$) when Larsen product implication is used (see Table 5.5), the total output $\mu_{OUT}(\Delta u)$ is also the *union* of the three individual outputs. Out of that we need to select a representative crisp value as input to the process. We note in Figure 6.7 that $\mu_{OUT}(\Delta u)$ looks quite different from the total fuzzy output obtained using Mamdani min shown in Figure 6.6. Other fuzzy implication operators (see Table 5.2) would produce different transformations in the shape of the RHS fuzzy value and, hence, a different $\mu_{OUT}(\Delta u)$.

Interpretations of *AND* other than min ($\wedge$) may be used in the *AND* connective found in the LHS of the rules, hence obtaining different degrees of fulfillment. Arithmetic product has been used (particularly in conjunction with max-product implication) and more generally *T-norms* (Zimmermann, 1985; Fuller and Zimmermann, 1992).[4] Using arithmetic product the degree of fulfillment for the rules of (6.2-5) that fire would be evaluated in the manner shown in Figure 6.8. The degree of fulfillment for R_3, R_4, and R_{11} are

$$\begin{aligned} \text{DOF}_3 &= \mu_{NS}(e') \cdot \mu_{ZE}(\Delta e') = 0.8 \cdot 0.6 = 0.48 \\ \text{DOF}_4 &= \mu_{ZE}(e') \cdot \mu_{ZE}(\Delta e') = 0.4 \cdot 0.6 = 0.24 \\ \text{DOF}_{11} &= \mu_{ZE}(e') \cdot \mu_{PS}(\Delta e') = 0.4 \cdot 1.0 = 0.4 \end{aligned} \qquad (6.2\text{-}10)$$

Comparing equations (6.2-10) with equations (6.2-6)–(6.2-8) we see that generally the two different interpretations of *AND* lead to different results under the same fuzzy implication operators as we can also see by comparing Figures 6.7 and 6.8.

After we defuzzify $\mu_{OUT}(\Delta u)$ by one of the methods discussed in the next section, we obtain a crisp value Δu_k^*, which in the case of the algorithm of (6.2-5) would be an integer between -6 and $+6$. Values greater than the extremes of the universe of discourse are set to the extreme values, in this case -6 or $+6$. This value is then multiplied by a scaling factor that maps it into the appropriate range of the manipulated variable before using it to actuate a device (Larkin, 1985).

Since so much of actual process control knowledge has historically been obtained through PID controllers, it is often convenient to emulate various modes and combinations of the PID controller by fuzzy rules. Thus a fuzzy controller emulating a conventional PD mode of control controller would

[4]See the Appendix for an introduction to *T norms* and their co-norms, called *S norms*.

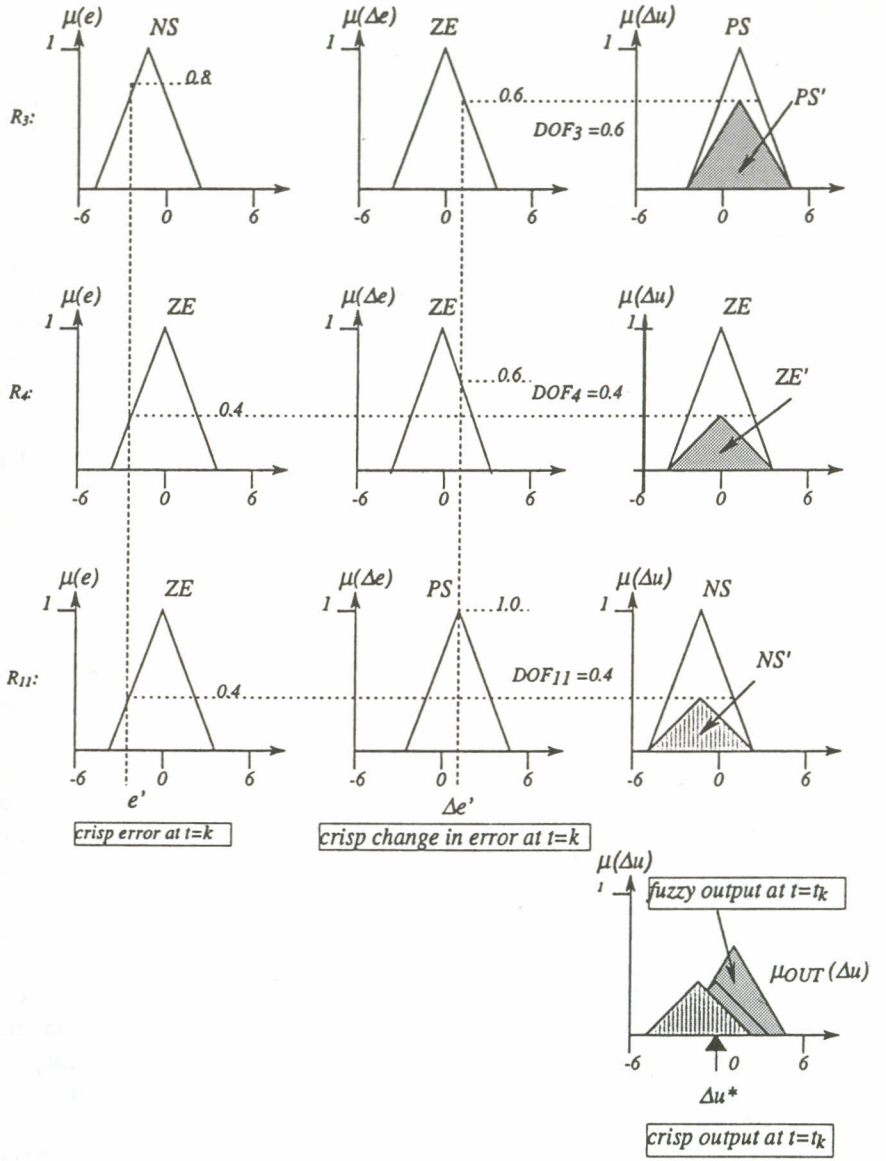


Figure 6.7 Evaluation of three control rules at time $t = k$ Lusing Larsen product implication and min DOF.

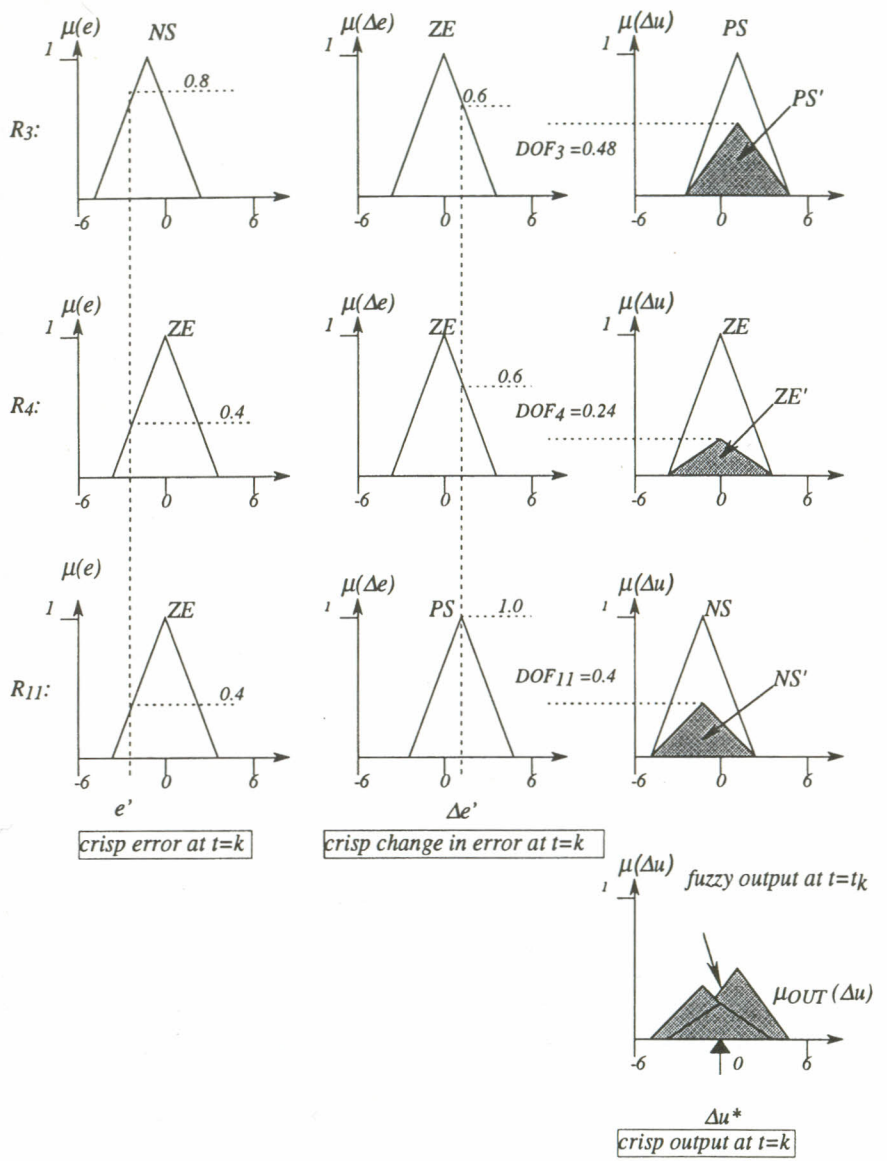


Figure 6.8 Evaluation of three control rules at time $t = k$ using Larsen product implication and product DOF.

consist of rules having the form

$$\textit{if}\quad e\ \textit{is}\ A\quad \textit{AND}\quad \Delta e\ \textit{is}\ B\quad \textit{then}\ u\ \textit{is}\ C \tag{6.2-11}$$

where e is the error and Δe is the change in error. A PI-like fuzzy controller would have rules of the form

$$\textit{if}\quad e\ \textit{is}\ A\quad \textit{AND}\quad \Delta e\ \textit{is}\ B\quad \textit{then}\ \Delta u\ \textit{is}\ C \tag{6.2-12}$$

while a P-like controller would have rules

$$\textit{if}\quad e\ \textit{is}\ A\quad \textit{then}\quad u\ \textit{is}\ C \tag{6.2-13}$$

The rule form of a PID-like fuzzy controller is

$$\textit{if}\quad e\ \textit{is}\ A\quad \textit{AND}\quad \Delta e\ \textit{is}\ B\quad \textit{AND}\quad \bar{e}\ \textit{is}\ C\quad \textit{then}\quad u\ \textit{is}\ D \tag{6.2-14}$$

where $\bar{e}$ is the sum of errors.

Although we have formulated fuzzy algorithms in terms of rules involving fuzzy values on their RHS (such rules are referred to as *Mamdani rules*), there are advantages to consider crisp or special shape membership functions as well. Several fuzzy controllers use rules where the output variable is given in terms of a functional relation of the inputs. This is known as the *Sugeno* or *TSK*[5] *form* of fuzzy rules. Such rules are typically written as

$$\textit{if}\quad x_1\ \textit{is}\ A_1\quad \textit{AND}\quad x_2\ \textit{is}\ A_2 \ldots \textit{then}\quad u = f(x_1, \ldots, x_n) \tag{6.2-15}$$

where f is a function of the inputs $x_1, \ldots, x_n$. When $f(x_1, \ldots, x_n)$ is a constant, rules of the form (6.1-15) constitute a *zero-order Sugeno controller*. When $f(x_1, \ldots, x_n)$ is a first-order polynomial we have what is called a *first-order Sugeno controller*. For example, we may describe a PI controller of (6.2-12) by rules of the form

$$\textit{if}\quad e\ \textit{is LARGE}\quad \textit{AND}\quad \Delta e\ \textit{is MEDIUM}\quad \textit{then}\ u = 2e + 3\Delta e \tag{6.2-16}$$

An interesting application of Sugeno rules is when a PID controller is put directly in the RHS of (6.2-15). The result is a fuzzy "supervisor" changing the parameters of a PID controller [see Tzafestas and Papanikolopoulos (1990)]. Sugeno fuzzy models are well suited for modeling nonlinear systems by interpolating multiple linear models and are also well suited to mathematical analysis and lend themselves to adaptive techniques, whereas Mamdani rules are more intuitive and better suited to human. [See Jang and Sun (1995) and Jang and Gulley (1995) for a review of different controllers.]

[5]After the *Takagi*, *Sugeno*, and *Kang* who first proposed it in 1985; also referred to as the *Sugeno fuzzy model*.

An alternative to either Sugeno or Mamdani rules is to consider rules whose consequents employ monotonical membership functions. This form of rules is known as the *Tsukamoto fuzzy model* (Tsukamoto, 1979). In Tsukamoto rules the inferred output of each rule is a crisp value equal to the rule's *degree of fulfillment*, with the overall output being taken as the weighted average of all outputs (a crisp value).

Fuzzy algorithms such as (6.2-5) are inherently parallel in the sense that individual *if/then* rules are fired independent of each other, with a specific input being processed by several rules each contributing to a collective result, namely, $\mu_{OUT}(\Delta u)$. Actual process systems, however, may have many inputs and outputs, and hence they are referred to as *many-input-many-output* (MIMO) systems. The question then arises of how relevant are the rather simple *if/then* rules we have seen thus far, such as the algorithm of (6.2-5), to the control of such systems and what happens to parallelism at a higher level of system complexity.

Generally the control strategies of complicated process systems may be organized in such a manner that relatively simple *if/then* rules are used (Terano, 1992; Yager, 1994). This is achieved by partitioning the knowledge base of the controller into rule clusters. In each cluster there are *if/then* rules that may have several LHS variables but only one RHS variable. Suppose that we have p input variables $x_1, x_2, \ldots, x_p$ and r manipulated variables $u_1, u_2, \ldots, u_r$. The algorithmic development generally proceeds from some general and maybe complicated *if/then* rules that form the *a priori* knowledge prescribing what has to be done under a set of hypothetical situations. Often, but not always, it is possible to reduce these initial rules to simpler rules with one control variable in the RHS. Rules that have the same RHS variable are collected together to form a rule cluster. In the end we have one cluster of rules whose RHS is used to manipulate variable u_1, another for variable u_2, and so on. Thus, a complicated process control system may be decomposed into a number of *many-input-single-output* controllers. Such rule clusters may be executed independently, hence maintaining the overall parallel characteristics of fuzzy systems. Of course, more elaborate architectures can be devised that may include *metarules*. The developers of fuzzy control algorithms exercise considerable creativity in setting up special variables and rules for the interaction of these clusters. In principle, however, rule clusters can be noninteractive, in which case they can be executed in parallel, achieving considerable speed and computational efficiency.

6.3 DEFUZZIFICATION METHODS

After the input to the controller has been processed by the control algorithm the result is a fuzzy output $\mu_{OUT}(u)$. Selecting a crisp number u^* representative of $\mu_{OUT}(u)$ is a process known as *defuzzification*. Over the years several

defuzzification techniques have been suggested (Terano et al., 1992; Pedrycz, 1993; Yager and Filev, 1994). The choice of defuzzification method may have a significant impact on the speed and accuracy of a fuzzy controller.[6] The most frequently used ones are the *centroid* or *center of area* (COA), the *center of sums* (COS), and *mean of maxima* (MOM).

Center of Area (COA) Defuzzification

In COA defuzzification[7] the crisp value u^* is taken to be the geometrical center of the output fuzzy value $\mu_{OUT}(u)$, where $\mu_{OUT}(u)$ is formed by taking the union of all the contributions of rules whose DOF > 0.[8] The center is the point which splits the area under the $\mu_{OUT}(u)$ curve in two equal parts. Let us assume we have a discretized universe of discourse. The defuzzified output is defined as

$$u^* = \frac{\sum_{i=1}^{N} u_i \, \mu_{OUT}(u_i)}{\sum_{i=1}^{N} \mu_{OUT}(u_i)} \tag{6.3-1}$$

where the summation (integration) is carried over (discrete) values of the universe of discourse u_i sampled at N points. COA is a well known and often used defuzzification method. Some potential drawbacks of COA are that it favors "central" values in the universe of discourse and that, due to its complexity, it may lead to rather slow inference cycles. COA defuzzification takes into account the area of the resultant membership function $\mu_{OUT}(u)$ as a whole. If the areas of two or more contributing rules overlap, equation (6.3-1) does not take into account the overlapping area only once [since we take the union to form $\mu_{OUT}(u)$, the resultant membership function].

When $\mu_{OUT}(u) = 0$ we simply set the crisp output to a pre-agreed value (in order to avoid dividing by zero), typically $u^* = 0$. The crisp output value may also be computed in terms of the DOF of each contributing rule as

$$u^* = \frac{\sum_{k=1}^{n} \mathrm{DOF}_k \cdot M_k}{\sum_{k=1}^{n} \mathrm{DOF}_k \cdot B'_k} \tag{6.3-2}$$

where B'_k is the contribution due to the firing of rule k,[9] M_k is the moment of B'_k, and DOF_k is the *degree of fulfillment* of the kth rule ($k = 1, \ldots, n$).

[6] Certain defuzzification methods may introduce nonlinearities and discontinuities in the control hypersurface [see Jager (1995)].

[7] Also known as *center of gravity* defuzzification, a name more appropriate for multidimensional fuzzy output.

[8] For convenience we use the control signal u as the output variable. The control signal change Δu or any other output variable may be used as well.

[9] The subscript k is used to indicate the kth rule and should not be confused with the letter k used earlier to indicate the $t = k$ time step.

We recall that the moment of B'_k is the product of B'_k and the distance of its center of gravity from the μ axis (the moment about zero).

Center of Sums (COS) Defuzzification

To address the problems associated with COA and take into account the overlapping areas of multiple rules more than once, a variant of COA called *center of sums* (COS) is used. As shown in Figure 6.9, COS builds the resultant membership function by taking the sum (not just the union) of output from each contributing rule. Hence overlapping areas are counted more than once. COS is actually the most commonly used defuzzification method. It can be implemented easily and leads to rather fast inference cycles. It is given by

$$u^* = \frac{\sum_{i=1}^{N} u_i \cdot \sum_{k=1}^{n} \mu_{B'_k}(u_i)}{\sum_{i=1}^{N} \sum_{k=1}^{n} \mu_{B'_k}(u_i)} \tag{6.3-3}$$

where $\mu_{B'_k}(u_i)$ is the membership function (at point u_i of the universe of discourse) resulting from the firing of the kth rule.

Mean of Maxima (MOM) Defuzzification

One simple way to defuzzify the output is to take the crisp value with the highest degree of membership in $\mu_{OUT}(u)$. Oftentimes, however, there may be more than one element in the universe of discourse having the maximum value, as may be seen in the $\mu_{OUT}(u)$ of Figure 6.6. In such cases we can randomly select one of them or, even better, take the mean value of the maxima. Suppose that we have M such maxima in a discrete universe of discourse. The crisp output can be obtained by

$$u^* = \sum_{m=1}^{M} \frac{u_m}{M} \tag{6.3-4}$$

where u_m is the mth element in the universe of discourse where the membership function of $\mu_{OUT}(u)$ is at the maximum value, and M is the total number of such elements.

MOM defuzzification is faster than COA, and furthermore it allows the controller to reach values near the edges of the universe of discourse. A disadvantage of this method, however, is that it does not consider the overall shape of the fuzzy output $\mu_{OUT}(u)$. On the other hand, with COA the extreme values of the universe of discourse cannot be reached—for example, near ± 6 in Figure 6.5. Both methods have been used in control applications; several variants of them exist, such as the *indexed center of gravity method*, where a threshold level is used to eliminate elements with degrees of membership lower than a threshold in the computation of $\mu_{OUT}(\Delta u)$

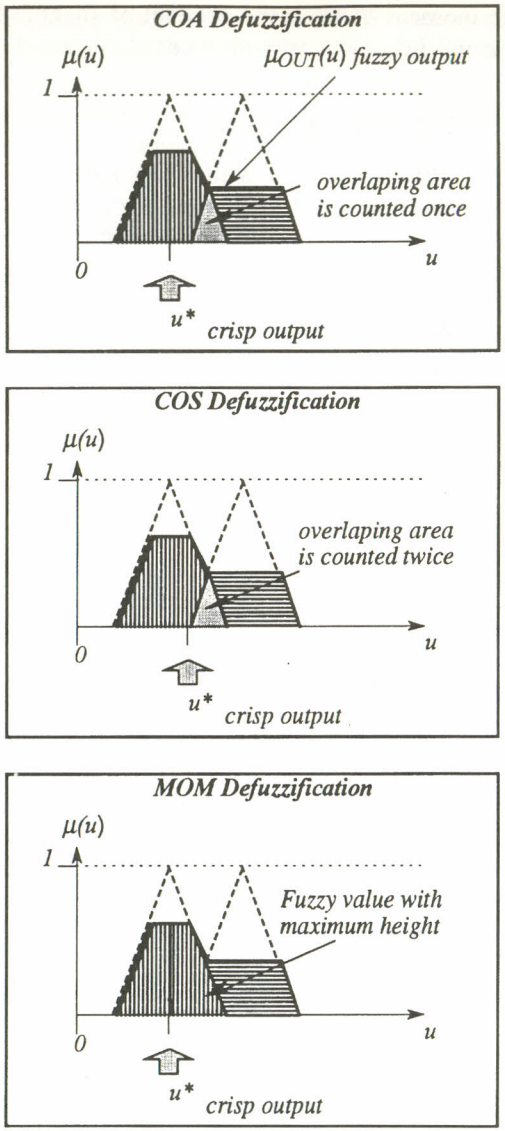


Figure 6.9 Three different defuzzification methods: center of area (COA, center of sums (COS), and mean of maxima (MOM).

(Pedrycz, 1993). It is also possible to employ defuzzification methods in an adaptive manner (Yager and Filev, 1993, 1994).

Example 6.2 A Simple Fuzzy Controller for Level Control. Consider the process system shown in Figure 6.1 (and controlled by a PID controller in

Example 6.1). We want to develop a fuzzy controller to control the flow of liquid in the tank and maintain its level at a reference value. The simplest possible fuzzy controller would have only one LHS variable and one RHS variable. We define fuzzy variables *error* and *output* for the LHS and RHS, respectively, as shown in Figure 6.10. It should be noted that the simplicity of the problem does not call for scaling our variables to a common universe of discourse. The universe of discourse for *error* is made of crisp percent error values from -20% to 20%. The fuzzy values of the variable *error* are: *NB* (*negative big*), *NS* (*negative small*), *Z* (*zero*), *PS* (*positive small*), and *PB* (*positive big*). The universe of discourse for *output* is the crisp values of output ranging from 0% to 100%, with fuzzy values: *VL* (*very low*), *LOW*, *MED* (*medium*), *HIGH*, and *VH* (*very high*). The fuzzy algorithm is com-

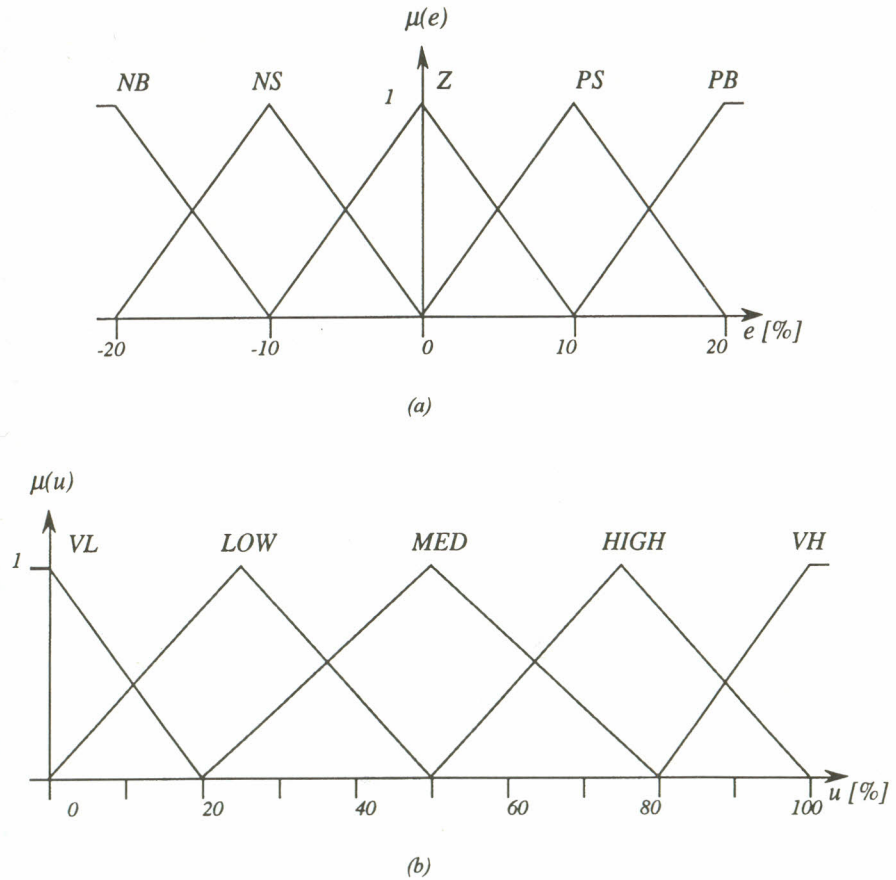


Figure 6.10 Fuzzy values for (*a*) *error* and (*b*) *output* fuzzy variables used in the rules of Example 6.2

prised of the following rules:

$$
\begin{aligned}
R_1&: \textit{ if error is NB then output is VH ELSE}\\
R_2&: \textit{ if error is NS then output is HIGH ELSE}\\
R_3&: \textit{ if error is Z then output is MED ELSE}\\
R_4&: \textit{ if error is PS then output is LOW ELSE}\\
R_5&: \textit{ if error is PB then output is VL}
\end{aligned}
\qquad \text{(E6.2-1)}
$$

We use Mamdani min for fuzzy implication and hence interpret the connective *ELSE* in (E6.2-1) as *OR* (see Table 5.5). Suppose that at time $t = 0$ min the output of our controller was at 50% of its full range and 2 min later we introduce the error shown in Figure 6.2. What would be the output according to the algorithm of (E6.2-1)? Let us look at what happens at $t = 3$ min. From Figure 6.2 we see that the input to the algorithm at this time is a crisp error $e_k = 1.25\%$, which belongs to the Z value of *error* to a degree of 0.87 and to *PS* to a degree of 0.12. The degree of membership to other fuzzy values is zero, as can be seen in Figure 6.10. Thus, rules R_3 and R_4 of algorithm (E6.2-1) will fire, since they are the only rules involving the Z and *PS* values. The situation is shown in Figure 6.11, where we see a schematic (geometrical) rendition of the evaluation of the control algorithm under GMP at time $t =$ 3 min. The *degree of fulfillment* of R_3 is $\text{DOF}_3 = 0.87$ and for R_4 we have $\text{DOF}_4 = 0.12$. All other rules have DOF = 0.0. Using Mamdani min implication the result of evaluating rules R_3 and R_4 under GMP is to *clip* the RHS values of rules R_3 and R_4—that is, *MED* and *LOW*—at $\mu_{LOW}(u) = 0.87$ and $\mu_{MED}(u) = 0.12$, respectively. In other words, *MED* is clipped at 0.12. Out of all rules, only R_3 and R_4 contribute at $t = 3$ min, and their contributions are $\mu_{LOW'}(u)$ and $\mu_{MED'}(u)$ as shown in Figure 6.11. Since we interpret the connective *ELSE* as *OR*, the total fuzzy output of the entire algorithm at $t = 3$ min is the union of these two values—that is,

$$\mu_{OUT}(u) = \mu_{LOW'}(u) \vee \mu_{MED'}(u) \qquad \text{(E6.2-2)}$$

$\mu_{OUT}(u)$ is shown in the lower part of Figure 6.11. We use the COA defuzzification—that is, equation (6.3-2)—to defuzzify $\mu_{OUT}(u)$. The result is $u_3^* = 47\%$. If MOM is used, the average value of the maximum values of $\mu_{OUT}(u)$ is at the middle of the plateau of $\mu'_{MED}(u)$—that is, at about 50%. Hence, the two methods give somewhat different results. In MOM the contribution of $\mu'_{LOW}(u)$ is totally ignored since only the values where $\mu_{OUT}(u)$ is at a maximum are taken into account. The above procedure is repeated in subsequent time steps. The defuzzified output of the controller using COA is shown in Figure 6.12. Comparing with the three different modes of PID control shown in Figure 6.3, we see that our fuzzy controller

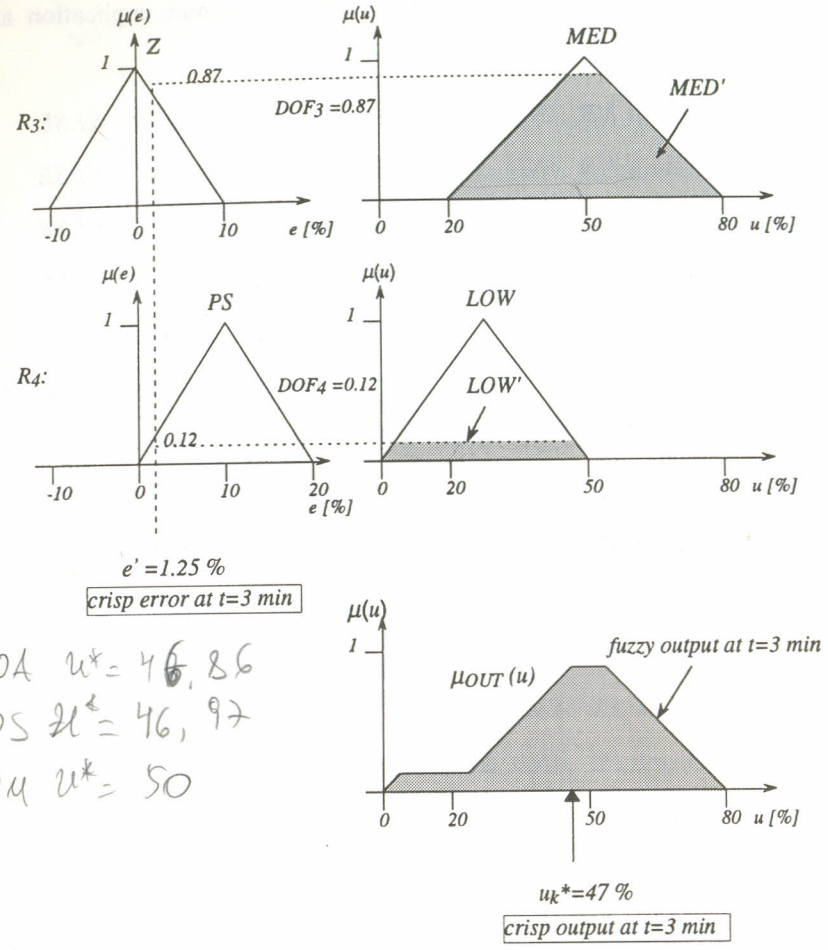


Figure 6.11 Output calculation at $t = 3$ min in Example 6.2.

behaves like a proportional controller, since the form (not the magnitude) of the output is similar to $u_p(t)$. □

Example 6.3 A Two-Input Fuzzy Controller for Level Control. Consider the process system shown in Figure 6.1, addressed by a PID controller in Example 6.1 and a simple fuzzy controller in Example 6.2. Let us now develop a fuzzy controller with two LHS and one RHS variables. We use *error* and change in error, $\Delta error$, in the LHS of the rules and use *output* in the RHS. The fuzzy values of these variables are shown in Figure 6.13. $\Delta error$ express the direction of change in error; that is, *increasing* is described by the fuzzy value P (*positive*), *decreasing* is described by N (*negative*), and *no*

change is described by *ZE* (*zero*). We use Mamdani min implication and COA defuzzification. The fuzzy algorithm is:

R_1: *if error is NB AND Δerror is N then output is HIGH ELSE*

R_2: *if error is NB AND Δerror is ZE then output is VH ELSE*

R_3: *if error is NB AND Δerror is P then output is VH ELSE*

R_4: *if error is NS AND Δerror is N then output is HIGH ELSE*

R_5: *if error is NS AND Δerror is ZE then output is HIGH ELSE*

R_6: *if error is NS AND Δerror is P then output is MED ELSE*

R_7: *if error is Z AND Δerror is N then output is MED ELSE*

R_8: *if error is Z AND Δerror is ZE then output is MED ELSE*

R_9: *if error is Z AND Δerror is P then output is MED ELSE*

R_{10}: *if error is PS AND Δerror is N then output is MED ELSE*

R_{11}: *if error is PS AND Δerror is ZE then output is LOW ELSE*

R_{12}: *if error is PS AND Δerror is P then output is LOW ELSE*

R_{13}: *if error is PB AND Δerror is N then output is LOW ELSE*

R_{14}: *if error is PB AND Δerror is ZE then output is VL ELSE*

R_{15}: *if error is PB AND Δerror is P then output is VL*

(E6.3-1)

We recall that with Mamdani min the connective *ELSE* in (E6.3-1) is interpreted as *OR* and therefore the total fuzzy output will be the *union* of individual rule contributions (see Table 5.5). It is customary in the control literature to refer to the fuzzy relations (E6.3-1) as *control surfaces* (or *hypersurfaces*). Figure 6.14 is a graphical representation of the control surface indicating hypersurface dependence on the rules. In Figure 6.14*a*, no rules exist in our algorithm; hence the control hypersurface is a flat plane at $u = 0$. If the control algorithm in (E6.3-1) was comprised only of the two rules R_1 and R_2 (the rest did not exist), then the control hypersurface would look like what is shown in Figure 6.14*b*. If only the first eight rules of the algorithm are present, the control hypersurface looks like Figure 6.14*c*, while if the first 13 rules are present, the control hypersurface would look like Figure 6.14*d*. Finally, if all 15 rules are present, the control hypersurface looks like

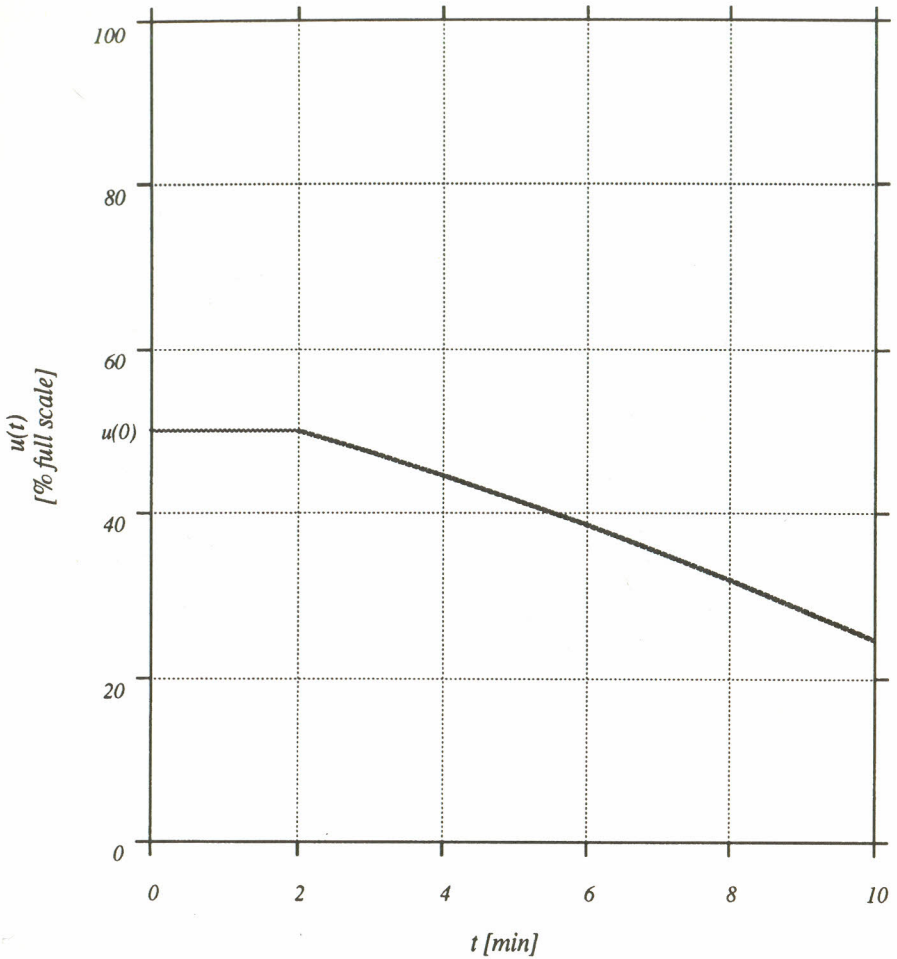
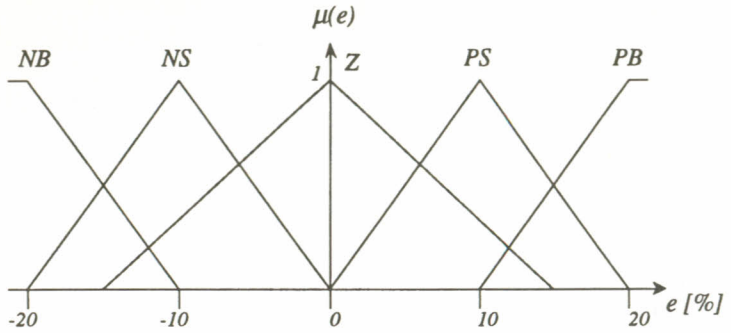


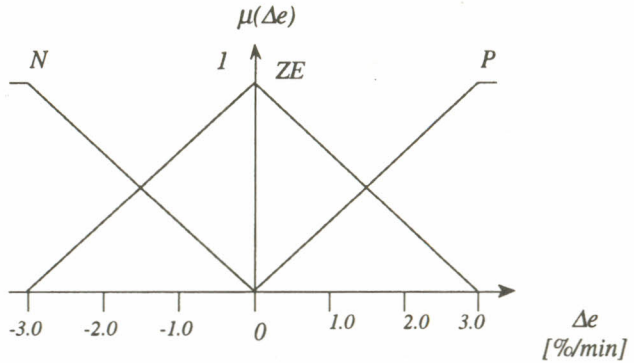
Figure 6.12 Defuzzified output of fuzzy level controller with one input and one output.

Figure 6.14*e*. Plotting the control hypersurface helps to visualize the manner in which a fuzzy controller covers the control space. Unfortunately it is not convenient to use when more than three variables are present.

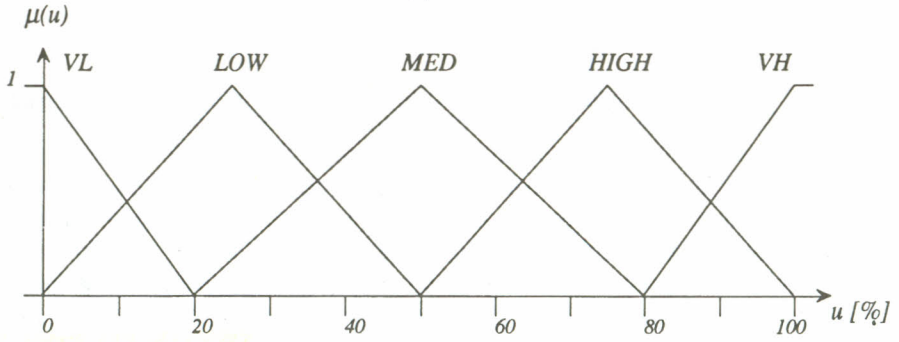
At $t = 2$ min we introduce the error shown in Figure 6.2 (same as in Examples 6.1 and 6.2). Figure 6.15 shows a schematic representation of the fuzzy inference or *generalized modus ponens* (GMP) at $t = 5$ min. Crisp inputs $e' = 3.75\%$ and $\Delta e' = 1.25\%$ are presented to the algorithm (E6.3-1) at this time. Crisp error $e' = 3.75\%$ belongs to fuzzy value Z to degree of 0.75 and to fuzzy value *PS* to a degree of 0.4. Similarly, crisp change-in-error $\Delta e' = 1.25\%$ belongs to fuzzy value *ZE* to a degree of 0.6 and to fuzzy value P to a degree of 0.4. Hence the only rules that will have DOF greater than



(a)

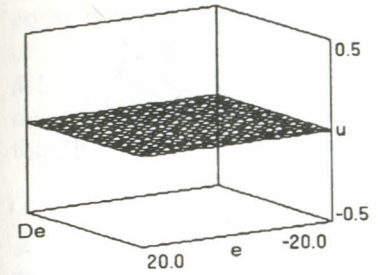


(b)

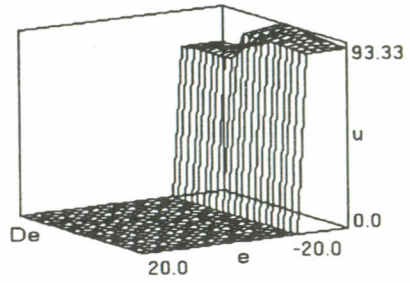


(c)

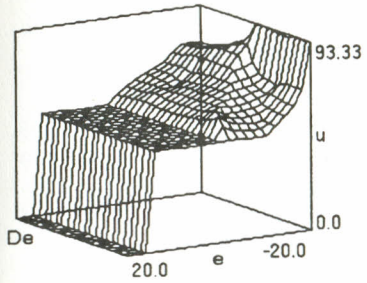
Figure 6.13 Fuzzy values for (*a*) *error*, (*b*) Δ*error*, and (*c*) *output* fuzzy variables used in Example 6.3.



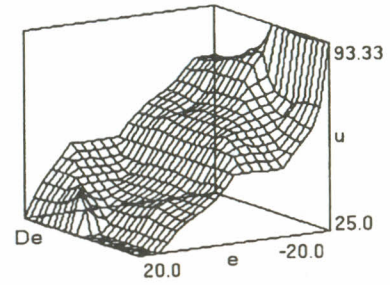
(a)



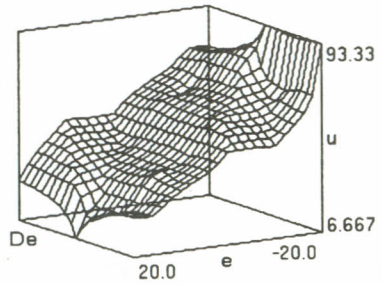
(b)



(c)



(d)



(e)

Figure 6.14 (*a*) The control hypersurface when there are no rules present, (*b*) with rules R_1 and R_2 only, (*c*) with rules R_1 through R_8, (*d*) with rules R_1 through R_{13}, and (*e*) with all 15 rules.

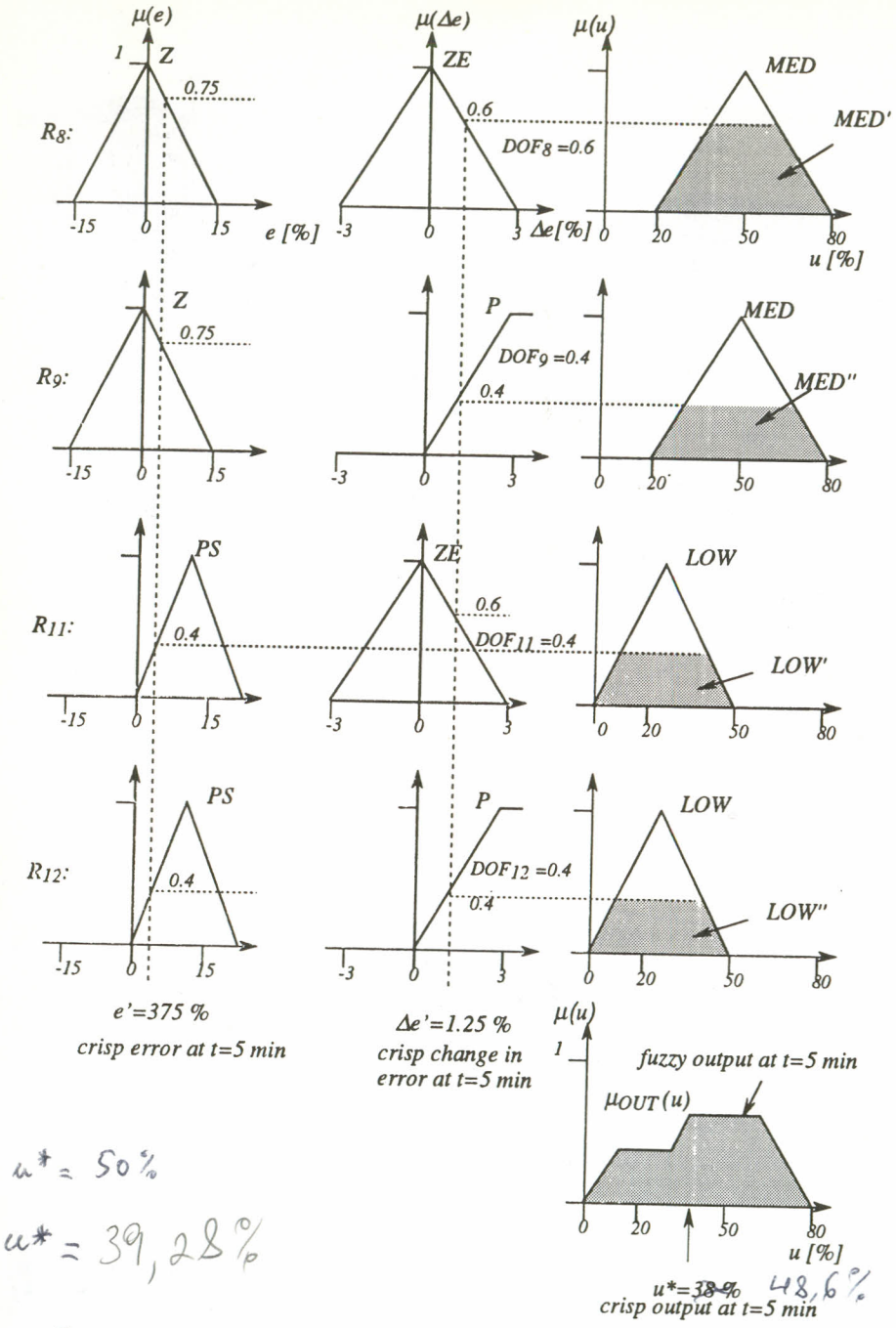


Figure 6.15 Evaluating the fuzzy algorithm of Example 6.3 at time $t = 5$ min.

zero in (E6.3-1)—that is, the rules that fire—will be R_8, R_9, R_{11}, and R_{12}. Using the min form of DOF (i.e., the min ($\wedge$) interpretation of *AND*), each rule contributes the shaded part of the RHS value shown in Figure 6.14. We recall that GMP with Mamdani min implication clips the RHS at the height of DOF, as shown in Figure 6.15. Rule R_8 contributes *MED'*, R_9 contributes *MED''*, R_{11} contributes *LOW'*, and R_{12} contributes *LOW''*. The fuzzy output $\mu_{out}(u)$ is the union (max) of these four contributions (shaded parts); that is,

$$\mu_{out}(u) = \mu_{MED'}(u) \vee \mu_{MED''}(u) \vee \mu_{LOW'}(u) \vee \mu_{LOW''}(u) \quad \text{(E6.3-2)}$$

$\mu_{out}(u)$ is shown at the lower part of Figure 6.15. Using COA defuzzification, we obtain the crisp output $u^* = 38\%$. The procedure is repeated for other time steps. The crisp output of the controller for the duration of the problem is shown in Figure 6.16. Comparing with Figure 6.13, we see that introducing

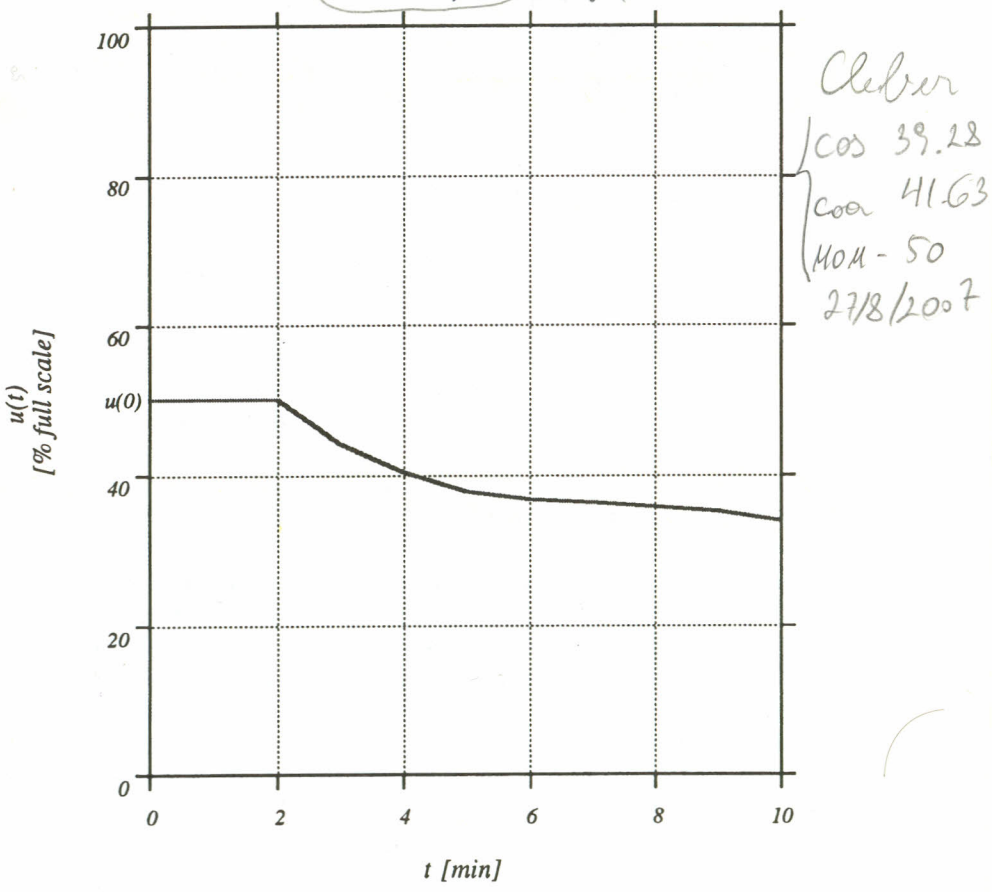


Figure 6.16 Defuzzified output of fuzzy controller with two inputs (*error* and *Δerror*) under ramp input.

one more variable, namely the change in error $\Delta error$, makes a significant difference in the control action. To obtain a desired response by our controller, we can now modify the shape and position of constituent membership functions or the rules, the implications used, and so on. A number of factors contribute to different outputs, such as the knowledge encoded in (E6.3-1), the fuzzy values used, the interpretation of *AND* (affecting DOF), the implication operator, and the defuzzification method used. The role and significance of these factors will be examined in the next section. □

6.4 ISSUES INVOLVED IN DESIGNING FUZZY CONTROLLERS

Although there are automatic ways of identifying the rules and membership functions involved in a fuzzy controller,[10] in many ways the development of a good fuzzy controller reflects the maturity of knowledge about a process. The choice of fuzzy variables and values and the rules themselves are intimately related to the knowledge a developer has about the entire process control system. The knowledge can be extracted by interviewing skilled operators or analyzing records of system responses to prototypes of input sequences (Dubois and Prade, 1980; Bernard, 1988). In addition, important decisions need to be made about the algorithm itself, such as what kind of implication to use, the appropriate defuzzification method, and implementation-related issues such as how to store the fuzzy relation of the algorithm, how to quantize membership functions, and so on. A difficult issue in fuzzy control arises in connection with determining the stability characteristics of the system. Stability itself can be thought of as a fuzzy variable and can be included in a description, with various degrees of stability (not just *stable* or *unstable*) being considered. Generally though, stability questions are hard to answer exclusively within fuzzy linguistic descriptions (Kiszka et al., 1985; Jianqin and Laijiu, 1993).

Once an algorithm has been developed, its quality can be assessed by examining the shape of the fuzzy output. Consider the situation shown in Figure 6.17 (King and Mamdani, 1977). Here we have three different general shapes for the membership function of the fuzzy output at some particular time step. They reveal three different instances of algorithmic quality. In situation A, a well-peaked fuzzy output indicates presence of strong firing rules. In situation B, the output points to two different and opposite areas of the universe of discourse, and hence we identify the presence of some contradictory rules or groups of rules, at the same time suggesting an output toward -3 and toward $+3$. An algorithm that points its output in opposite directions at the same time needs some further refinement to remove this kind of contradiction. In situation C we have the presence of an *unsatisfactory* set of rules since there is no representative output. In general, low

[10] Often referred to as *structure* and *parameter identification*.

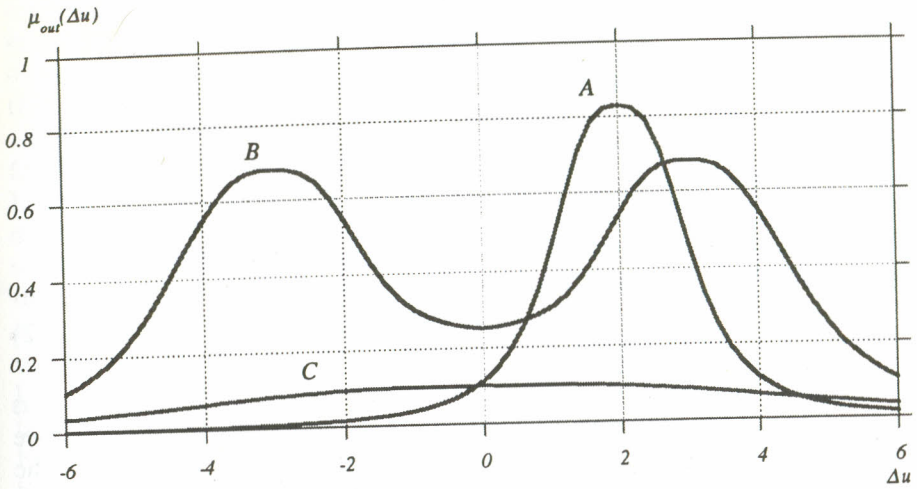


Figure 6.17 Three different cases of fuzzy output indicative of algorithmic quality: (*A*) *dominant* rule, (*B*) *contradictory* rules, and (*C*) *no satisfactory* rule.

plateaus like what is shown in situation C indicate that the knowledge encoded in the algorithm is incomplete and that additional rules are needed.

Let us now turn our attention to the various factors involved in the development of a fuzzy algorithm other than the quality of the encoded knowledge. Fuzzy algorithms are linguistic descriptions of the desirable behavior of a system. As such they have an analytical form involving fuzzy variables, relations, implication operators, and inferencing procedures. In order to examine the factors involved in algorithmic synthesis and analysis, let us look at the general analytical description of a fuzzy algorithm. Suppose that we have a control algorithm with linguistic form:

$$\begin{aligned}
&\textit{if} \quad x \textit{ is } A_1 \quad \textit{AND} \quad y \textit{ is } B_1 \quad \textit{then } u \textit{ is } C_1 \quad \textit{ELSE} \\
&\textit{if} \quad x \textit{ is } A_2 \quad \textit{AND} \quad y \textit{ is } B_2 \quad \textit{then } u \textit{ is } C_2 \quad \textit{ELSE} \\
&\cdots \\
&\textit{if} \quad x \textit{ is } A_j \quad \textit{AND} \quad y \textit{ is } B_j \quad \textit{then } u \textit{ is } C_j \quad \textit{ELSE} \\
&\cdots \\
&\textit{if} \quad x \textit{ is } A_n \quad \textit{AND} \quad y \textit{ is } B_n \quad \textit{then } u \textit{ is } C_n
\end{aligned} \tag{6.4-1}$$

At time $t = k$, crisp inputs x' and y' are given to algorithm (6.4-1), and through GMP (see Chapters 3 and 5) we determine the output membership function. Analytically the operation of inferring a fuzzy output at any given

time step may be written as

$$\mu_{C'}(u) = \phi\big(\mathrm{DOF}_1(k), \mu_{C_1}(u)\big) \vee \phi\big(\mathrm{DOF}_2(k), \mu_{C_2}(u)\big) \vee \cdots \vee \phi\big(\mathrm{DOF}_j(k), \mu_{C_j}(u)\big) \vee \cdots \vee \phi\big(\mathrm{DOF}_n(k), \mu_{C_n}(u)\big) \tag{6.4-2}$$

for implication operators ϕ where the connective *ELSE* is interpreted as *union* (see Table 5.5). For implication operators interpreted as *intersection* we change $(\vee)$ in equation (6.4-2) to $(\wedge)$. Equation (6.4-2) tells us that the collective output of the controller depends on aggregating the outputs of individual rules with the output of each rule depending, in turn, on the degree of fulfillment plus the consequent membership function of the rule.

The degree of fulfillment of the jth rule DOF_j in (6.4-2) depends on the interpretation of the connective *AND* (generally thought of as a T norm (see Appendix)). If *AND* is analytically described as min $(\wedge)$, the *degree of fulfillment* at timestep k is

$$\mathrm{DOF}_j(k) = \mu_{A_j}(x') \wedge \mu_{B_j}(y') \tag{6.4-3}$$

where x' and y' are the measured input values at a given time k. If *AND* is analytically described as *product* $(\cdot)$, the DOF is

$$\mathrm{DOF}_j(k) = \mu_{A_j}(x') \cdot \mu_{B_j}(y') \tag{6.4-4}$$

3 It should be noted that the DOF is a function of time as different input values activate the rules to different degrees at different times. Equation (6.4-2) gives the fuzzy output (before defuzzification) of the controller in a general form and helps us to identify choices the developer needs to make such as the appropriate fuzzy implication operator ϕ and the associated interpretation of the connective *ELSE*, the form of DOF, and the defuzzification method.

4 In the design of fuzzy systems it is important to adequately cover the state space of the problem. Generally the development of a rule set that is both complete and correct is one of the most difficult problems in fuzzy control. Although various approaches have been suggested for learning a control algorithm on-line and adapting it to changing process conditions (Graham and Newell, 1988; Cox, 1993), this is still a rather heuristic process, and a good understanding of the various factors influencing the output of the

controller is very helpful in its development and evaluation. Generally, which rules and to what extent will contribute toward an output at any given time depends primarily on the *form of the degree of fulfillment* (min or product), the *defuzzification method*, and the *implication operator*.

Let us consider the jth rule of (6.4-1) where triangular membership functions are used as shown in Figure 6.18. We assume min $(\wedge)$ form for DOF as in equation (6.4-3). We also assume common quantized universe of discourse for all variables, of the type shown in Table 6.1. In Figure 6.18 we see the part of the Cartesian product of LHS variables covered by the jth rule. The $x \times y$ plane is the *state space* of our system. The state space covered by the jth rule is a square of six units edge, centered at (x_{jc}, y_{jc}) as shown in the figure. At time $t = k$, crisp inputs (x', y') are given to the rule. Let us first see what happens when the point (x', y') is located within the innermost square centered at (x_{jc}, y_{jc}) that has an edge of 2 units as shown in Figure 6.18. In such a case the degree of fulfillment DOF_j of the jth rule will be the same regardless of the exact location of point (x', y') so long as it remains within this particular square, since we have that

$$\begin{aligned}
\mu_{A_j}(x_{jc} + 1) \wedge \mu_{B_j}(y_{jc} + 1) &= 0.67 \wedge 0.67 = 0.67 \\
\mu_{A_j}(x_{jc}) \wedge \mu_{B_j}(y_{jc} - 1) &= 0.67 \wedge 0.67 = 0.67 \\
\mu_{A_j}(x_{jc}) \wedge \mu_{B_j}(y_{jc} + 1) &= 0.67 \wedge 0.67 = 0.67 \\
\mu_{A_j}(x_{jc} - 1) \wedge \mu_{B_j}(y_{jc} - 1) &= 0.67 \wedge 0.67 = 0.67 \\
\mu_{A_j}(x_{jc} - 1) \wedge \mu_{B_j}(y_{jc}) &= 0.67 \wedge 0.67 = 0.67 \\
\mu_{A_j}(x_{jc} - 1) \wedge \mu_{B_j}(y_{jc} + 1) &= 0.67 \wedge 0.67 = 0.67
\end{aligned} \tag{6.4-5}$$

Thus the DOF is 0.67 everywhere within this innermost square. Similar considerations lead us to the conclusion that if the point (x', y') falls within a square of edge 4, the DOF is 0.33 everywhere, whereas if it falls outside, the DOF is 0. Thus the distribution of the DOFs of a rule centered at (x_{jc}, y_{jc}) is as shown in Table 6.3.

When at time $t = k$ the crisp inputs (x', y') are given to the controller, the DOF of individual rules depends linearly on the distance between the input and the center (or peak) of the rule (x_{jc}, y_{jc}). Obviously the number of rules that will influence and contribute to the collective fuzzy output at any given time are only those within a distance d from input (x', y') (see Figure 6.18). Thus in a control algorithm, only the part of state space a distance d from a crisp input needs to be considered for rules that may be "fired." The rest have DOF = 0. The distance d is taken to be half of the support of a fuzzy value (considering the support to be where the membership function is not trivial). As shown in Figure 6.18, the edge of a square with (x_{jc}, x_{jc}) at its

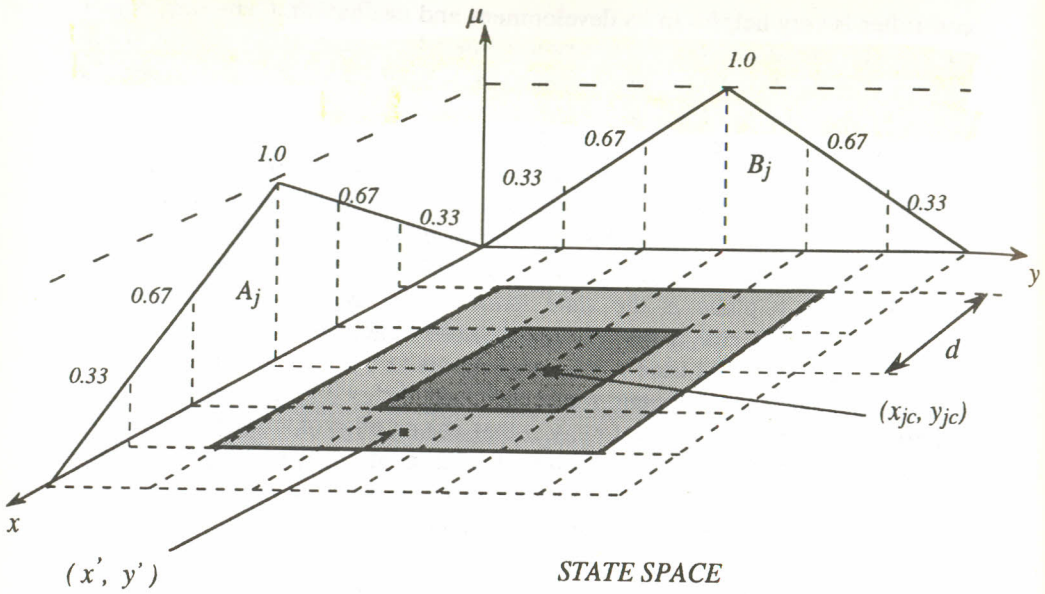


Figure 6.18 Region of influence of a rule in state space.

Table 6.3 Distribution of DOFs around rule center (rule with min DOF)

0	0	0	0	0	0	0
0	0.33	0.33	0.33	0.33	0.33	0
0	0.33	0.67	0.67	0.67	0.33	0
0	0.33	0.67	1.0	0.67	0.33	0
0	0.33	0.67	0.67	0.67	0.33	0
0	0.33	0.33	0.33	0.33	0.33	0
0	0	0	0	0	0	0

center is $2d$. We assumed of course that the two fuzzy values A_j and B_j have supports of the same length. When the support sets do not have the same length, then instead of a square we have a parallelogram and the distance of (x', y') from (x_{jc}, x_{jc}) will vary directionally as we move in different locations of state space.

The number of rules contributing to the fuzzy output depends also on the defuzzification method. When MOM is used, only the rules that are very close to the input (x', y') contribute maximum values to the output and therefore they are the only ones that need to be taken into account. We recall that with MOM, only the maximum values of the various contributions to the fuzzy output are used; and hence, only rules with high DOF and therefore small distance from (x', y') will influence the output. When COA is used, all the rules within a distance d from (x', y') need to be taken into account. Of course their contribution is in proportion to their distance from the input. Those which are the closest have the highest degree of fulfillment and therefore contribute more than those far away. Nonetheless, all rules within a distance d from (x', y') need to be taken into account.

On the other hand, if product is used in the DOF—that is, $\text{DOF}_j = \mu_{A_j}(x_k) \cdot \mu_{B_j}(y_k)$—the distribution of the DOF for input values in the vicinity of the jth rule would be as shown in Table 6.4. We see that DOF is varying with distance from (x_{jc}, y_{jc}) in a nonlinear manner. Again, if COA defuzzification is used, all the rules within a distance d need to be taken into account. When MOM is used, the rule peaking at (x_{jc}, y_{jc}) will have less influence than the earlier situation when the degree of fulfillment was defined through min. In the present case, it will influence the rule in a directional manner. For this reason with product DOF, COA defuzzification is more appropriate. Sometimes we may have very low DOFs, and therefore a cutoff number ought to be used to limit the number of rules that need to be considered. Thus we may choose $\text{DOF}_j = \alpha$ and ignore rules below whose DOF is less than α.

When continuous instead of discrete fuzzy values are used, their membership functions can be defined by various functions such as *S-shaped* and Π-*shaped* functions (see Sections 2.6.3, 2.6.4, and 2.6.5). Similar considerations hold for such cases as for discretized membership functions. With MOM, tremendous accuracy is not required since only the relative size of the membership values influences the final result and not the precise magnitude. Thus in order to take advantage of the fact that we have more precise membership values with continuous membership functions, it is best to utilize the COA (or COS) method of defuzzification.

We turn our attention now to the influence of the shape of membership functions describing the antecedent and consequent fuzzy values. The support of the fuzzy values of the antecedents (e.g., $2d$) determines the area of influence of every rule and hence plays a crucial role in the calculation of the control value. Generally the *shape* of the membership functions of LHS values has a substantial impact on the computation of the control action at any given time since it affects the DOF of each rule. Obviously the shape of

Table 6.4 Distribution of DOFs around rule center (rule with product DOF)

0	0	0	0	0	0	0
0	0.09	0.21	0.33	0.21	0.09	0
0	0.21	0.49	0.67	0.49	0.21	0
0	0.33	0.67	1.0	0.67	0.33	0
0	0.21	0.49	0.67	0.49	0.21	0
0	0.09	0.21	0.33	0.21	0.09	0
0	0	0	0	0	0	0

the RHS membership function affects directly the contribution of the rule to the overall fuzzy output.

When MOM is used, the exact shape of the LHS membership functions does not play a major role provided that it is in the general shape of a "hill, and symmetric with respect to a normal point." MOM defuzzification effectively distinguishes the rules with the highest priority (highest DOF) that is, the rules closest to the input (x', y'). Thus with MOM, the DOF suggests the distance from (x', y') and therefore the absolute values of the membership functions are not crucial, just their magnitude in relation to the membership functions of other rules. Similarly the exact shape of RHS membership functions does not play a crucial role in the calculation of the crisp output. When the support is not symmetric, the peaks in the membership function of the antecedents move relative to the support set and thus offer different DOF for the same inputs to the controller and different nonsymmetric shapes of "hills."

When COA defuzzification is used, the exact shape of the membership functions of antecedent as well as consequent plays an important role, even when symmetric membership functions are used. This happens because COA

defuzzification takes into account the area under the curve of the total fuzzy output at any given time. This area is influenced directly by the shape of the consequent membership functions of the contributing rules, and indirectly through the DOF (the shape of the consequent membership functions of the contributing rules). The above-mentioned influence on the crisp controller output is emphasized (accentuated) even more if nonsymmetric membership functions are used, as well as if different membership functions are used for the different variables.

Let us now examine the influence of fuzzy implication operator ϕ on the computations of the controller output at a time $t = k$. We consider a hypothetical case where only one rule exists in the vicinity of (x', y'); that is, only one rule fires. Suppose that we use Mamdani min implication operator ϕ_c and fuzzy sets defined through symmetric triangular functions of the form shown in Table 6.1 (also Figure 6.5). Figure 6.19 (top) shows what happens to the RHS value for different degrees of fulfillment of a rule. The consequent membership function remains the same for DOF = 1 and is gradually clipped, finally becoming zero when DOF = 0. The defuzzified output is the same with either COA or MOM methods. The situation is similar when Larsen product implication operator ϕ_p is used as can be seen in Figure 6.18.

On the other hand, if the *Boolean* implication operator ϕ_b is used, a "plateau" is created that grows, as DOF is getting smaller, until it covers the entire universe of discourse when DOF = 0. While this is exactly the opposite of what happens when the Mamdani min implication operator is used, it is counterbalanced by interpreting *ELSE* as intersection (min) when a number of rules are connected in order to compute the total fuzzy output of the controller. In fact, this is the reason for using min for the connective *ELSE* with this implication operator (see Table 5.5). It should be noted from Figure 6.19 that COA and MOM defuzzification may give different crisp outputs when Boolean implication is used.

If the *arithmetic* implication operator ϕ_a is used, a "plateau" is also formed as it happens with Boolean implication. The peak of the function is *clipped* as with Mamdani min implication. When DOF = 0, the "plateau" covers the entire universe of discourse. Again it should be noted that COA and MOM defuzzification may give different crisp outputs.

As can be seen in Figure 6.19, when Mamdani min and Larsen product implications are used, both COA and MOM defuzzification methods give similar results. In Boolean and arithmetic implications, on the other hand, the two defuzzification methods will give rather different results due to the developing plateaus. When plateaus appear, MOM defuzzification is better because COA considers the peak of the rule together with the developing plateau, and hence it shifts the final crisp output away from the location that is suggested by the peak of the rule. This is undesirable since "plateaus" do not contain useful information. They can be interpreted as a fuzzy value "*unknown*." In Figure 6.19 we note that when DOF = 0, Mamdani min and Larsen product implications give "nothing" as the output of the controller.

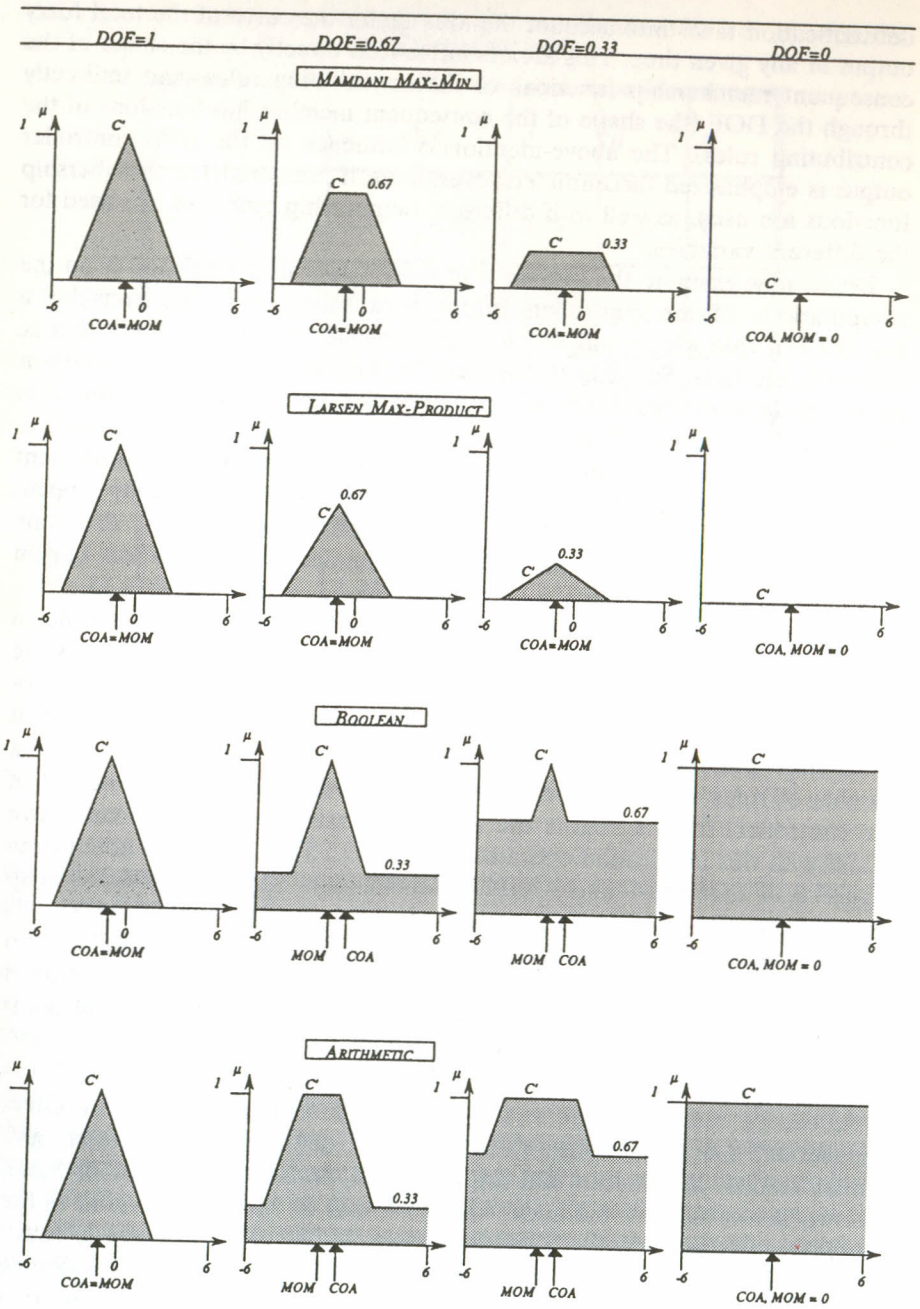


Figure 6.19 Influence on output of various implications under different degrees of fulfillment.

On the other hand, using the Boolean and arithmetic implications produces *unknown* as output. In all cases we can set the output to crisp zero.

In addition, with Mamdani min and Larsen product implications we may effectively use either method of defuzzification since the total fuzzy output contains contributions from all (or many) rules. Generally, if a rule or several rules are an equal distance away from the point (x', y'), we have the same results with either defuzzification method. With Boolean and arithmetic implications we do not need to use COA defuzzification (which is actually computationally more demanding) since in many cases the total fuzzy output does not effectively represent the contribution of the individual rules (due to the "plateau" or "flattening" effect). In general, it is preferable (but not required) to use MOM in conjunction with Boolean and arithmetic implications.

REFERENCES

Bernard, J. A., Use of a Rule-Based System for Process Control, *IEEE Control Systems Magazine*, pp. 3–13, October 1988.

Cox, E., Adaptive Fuzzy Systems, *IEEE Spectrum*, pp. 27–31, February 1993.

Driankov, D., Hellendoorn, H., and Reinfrank, M., *An Introduction to Fuzzy Control*, Springer-Verlag, Berlin, 1993.

Dubois, D., and Prade, H., *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, Boston, 1980.

Efstathiou, J., Rule-Based Process Control Using Fuzzy Logic, in *Approximate Reasoning in Intelligent Systems, Decision and Control*, E. Sanchez and L. A. Zadeh, eds., Pergamon Press, Oxford, 1987, pp. 145–158.

Fuller, R., and Zimmermann, H.-J., On Computation of the Compositional Rule of Inference under Triangular Norms, *Fuzzy Sets and Systems*, Vol. 51, pp. 267–275, 1992.

Graham, B. P., and Newell, R. B., Fuzzy Identification and Control of a Liquid Level Rig, *Fuzzy Sets and Systems*, Vol. 26, pp. 255–273, 1988.

Harris, C. J., Moore, C. G., and Brown, M., *Intelligent Control—Aspects of Fuzzy Logic and Neural Nets*, World Scientific, Singapore, 1993.

He, S-Z., Tan, S., and Xu, F-L., Fuzzy Self-Tuning of PID Controllers, *Fuzzy Sets and Systems*, Vol. 56, pp. 37–46, 1993.

Hirota, K., *Industrial Applications of Fuzzy Control*, Springer-Verlag, Tokyo, 1993.

Jager, R., *Fuzzy Logic in Control*, Unpublished Ph.D. Dissertation, University of Delft, The Netherlands, 1995.

Jang, J-S., and Gulley, N., *Fuzzy Logic Toolbox User's Guide*, Mathworks, 1995.

Jang, J-S., and Sun, C-T., Neuro-Fuzzy Modeling and Control, *Proceedings of the IEEE*, Vol. 83, No. 3, pp. 378–406, 1995.

Jiangin, C., and Laijiu, C., Study on Stability of Fuzzy Closed-Loop Control Systems, *Fuzzy Sets and Systems*, Vol. 57, pp. 159–168, 1993.

Johnson, C. D., *Process Control Instrumentation Technology*, John Wiley & Sons, New York, 1977.

King, P. J., and Mamdani, E. H., The Application of Fuzzy Control Systems to Industrial Processes, in *Fuzzy Automata and Decision Processes*, M. M. Gupta, G. N. Saridis, and B. R. Gaines, eds., North-Holland, New York, 1977, pp. 321–330.

King, R. E., and Karonis, F. C., Multi-Level Expert Control of a Large-Scale Industrial Process, in *Fuzzy Computing Theory, Hardware, and Applications*, M. M. Gupta and T. Yamakawa, eds., Elsevier/North-Holland, Amsterdam, 1988, pp. 323–340.

Kiszka, J. B., Gupta, M. M., and Nikiforuk, P. N., Some Properties of Expert Control Systems, in *Approximate Reasoning in Expert Systems*, M. M. Gupta, A. Kandel, W. Bandler, and J. B. Kiszka, eds., Elsevier/North-Holland, Amsterdam, 1985, pp. 283–306.

Larkin, L. I., A Fuzzy Logic Controller for Aircraft Flight Control, in *Industrial Applications of Fuzzy Control*, M. Sugeno, ed., North-Holland, New York, 1985, pp. 87–103.

Lee, C. C., Fuzzy Logic in Control Systems: Fuzzy Logic Controller—Part-I, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 20, No. 2, pp. 404–418, March/April 1990a.

Lee, C. C., Fuzzy Logic in Control Systems: Fuzzy Logic Controller—Part-II, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 20, No. 2, pp. 419–435, March/April 1990b.

Maeda, M., and Murakami, S., A Self-Tuning Fuzzy Controller, *Fuzzy Sets and Systems*, Vol. 51, pp. 29–40, 1992.

Mamdani, E. H., Application of Fuzzy Algorithms for Control of Simple Dynamic Plants, *Proceedings of IEE*, Vol. 121, No. 12, pp. 1585–1588, 1974.

Mamdani, E. H., Advances in the Linguistic Synthesis of Fuzzy Controllers, *International Journal of Man-Machine Studies*, Vol. 8, pp. 669–678, 1976.

Mamdani, E. H., Applications of Fuzzy Set Theory to Control Systems: A Survey, in *Fuzzy Automata and Decision Processes*, M. M. Gupta, G. N. Saridis, and B. R. Gaines, eds., North-Holland, New York, 1977, pp. 1–13.

Matia, F., Jimenez, A., Galan, R., and Sanz, R., Fuzzy Controllers: Lifting the Linear–Nonlinear Frontier, *Fuzzy Sets and Systems*, Vol. 52, pp. 113–128, 1992.

Mizumoto, M., Fuzzy Controls Under Various Reasoning Methods, *Information Sciences*, Vol. 45, pp. 129–141, 1988.

Ostergaard, J. J., and Holmblad, L. P., Control of Cement Kiln by Fuzzy Logic, in *Fuzzy Information and Decision Processes*, M. M. Gupta and E. Sanchez, eds., North-Holland, Amsterdam, 1982, pp. 389–399.

Pedrycz, W., *Fuzzy Control and Fuzzy Systems*, second (extended) edition, John Wiley & Sons, New York, 1993.

Schwartz, D. G., Fuzzy Logic Flowers in Japan, *IEEE Spectrum*, pp. 32–35, July 1992.

Shoureshi, R., and Rahmani, K., Derivation and Application of an Expert Fuzzy Optimal Control System, *Fuzzy Sets and Systems*, Vol. 49, pp. 93–101, 1992.

Sugeno, M., An Introductory Survey of Fuzzy Control, *Information Sciences*, Vol. 36, pp. 59–83, 1985.

Terano, T., Asai, K., and Sugeno, M., *Fuzzy Systems Theory and Its Applications*, Academic Press, Boston, 1992.

Tsukamoto, Y., An Approach to Fuzzy Reasoning Method, *Advances in Fuzzy Set Theory and Applications*, M. M. Gupta, R. K. Ragade, and R. R. Yager, eds., North-Holland, Amsterdam, 1979, pp. 134–149.

Tzafestas, S. G., and Papanikolopoulos, N. P., Incremental Fuzzy Expert PID Control, *IEEE Transactions on Industrial Electronics*, Vol. 37, No. 5, pp. 365–371, 1990.

Wang, Li-Xin, *Adaptive Fuzzy Systems and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1994.

Yager, R. R., and Filev, D. P., SLIDE: A Simple Adaptive Defuzzification Method, *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 1, pp. 69–78, 1993.

Yager, R. R., and Filev, D. P., *Essentials of Fuzzy Modeling and Control*, John Wiley & Sons, New York, 1994.

Yamakawa, T., Fuzzy Hardware Systems of Tomorrow, in *Approximate Reasoning in Intelligent Systems, Decision and Control*, E. Sanchez and L. A. Zadeh, eds., Pergamon Press, Oxford, 1987, pp. 1–20.

Yamakawa, T., Stabilization of an Inverted Pendulum by a High-Speed Fuzzy Logic Controller Hardware System, *Fuzzy Sets and Systems*, Vol. 32, pp. 161–180, 1989.

Yasunobu, S., and Miyamoto, S., Automatic Train Operation System by Predictive Fuzzy Control, in *Industrial Applications of Fuzzy Control*, M. Sugeno, ed., North-Holland, Amsterdam, 1985, pp. 1–18.

Zimmermann, H.-J., *Fuzzy Set Theory and Its Applications*, Kluwer-Nijhoff, Boston, 1985.

PROBLEMS

1. A fuzzy control system used inputs of error e and change in error Δe to control an output variable u. Their fuzzy membership functions have the following characteristics:

Variable	Range		Description of membership function
e Error (%)	-20 to $+20$	N (negative)	Straight line from 1 at -20% to 0 at 0%
		Z (zero)	Straight line from 0 at -20% to 1 at 0% and another straight line from 1 at 0% to 0 at $+20\%$
		P (positive)	Straight line from 0 at 0% to 1 at $+20\%$
Δe Change in	-10 to $+10$	N (negative)	Straight line from 1 at -10%/min to 0 at $+10\%$/min
error (%/min)		P (positive)	Straight line from 0 at -10%/min to 1 at $+10\%$/min
u Output (%)	-25 to $+25$	N (negative)	Straight line from 1 at -25% to 0 at 0%
		Z (zero)	Straight line from 0 at -25% to 1 at 0% and another straight line from 1 at 0% to 0 at $+25\%$
		P (positive)	Straight line from 0 at 0% to 1 at $+25\%$

The fuzzy algorithm is given below. Determine the output u for $e' = +16\%$ and $\Delta\dot{e}' = -2\%$/min using the Mamdani min implication operator and

max–min composition (as well as min interpretation for *AND* in the degree of fulfillment). Use the Center of area method to defuzzify the answer. (Sketch the various membership functions involved and show how you obtained your solution.)

FUZZY ALGORITHM

R_1 *if* e *is* N *AND* Δe *is* N *then* u *is* P *ELSE*

R_2 *if* e *is* N *AND* Δe *is* P *then* u *is* P *ELSE*

R_3 *if* e *is* Z *AND* Δe *is* N *then* u *is* Z *ELSE*

R_4 *if* e *is* Z *AND* Δe *is* P *then* u *is* Z *ELSE*

R_5 *if* e *is* P *AND* Δe *is* N *then* u *is* N *ELSE*

R_6 *if* e *is* P *AND* Δe *is* P *then* u *is* N

2. Repeat Problem 1 using the Larsen product implication, max–min composition, and product for the degree of fulfillment.

3. In Problem 1, the error starts at a value of $+16\%$ at time 0 and decreases at a rate of $2\%/\text{min}$ for 4 minutes. Determine the output u at times $t = 0, 1, 2, 3,$ and 4.

4. Analyze the fuzzy controller given in Problem 1 using the criteria given in Section 4. Are there contradictions within the rule set? Is there a dominant rule? Are the rules covering the state space in a satisfactory manner?

5. Using MATLAB, draw the control hypersurface for the fuzzy controller given in Problem 1. Simulate the controller for the range of all possible inputs and answer the questions posed in Problem 4.

6. Show what the different interpretations for *ELSE* could be for the fuzzy controller of Problem 1 and the implication operators given in Table 5.2.