5. Adaptive Fuzzy Control

5.1 Introduction

Most of the real-world processes that require automatic control are nonlinear in nature. That is, their parameter values alter as the operating point changes, over time, or both. As conventional control schemes are linear, a controller can only be tuned to give good performance at a particular operating point or for a limited period of time. The controller needs to be retuned if the operating point changes, or retuned periodically if the process changes with time. This necessity to retune has driven the need for adaptive controllers that can automatically retune themselves to match the current process characteristics. An excellent introduction to "conventional" adaptive control systems is by Åström [8].

FKBC are nonlinear and so they can be designed to cope with a certain amount of process nonlinearity. However, such design is difficult, especially if the controller must cope with nonlinearity over a significant portion of the operating range of the process. Also, the rules of the FKBC do not, in general, contain a temporal component, so they cannot cope with process changes over time. So there is a need for adaptive FKBC as well.

There is still contention as to what exactly constitutes an adaptive controller [8], and there is no consensus on the terminology to use in describing adaptive controllers. However, to enable the clear classification of the differing attempts at developing adaptive FKBC, we will introduce some definitions and terminology.

Adaptive controllers generally contain two extra components on top of the standard controller itself. The first is a "process monitor" that detects changes in the process characteristics. It is usually in one of two forms:

- a performance measure that assesses how well the controller is controlling, or
- a parameter estimator that constantly updates a model of the process.

The second component is the adaptation mechanism itself. It uses information passed to it by the process monitor to update the controller parameters and so adapts the controller to the changing process characteristics. Adaptive controllers can be classified as performance-adaptive or parameter-adaptive depending on which type of process monitor they employ [184]. Both types of process monitor will be seen in the adaptive FKBC that follow.

FKBC contain a number of sets of parameters that can be altered to modify the controller performance. These are:

- the scaling factors for each variable,
- the fuzzy set representing the meaning of linguistic values,
- the if-then rules.

A non-adaptive FKBC is one in which these parameters do not change once the controller is being used on-line. If any of these parameters are altered on-line, we will call the controller an adaptive FKBC. Each of these sets of parameters has been used as the controller parameters to be adapted in different adaptive FKBC. For the purposes of this chapter, adaptive FKBC that modify the fuzzy set definitions or the scaling factors will be called self-tuning controllers. Altering these parameters essentially fine tunes an already working controller. Adaptive FKBC that modify the rules will be called self-organizing controllers. They can either modify an existing set of rules, in which case they are similar to self-tuning controllers, or they can start with no rules at all and "learn" their control strategy as they go.

Detailed descriptions of example adaptive FKBC are given in Section 5.3. In the following section the design options available when building an adaptive FKBC will be given in terms of choosing a suitable process monitor and adaptation mechanism.

5.2 Design and Performance Evaluation

As described above, the adaptive component of an adaptive controller consists of two parts, namely,

1. the process monitor,
2. the adaptation mechanism.

The process monitor looks for changes in process characteristics and the adaptation mechanism alters the controller parameters on the basis of any detected changes. The nature of the performance monitor is not dictated by whether the underlying controller is fuzzy or not. So identical performance monitors can be used in both adaptive FKBC and non-fuzzy adaptive controllers. The adaptation mechanism, on the other hand, is specifically designed for altering the parameters of a FKBC. These two components will now be described in more detail.

5.2.1 The Performance Monitor

Parameter Estimators

Changes in process characteristics can either be detected through on-line identification of process model, or by assessment of the controlled response of the

process. A process model is a mathematical description of the process that gives values of the process-outputs, given the current process-state and the inputs to the process. The process-inputs come from the controller. An on-line identifier requires the assumption of a particular form of process model.

The most commonly used model in controller design, fuzzy or otherwise, is the single-input, single-output, linear, first-order plus dead-time model described by the transfer function

$$\frac{\bar{y}(s)}{\bar{u}(s)} = \frac{K_P e^{-t_d s}}{\tau s + 1}, \tag{5.1}$$

where $\bar{y}(s)$ is the Laplace transform of the process-output, $\bar{u}(s)$ is the Laplace transform of the process-input, K_P is the gain, t_d is the dead-time, and τ is the time constant. This type of model is used in the self-tuning regulator [8], one of the most popular non-fuzzy adaptive controllers. The process monitor must continually estimate values for the process parameters, K_P, τ and t_d. This model has also been used in adaptive FKBC [208, 89].

A discrete-time, single-input, single-output (SISO), linear model is the time-series model of the form

$$y_{t+1} = \sum_{i=0}^{n} a_i(t) y_{t-i} + \sum_{j=0}^{m} b_j(t) u_{t-j}, \tag{5.2}$$

where y_t and u_t are the process-output and input at time t, respectively. In this case the process monitor must estimate values for the parameters $a_i(t)$ and $b_j(t)$ from collections of input-output data, $\{u_k, y_k\}_{k=1}^{K}$. This type of model has also been used in an adaptive FKBC [14].

The model can also be a fuzzy model of the process. Such a model consists of a set of rules of the same form as FKBC rules, but which describe the linguistic values of the process-output for given linguistic values of the process-input and process-state variable. For a multi-input, multi-output system (MIMO), with process-state vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, $(x_j \in \mathcal{X}_j)$, process-input vector $\mathbf{u} = (u_1, u_2, \ldots, u_m)^T$, and $(u_k \in \mathcal{U}_k)$, process-output vector $\mathbf{y} = (y_1, y_2, \ldots, y_\ell)^T$, $(y_r \in \mathcal{Y}_r)$, a fuzzy model consists of rules of the form:

if (x_1 is $LX_1^{(i)}$ and ... and x_n is $LX_n^{(i)}$)
and (u_1 is $LU_1^{(i)}$ and ... and u_m is $LU_m^{(i)}$)
then (y_1 is $LY_1^{(i)}$ and ... and y_ℓ is $LY_\ell^{(i)}$),

where i designates the i-th rule, and $LX_j^{(i)}$ $(j = 1, \ldots, n)$, $LU_k^{(i)}$ $(k = 1, \ldots, m)$, and $LY_r^{(i)}$ $(r = 1, \ldots, \ell)$ are linguistic values of the process-state, process-input and process-output variables, respectively. In the fuzzy model the meaning of these linguistic values is represented by fuzzy sets $\widetilde{LX}_j^{(i)}$, $\widetilde{LU}_k^{(i)}$ and $\widetilde{LY}_r^{(i)}$. These fuzzy sets are defined on the respective universes of discourse $\mathcal{X}_j$, $\mathcal{U}_k$ and $\mathcal{Y}_r$. The meaning of such a rule is given as a fuzzy relation $\tilde{R}^{(i)}$ on $\mathcal{X} \times \mathcal{U} \times \mathcal{Y}$, where

$$\begin{aligned} \mathcal{X} &= \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \\ \mathcal{U} &= \mathcal{U}_1 \times \ldots \times \mathcal{U}_m \\ \mathcal{Y} &= \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_\ell. \end{aligned} \tag{5.3}$$

In the particular case of a Mamdani-type implication we will have that the fuzzy relation $\tilde{R}^{(i)}$ is given as

$$\tilde{R}^{(i)} = \left(\widetilde{LX}_1^{(i)} \times \ldots \times \widetilde{LX}_n^{(i)}\right) \times \left(\widetilde{LU}_1^{(i)} \times \ldots \times \widetilde{LU}_m^{(i)}\right) \times \left(\widetilde{LY}_1^{(i)} \times \ldots \times \widetilde{LY}_\ell^{(i)}\right). \tag{5.4}$$

For the whole set of rules we have

$$\tilde{R} = \bigcup_i \tilde{R}^{(i)}. \tag{5.5}$$

If we denote

$$\begin{aligned} \widetilde{L\mathbf{x}} &= \left(\widetilde{LX}_1 \times \ldots \times \widetilde{LX}_n\right) \\ \widetilde{L\mathbf{u}} &= \left(\widetilde{LU}_1 \times \ldots \times \widetilde{LU}_m\right) \\ \widetilde{L\mathbf{y}} &= \left(\widetilde{LY}_1 \times \ldots \times \widetilde{LY}_\ell\right), \end{aligned} \tag{5.6}$$

then $\tilde{R}^{(i)} = \widetilde{L\mathbf{x}} \times \widetilde{L\mathbf{u}} \times \widetilde{L\mathbf{y}}$. The *fuzzy model* of the process is given as

$$\widetilde{L\mathbf{y}} = \left(\widetilde{L\mathbf{x}} \times \widetilde{L\mathbf{u}}\right) \circ \tilde{R} = \widetilde{L\mathbf{x}} \circ \widetilde{L\mathbf{u}} \circ \tilde{R}. \tag{5.7}$$

Example 5.1 A *fuzzy model* of a gas furnace, for example, where the input is the methane feed rate, and the output is the CO_2 concentration in the outlet gases, may consist of rules such as:

> *if* the current CO_2 concentration is MEDIUM
> *and* the previous methane feedrate was LOW
> *then* the next CO_2 concentration will be JUST HIGH.

This should be contrasted with a *FKBC if-then rule* for the same process. Such a rule would be of the form:

> *if* the previous CO_2 concentration was MEDIUM
> *and* the current CO_2 concentration is JUST HIGH
> *then* change the methane feed rate with a SMALL INCREASE.

Identification of a fuzzy process model involves estimation of the fuzzy relation, $\tilde{R}$, from process input-output data. $\tilde{R}$ is also called fuzzy relation matrix or relation matrix for short. There is an extensive literature covering techniques for doing this [43, 159, 160, 161, 98, 10, 204]. A particular method is given in detail in Section 5.3.4 where an adaptive FKBC that uses on-line identification of a fuzzy process model is described [81]. Other useful references that address the wider issues of fuzzy modelling, as well as fuzzy identification, include [163, 202, 203, 216, 217].

An interesting comparison has been made between time-series models and fuzzy models. A popular basis for the comparison of model identification techniques is the gas furnace data of Box and Jenkins [28]. This data consists of 296 pairs of input-output data that relate the flow rate of methane gas into the furnace to the concentration of carbon dioxide in the outlet gases. Box and Jenkins used this data to demonstrate the development of time-series models of the form shown in (5.2). Their model, obtained by a least-squares fit of the model parameters to the data, is

$$y_t = 0.55 \cdot y_{t-1} - 0.52 \cdot u_{t-3} - 0.4 \cdot u_{t-4} - 0.51 \cdot u_{t-5} + N_t, \tag{5.8}$$

$$N_t - 1.53 \cdot N_{t-1} + 0.63 \cdot N_{t-2} = a_t, \tag{5.9}$$

where y_t is the concentration of carbon dioxide in the outlet gases and u_t is the methane flow rate, at time t. The N_t is the noise in the system, and a_t is white noise. This is a rather complicated six-parameter, third-order model. It gives a fitting factor of 0.202, where the fitting factor is calculated as the sum of the squared error between the model output and the data, divided by the number of data points. For further details on how this model is derived, the reader should consult [28].

This same data set has been used to test fuzzy identification methods [81, 159, 217, 216]. Using a simpler model structure that gives

$$y_t = f(y_{t-1}, u_{t-4}), \tag{5.10}$$

fuzzy models that are nearly as accurate as the time-series model have been developed using fuzzy identification on the gas furnace data. The models have the form

$$\widetilde{LY}_t = \widetilde{LY}_{t-1} \circ \widetilde{LU}_{t-4} \circ \tilde{R}, \tag{5.11}$$

where $\tilde{R}$ is the fuzzy relation. Models using different numbers and shapes of fuzzy sets for each linguistic variable have been tried. A particular model, that gives a fitting factor of 0.47 to the data, in which five fuzzy sets were defined to cover each universe of discourse, is shown as a rule table in Table 5.1. Reading from the table, one rule is, for example:

> *if* y_{t-1} is JUST_LOW *and* u_{t-4} is LOW *then* y_t is MEDIUM.

Adaptive controllers that use on-line identification of a process model as their performance monitor are known as parameter-adaptive controllers, since their adaptation mechanisms rely on an estimate for values of the process model parameters.

Performance Measures

The alternative type of process monitor forms an assessment of controller performance based on readily measured variables. For the regulatory control problem, where the aim is to keep a process-state variable at its specified set-point, a number of performance-related variables are of potential use. These include [198]:

Table 5.1. Rule table for fuzzy model of the gas furnace data. The table entries are y_t. (Notation: L=low, M=medium, H=high, J=just) (from Graham [81])

		y_{t-1}				
		LOW	JUST_LOW	MEDIUM	JUST_HIGH	HIGH
u_{t-4}	LOW	JL	M	M	H	H
	JUST_LOW	L	JL	M	JH	H
	MEDIUM	L	JL	M	JH	H
	JUST_HIGH	L	JL	M	JH	H
	HIGH	L	L	M	M	JH

- Overshoot
- Rise time
- Settling time
- Decay ratio
- Frequency of oscillations of the transient
- Integral of the square error
- Integral of the absolute value of the error
- Integral of the time-weighted absolute error
- Gain and phase margins.

The choice of one or more performance measures depends on the type of response the control system designer wishes to achieve. Many of these measures have been used in adaptive FKBC [13, 105, 14, 11, 230, 150, 128, 169]. The final output of the performance monitor, as seen by the adaptation mechanism, is either the actual values of the performance measures (e.g., [13]), or a performance index derived from the performance measures (e.g., [169]).

For more complex supervisory control problems, the controller performance is measured by the level of satisfaction of any number of different goals and constraints. The overall performance is derived as a suitable confluence of the goals and constraints that takes into account the relative priorities involved. A fuzzy performance measure where goals and constraints are described by performance fuzzy sets is used in the adaptive FKBC of [81]. This is based on the theory of fuzzy decision making [18]. A more detailed description of this performance measure is given in Section 5.3.4.

Adaptive controllers that use a measure of controller performance as their performance monitor are known as performance-adaptive controllers.

5.2.2 The Adaptation Mechanism

The adaptation mechanism must modify the controller parameters to improve the controller performance on the basis of the output from the process monitor. Adaptation mechanisms for FKBC can be classified according to which parameters are adjusted. Parameters that can be adjusted include the scaling factors with which controller input and output values are mapped onto the universe of discourse of the fuzzy set definitions. The set definitions will often be defined on a normalized universe from, say, −1.0 to +1.0, as illustrated in Fig. 5.1.

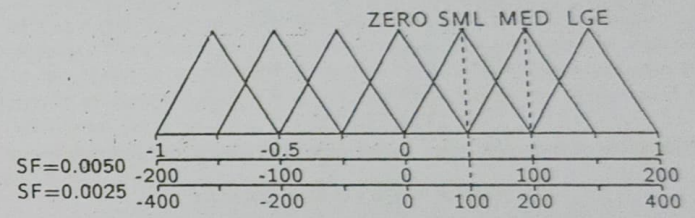


Fig. 5.1. The effect of altering a scaling factor.

The real values of the input variables may actually range from, say, −200 to +200, and so need to be scaled. If, for example, the input value is multiplied by a scaling factor of 0.005. the input is mapped to the universe of discourse as shown by the middle scale in Fig. 5.1. In this case an input value of 100 is classified as MEDIUM. Altering the scaling factor changes the classification of an input value. For example, with a scaling factor of 0.0025 a value of 100 is now classified as SMALL, as shown by the bottom scale of Fig. 5.1. This reduces the sensitivity of the controller to the input, and so reduces the controller gain. In this way, altering scaling factors is similar to gain tuning in standard PID-controllers.

Another gain tuning mechanism is to alter the shapes of the fuzzy sets. An example where the sets are altered to increase the sensitivity of the controller to small values of the input is shown in Fig. 5.2.

Whereas altering the scaling factors alters the gain uniformly across the entire input universe, changing the shapes of specific fuzzy sets allows the gain to be modified within specific regions of the universe. Adaptive controllers that adjust the fuzzy set definitions or scaling factors are called self-tuning controllers.

A third set of parameters that can be altered are the if-then rules themselves. For example, a rule may be changed from

if temperature is HIGH *then* SMALL INCREASE in cooling

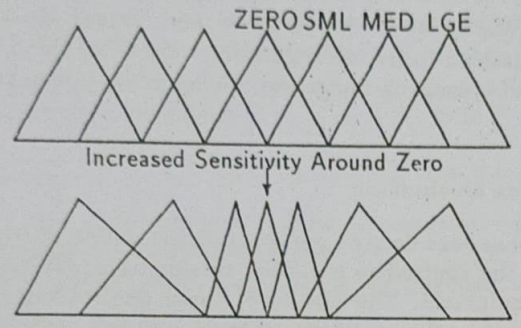


Fig. 5.2. Adapting fuzzy set definitions.

to

if temperature is HIGH *then* LARGE INCREASE in cooling

to adapt to changing operating conditions. Adaptive controllers that alter the rules are called self-organizing controllers.

Altering Scaling Factors

Quite simple schemes for altering the scaling factors to meet various performance criteria can be devised. An example is the work of Yamashita et al. [230]. Their control problem was temperature control during start-up of a packed-bed catalytic reactor. The temperature is controlled by altering the flow rate of hydrogen gas to the reactor. This process is highly exothermic and nonlinear, making it a difficult control problem. During startup the catalyst temperature must be increased to its operating point by increasing the hydrogen gas flow rate. The temperature increases slowly at first, and then rises abruptly with the onset of ignition. In order to prevent overshoot and oscillations in the temperature, the controller gain must be kept low during this period, so that only small changes in gas flow rate are made for large changes in temperature. However, if this low gain is retained once the operating temperature is reached, small, low frequency oscillations in the temperature result, due to the low sensitivity of the controller to fluctuations in the temperature. The controller gain consequently needs to be increased. Yamashita et al. designed a Mamdani FKBC with the error and change-of-error of the temperature as the inputs, and the change in hydrogen gas flow rate to the reactor as the output. They used the following scheme to automatically increase the controller gain once the operating temperature was reached by altering the scaling factors for the error and change-of-error.

The performance measure is the average of the squared error over the previous three sampling times. At sample time, k, a scaling factor modifier, C_k, is calculated as a function of the performance measure, P_k, according to the set of linguistic rules:

if P_k is VERY LARGE *then* C_k is VERY SMALL
if P_k is LARGE *then* C_k is SMALL
if P_k is MEDIUM *then* C_k is MEDIUM
if P_k is SMALL *then* C_k is LARGE.

The scaling factors for the error (GE) and change-of-error (GΔE) are then calculated via

$$\mathrm{GE}_k = C_k \, \mathrm{GE}_0 \tag{5.12}$$

$$\mathrm{G}\Delta\mathrm{E}_k = C_k \, \mathrm{G}\Delta\mathrm{E}_0, \tag{5.13}$$

where GE_0 and $\mathrm{G}\Delta\mathrm{E}_0$ are fixed initial values.

These rules for C_k can be implemented in a fuzzy way, but Yamashita et al. used a set of crisp relations that specified a specific crisp value of C_k for different ranges of P_k. The rules have the effect of increasing the controller gain by increasing the scaling factors, as the average squared error decreases as the process is maintained around its set-point. Results with an experimental reactor showed the expected improvement in control with this simple adaptive scheme.

Various schemes for altering the scaling factors on the basis of process models have also been devised [208, 89]. For example, Hayashi [89] has derived a set of equations for calculating the input and output scaling factors for a PI-like FKBC from the parameters of the first-order plus dead-time model of the process given by equation (5.1).

Modifying Fuzzy Set Definitions

A similar tuning mechanism is to alter the shapes of the fuzzy sets defining the meaning of linguistic values. There has been some argument [131, 215] that changing the fuzzy set definitions should not be used to tune the controller. The fuzzy set definitions are not arbitrary but are chosen to reflect the meaning of the linguistic values taken by the variables. While this is certainly true for the broad shapes of the sets, small modifications can still be made without endangering the underlying linguistic meaning. Work has shown [150, 128, 77, 13] that this can be an effective means of tuning a FKBC.

Recent work has centred on the use of mathematical optimization techniques to alter the set definitions so that the output from the FKBC matches a suitable set of reference data as closely as possible [150, 128]. This procedure is carried out off-line and so tunes the controller before it is used. No subsequent on-line adaptation is performed, so the controllers are not strictly adaptive FKBC. However, the technique is closely allied to the adaptive methods discussed in

this chapter, and it has been demonstrated that it can be used on-line [77]. The basic tuning method is detailed in Section 5.3.1.

A truly adaptive FKBC that modifies the set definitions on-line has been developed by Bartolini et al. [13]. The controller is similar to [230] in that a simple algorithm has been devised to alter the set definitions in response to a number of different performance measures, such as the average squared error. For example, if the average error is consistently below set-point, the degree to which values of the error are recognized as "negative" is incremented, and the corresponding degrees for "OK" and "positive" are decremented. This has the effect of increasing the controller's ability to detect "negative" values of the error. Bartolini et al. applied this adaptive controller to the control of a simulation of a continuous casting plant. A detailed description is given in Section 5.3.2.

Self-Organizing Controllers

A number of schemes for self-organizing FKBC have been developed. The original attempt is that of Mamdani and his coworkers [130, 132, 169, 231]. Their idea is to try to identify which rule is responsible for the current poor control performance, and then to replace it with a better rule. This requires both a mechanism for determining which rule is incorrect, and one for determining an improved rule to take its place. The performance monitor assesses the controller performance on the basis of the error and change-of-error of the process-output variables compared with what is desirable from a regulatory control perspective. The final performance monitor output is an indication of the change required in the process-output variable to achieve good control. For example, if a variable is well above set-point and moving further away from set-point, then its performance is very poor and a large change is required. The adaptation mechanism must translate the required changes in the process-output variables into required changes in the process-input variables (control-outputs). This can be done using a simple incremental model of the process that relates changes in process-inputs to changes in process-outputs (how such a model can be obtained is described in Section 5.3.3). By inverting such a model, so that it gives the changes in process-inputs required to achieve given changes in process-outputs, a new control rule can be formulated. Some knowledge of the process order and dead times is used to identify which past control-outputs are responsible for the current poor performance, and so should be corrected. This controller can modify a predefined set of rules, or it can start with no rules at all and "learn" its control policy as it goes.

A model-based self-organizing FKBC has been developed by Graham and Newell [81, 82, 83]. It is a parameter-adaptive controller whose performance monitor performs on-line identification of a fuzzy process model. There is no explicit adaptation mechanism as the model-based controller uses the fuzzy process model directly in calculating the appropriate control-output. The performance of the controller is adapted to the process as the on-line identification modifies the model to make it a more accurate representation of the process.

The controller can start with no model at all and "learn" its control policy on-line in a similar fashion to the self-organizing controller (SOC) of Mamdani.

Both of these self-organizing controllers are described in more detail in Section 5.3.

5.2.3 Performance Evaluation

Little serious work has been done on performance evaluation of adaptive FKBC. All application of the controllers has been to simulations of processes, or to laboratory-scale equipment. The results of these applications, however, are encouraging. Every adaptive controller showed improvement in performance over a non-adaptive version of the same controller, or over a conventional PID-controller, when comparisons were made. The adaptive FKBC also performed well compared with "conventional" adaptive controllers. The adaptive fuzzy controller for a catalytic reactor [230] outperformed a model-reference adaptive controller (MRAC). The FKBC of [77] learned a satisfactory control scheme faster than a comparable neural network-based controller. Finally, the performance-adaptive controller of [13] did as well as a self-tuning regulator (STR) in the control of a continuous casting plant.

Since few results have been obtained for the stability of non-adaptive FKBC, especially when they are nonlinear, it is not surprising that the stability of adaptive FKBC is still basically an open question. Procyk and Mamdani [169] carried out an empirical study of the effect of various design parameters on the performance of their self-organizing controller. These results are discussed in Section 5.3.3.

Batur and Kasparian [14] demonstrate that the circle stability criterion [173] can be used to establish the stability of their adaptive FKBC, provided that the process being controlled is linear and bounds are placed on the maximum and minimum change in control-output that can be made at each sampling time.

Further results, in the near future, are likely to be of an empirical nature. Application of these controllers to pilot-scale plants is needed to give a good evaluation of their potential worth.

5.3 The Main Approaches to Design

In the following paragraphs some of the main approaches to the design of adaptive FKBC are described in detail. The examples chosen cover the full range of different types of adaptive controllers, as described by the classifications given earlier.

5.3.1 Membership Function Tuning using Gradient Descent

This method relies on having a set of training data against which the controller is tuned. If a reliable set of controller input-output data is available, it is possible

to tune the membership functions used in the FKBC rules using a numerical optimization procedure. The basic example of this is given by Nomura et al. [150] where they use gradient descent to tune simple membership functions.

The FKBC from Nomura et al.

Their FKBC consists of a set of n if-then rules of the form

$$\text{Rule } i\text{: } \textit{if } x_1 \text{ is } LX_1^{(i)} \text{ and } \ldots \text{ and } X_m \text{ is } LX_m^{(i)} \textit{ then } u \text{ is } LU^{(i)}, \tag{5.14}$$

where $x_1, x_2, \ldots, x_m$ are the controller inputs (process-state variables), u is the control-output variable, i is the rule number, $LX_1^{(i)}, \ldots, LX_m^{(i)}$ are the linguistic values of the rule-antecedent, and $LU^{(i)}$ is the linguistic value of the rule-consequent.

The membership functions, $\widetilde{LX}_j^{(i)}$, of the antecedent part are simply triangles described by a peak value, a_{ij}, and a support, b_{ij}, in the universe of discourse, as shown in Fig. 5.3. The membership function is thus given by

$$\widetilde{LX}_j^{(i)}(x) = 1 - \frac{2|x - a_{ij}|}{b_{ij}} \tag{5.15}$$

where x is a crisp controller input in the domain $\mathcal{X}_j$ of the input variable x_j.

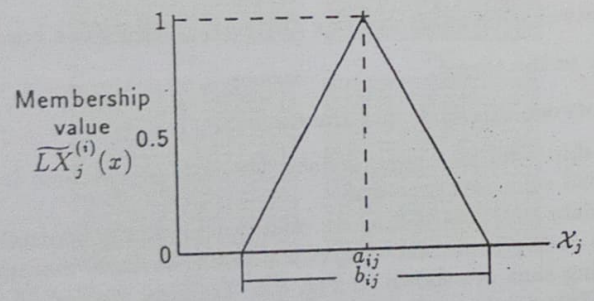


Fig. 5.3. Triangular fuzzy set (redrawn from Nomura *et al.* [150]).

The control-output membership function, $\widetilde{LU}^{(i)}$, is a fuzzy singleton set defined on the real number, u_i, i.e., $\widetilde{LU}^{(i)} = 1/u_i$.

Using the max-product composition (see Section 2.2.4) and the Center-of-Area defuzzification method, the non-fuzzy output u^* from the rule set is given by

$$u^* = \frac{\sum_{i=1}^{n} \mu_i \cdot u_i}{\sum_{i=1}^{n} \mu_i}, \tag{5.16}$$

where μ_i is given by the product of the membership degrees of the input variables in the i-th rule

$$\mu_i = \widetilde{LX}_1^{(i)}(x_1) \cdot \widetilde{LX}_2^{(i)}(x_2) \cdot \ldots \cdot \widetilde{LX}_m^{(i)}(x_m), \tag{5.17}$$

where x_j is a crisp controller input value from the domain $\mathcal{X}_j$ of the input variable X_j.

The Gradient Descent Algorithm

If a set of operating data is available that describes the desirable control-output, u^r, for various values of the process-state, $x_1^r, \ldots, x_m^r$, the fuzzy controller can be optimized by minimizing some criterion on the error between the FKBC output given by (5.16) and the desired output given by the reference data. By substitution of (5.15) and (5.17) into (5.16), we have an equation for the control-output, u, in terms of the membership function parameters a_{ij}, b_{ij} and u_i, $i = 1, \ldots, n$; $j = 1, \ldots, m$. These are the parameters to be tuned by the optimization procedure.

Nomura et al. have chosen to minimize the objective function, E, given by

$$E = \frac{1}{2}(u - u^r)^2, \tag{5.18}$$

where u^r is the desired real-valued control-output as given by the reference data, and u is the FKBC output, for a particular process-state.

One of the simplest methods for solving this optimization problem is the steepest descent algorithm (see, for example [180]). This is an iterative algorithm that seeks to decrease the value of the objective function with each iteration. It relies on the fact that from any point the objective function decreases most rapidly in the direction of the negative gradient vector of its parameters at that point. If we have $E(Z)$, where $Z = (z_1, z_2, \ldots, z_p)$, then this vector is

$$\left(-\frac{\partial E}{\partial z_1}, -\frac{\partial E}{\partial z_2}, \ldots, -\frac{\partial E}{\partial z_p}\right).$$

If $z_i(t)$ is the value of the i-th parameter at iteration t, the steepest descent algorithm seeks to decrease the value of the objective function by modifying the parameter values via

$$z_i(t+1) = z_i(t) - K\frac{\partial E(Z)}{\partial z_i}, \quad i = 1, \ldots, p, \tag{5.19}$$

where K is a constant which controls how much the parameters are altered at each iteration (choosing a suitable K can be difficult – consult a textbook on minimization techniques, such as [180], for details). As the iterations proceed, the objective function converges to a local minimum.

In this case, the objective function parameters we wish to alter are the membership function parameters a_{ij}, b_{ij} and u_i, i.e.,

$$(z_1, \ldots, z_p) = (a_{11}, \ldots, a_{nm}, b_{11}, \ldots, b_{nm}, w_1, \ldots, w_n), \quad p = 2nm + n. \quad (5.20)$$

Substituting (5.17) and (5.16) into (5.18) gives the objective function in terms of the membership functions,

$$E = \frac{1}{2} \cdot \left(\frac{\sum_{i=1}^{n} (\prod_{j=1}^{m} \widetilde{LX}_j^{(i)}(x_j^r)) \cdot u_i}{\sum_{i=1}^{n} (\prod_{j=1}^{m} \widetilde{LX}_j^{(i)}(x_j^r))} - u^r \right)^2. \quad (5.21)$$

The steepest descent algorithm gives the following iterative equations for the parameter values,

$$a_{ij}(t+1) = a_{ij}(t) - K_a \cdot \frac{\partial E}{\partial a_{ij}}, \quad i = 1, \ldots, n;\ j = 1, \ldots, m, \quad (5.22)$$

$$b_{ij}(t+1) = b_{ij}(t) - K_b \cdot \frac{\partial E}{\partial b_{ij}}, \quad i = 1, \ldots, n;\ j = 1, \ldots, m, \quad (5.23)$$

$$u_i(t+1) = u_i(t) - K_u \cdot \frac{\partial E}{\partial u_i}, \quad i = 1, \ldots, n. \quad (5.24)$$

Taking the partial derivatives of E yields the following equations,

$$a_{ij}(t+1) = a_{ij}(t) - \frac{K_a \cdot \mu_i}{\sum_{k=1}^{n} \mu_k} \cdot (u - u^r) \cdot (u_i(t) - y) \cdot \operatorname{sgn}(x_j^r - a_{ij}(t)) \cdot \frac{2}{b_{ij}(t) \cdot \widetilde{LX}_j^{(i)}(x_j^r)} \quad (5.25)$$

$$b_{ij}(t+1) = b_{ij}(t) - \frac{K_b \cdot \mu_i}{\sum_{k=1}^{n} \mu_k} \cdot (u - u^r) \cdot (u_i(t) - y) \cdot \frac{(1 - \widetilde{LX}_j^{(i)}(x_j^r))}{\widetilde{LX}_j^{(i)}(x_j^r)} \cdot \frac{1}{b_{ij}(t)} \quad (5.26)$$

$$u_i(t+1) = u_i(t) - \frac{K_u \cdot \mu_i}{\sum_{k=1}^{n} \mu_k} \cdot (u - u^r). \quad (5.27)$$

The Tuning Procedure

The FKBC rules are extracted from the process operators. The membership functions are defined initially such that the input domains are divided equally, by a suitable choice of the a_{ij}, and the sets overlap, by a suitable choice of the b_{ij} (for an example see the top half of Fig. 5.2). The u_i are chosen to give a suitable range of control-outputs covering, for example, a large decrease, small decrease, no change, small increase, and a large increase.

Once a set of reliable controller input/output data $(x_1^r, \ldots, x_m^r, u^r)$ has been collected, a possible optimization procedure is as follows:

1. the rules are fired on the input data $(x_1^r, \ldots, x_m^r)$ to obtain the antecedent value, μ_i, for each rule, and the real-valued control-output, u,
2. parameters u_i are updated using (5.27),
3. rule firing is repeated using the new values of u_i,

4. parameters a_{ij} and b_{ij} are updated by (5.25) and (5.26), using the new values of u_i, μ_i and u,
5. inference error $D(t) = 1/2 \cdot (u(t) - u^r)^2$ is calculated,
6. if the change-of-error $|D(t) - D(t-1)|$ is suitably small, the optimization is complete; otherwise it is repeated from step 1.

This procedure will modify the actual values, u_i, used for the controller outputs, and will change the centre, a_{ij}, and width, b_{ij}, of the antecedent fuzzy sets. The new fuzzy sets may look something like those seen in the bottom half of Fig. 5.2.

Example 5.2 Nomura et al. have applied this to several simulated systems, including the problem of a mobile robot avoiding a moving obstacle, with some success. The robot problem was for the robot to move from its starting point to a fixed target point while avoiding a single obstacle moving across its path. Only the steering angle of the robot was controlled, while its driving speed was constant. The dynamics of the robot were not considered. The robot received four inputs:

1. distance to the obstacle,
2. angle between the robot and the obstacle, from a fixed baseline,
3. distance to the target,
4. angle between the robot and the target.

Five membership functions were defined for each of the four inputs, and an initial set of 625 rules was provided.

Training data was obtained by an operator manually controlling the robot steering angle during trials involving two different obstacle movement patterns. Typical training runs are shown in Fig. 5.4. Training yielded 66 sets of input-output data. This data was then used to tune the membership functions, using the algorithm given above. Once trained, the robot was trialled on automatically avoiding the obstacle. Results are shown in Fig. 5.5. In panels (a) and (b), the robot is seen avoiding the obstacle for the same obstacle trajectories as used in the training runs. The automatic robot tracks are smoother than the operator controlled tracks. In the remaining panels, the robot is shown avoiding the obstacle moving along trajectories for which the robot was not trained. The robot still manages to avoid the obstacle while travelling along smooth trajectories.

Alternative Methods

The algorithm described is among the simplest for this type of self-tuning of membership functions, but it serves to illustrate the ideas. Alternative algorithms could include more sophisticated optimization methods and more general membership functions. Work along these lines is already being done, for

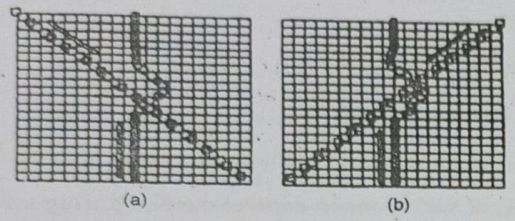


Fig. 5.4. Robot tracks taught by operator (from Nomura et al. [150]).

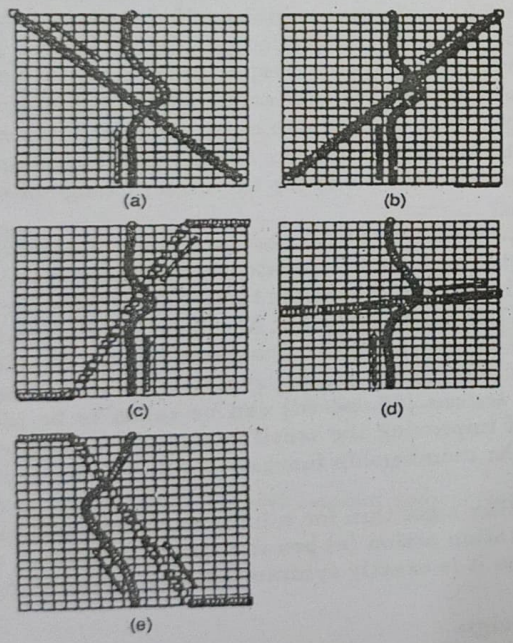


Fig. 5.5. Robot tracks controlled by tuned rules (from Nomura et al. [150]).

example by Maeda et al. [128]. Their Supervised Learning Algorithm is, they claim, some 40 times faster than the gradient descent method.

On-line Gradient Descent

The gradient descent tuning algorithm can be used on-line to form an adaptive FKBC, if suitable reference data can be generated. An architecture that achieves this has been developed by Glorennec [77]. It is illustrated in Fig. 5.6.

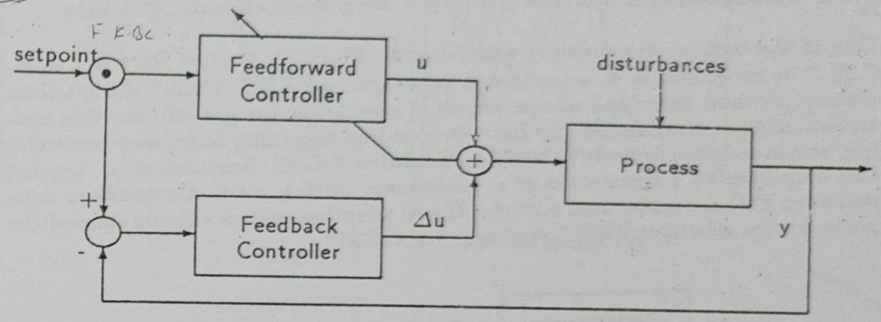


Fig. 5.6. Adaptive feedforward/feedback fuzzy controller (redrawn from Glorennec [77]).

The FKBC to be tuned is a feedforward controller. Given the required set-point, it calculates the required process input using an inverse fuzzy process model. Instead of specifying the desired control-output for a given process state, the inverse fuzzy process model predicts the process-output to be expected in time due to the current and previous process-states and control-outputs. In the absence of unmeasured disturbances, the quality of its control is dependent on the accuracy of the inverse fuzzy process model it employs. If the process-input it calculates is wrong, errors will result, and the PI-like FKBC will come into play to drive the process to set-point. As it does so, the corrections it makes to the process input, Δu, can be used as the error data, $u - u^r$, for the steepest descent tuning algorithm, to tune the feedforward controller's membership functions. In this way the feedforward controller will learn the correct process-input for a particular set-point. Glorennec chooses to only tune the rule-consequent membership functions, via equation (5.27). In principal, the full algorithm could be used to tune all the membership functions.

Glorennec has applied this scheme to a simulation of a mixer tap. The problem is to control the temperature and flow rate of the output water stream by manipulating the cold and hot water openings. He initially generated a set of process data by measuring the temperature and flow rate that resulted from various hot and cold water openings. This data was then used to tune the

feedforward controller off-line. The resultant controller performed well. The on-line architecture was then used on the same problem, with all the consequent membership functions of the feedforward controller set to zero. This meant that initially the feedforward controller had no control knowledge. The final tuned feedforward controller performed identically to the controller tuned off-line.

5.3.2 Membership Function Tuning Using Performance Criteria

One of the earliest examples of a self-tuning FKBC is that of Bartolini et al. [13]. It is an example of a performance adaptive FKBC. Unlike the gradient descent method described above, which is essentially for use off-line, this controller modifies its membership functions on-line according to various controller performance criteria. So it is truly an adaptive FKBC. Bartolini et al. applied this controller to a simulation of a continuous casting plant. Comparison with standard PID-controller and a conventional adaptive control scheme showed the performance adaptive FKBC worked very well.

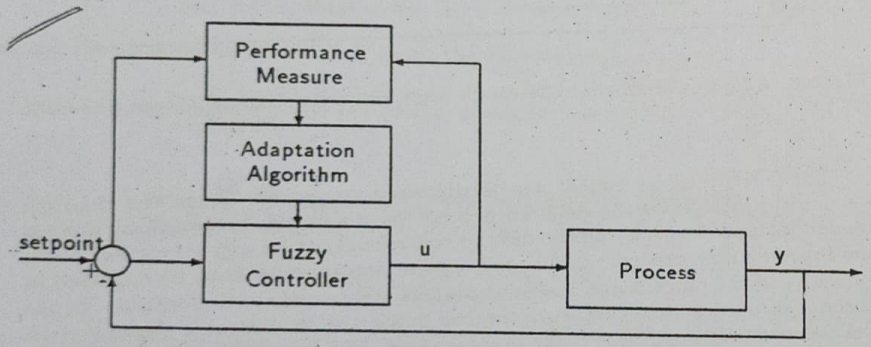


Fig. 5.7. Performance adaptive fuzzy controller.

The Controller

The structure of the controller is shown in Fig. 5.7. Only the single-input, single-output (SISO) control problem is considered. The aim is to maintain a single process-state variable at set-point. The controller is a PD-like FKBC, with the inputs being the error and change-of-error, and its output being the required change in the manipulated (control) variable. The performance monitor uses a set of six performance criteria to assess the FKBC while the controller is operating on-line. The indices are:

1. average square error, $\bar{e}^2$,

2. average error, $\bar{e}$,
3. average absolute error, $|\bar{e}|$,
4. maximum absolute error, $|e|_{max}$,
5. number, n_1, of consecutive variations in control-output,
6. number, n_2, of variations in control-output during a certain time interval.

The indices are evaluated over a fixed observation period, the length of which is a tuning parameter for the controller. As the primary aim of the controller is to maintain the process at set-point, the first four performance indices are of major importance. The last two indices express the secondary control objective of reducing the number of command variations.

The adaptation algorithm concerned with improving the set-point control is shown in Fig. 5.8. Four different adaptation actions are taken depending on the values of the first four performance indices. If the average error is too large, then adaptation action (a) or (b) is carried out depending on whether the average error is below or above set-point, respectively. Adaptation action (c) is taken if the average squared error is too large, indicating imprecise control. Finally, adaptation action (d) is taken if the error becomes too large, even for a single sampling instant. All the adaptation actions result in changes to the fuzzy set membership function definitions for the error, change-of-error, and change in control-output.

The adaptation action (a) is concerned with improving the controller performance when the process is consistently below set-point. This can be done, for instance, by increasing the degree to which values of the error are recognized as negative, i.e., below the set-point. Modification of the membership functions for the error (E) to achieve this is illustrated in Fig. 5.9. The modifications due to adaptation action (b) are exactly symmetrical with those of (a).

Adaptation actions (c) and (d) can be taken to be identical as they have the basic aim of improving the sensitivity of the controller. This can be done by sharpening the membership function definitions for the error, as illustrated in Fig. 5.10.

The adaptation algorithm for minimizing command variations is shown in Fig. 5.11. Adaptation action (e) has the aim of decreasing the sensitivity of the controller, and so it is exactly symmetrial with adaptation actions (c) and (d).

Controller Design

These adaptation algorithms contain a large number of parameters whose values must be chosen by the designer. Currently there are no specific selection criteria for these parameters. However, the following qualitative observations can be made. The aim of the adaptation process is to provide quick controller adaptation without causing instability or oscillations. In general, only small changes should be made to the fuzzy set definitions at any one time. In addition, the adaptation procedure need not be carried out at every sampling time. Bartolini

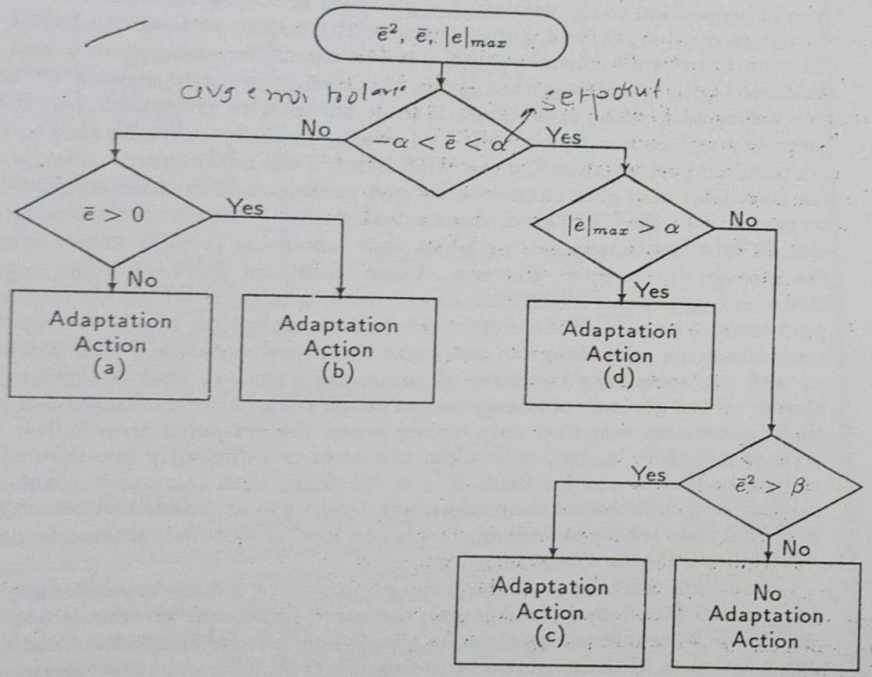


Fig. 5.8. Adaptation algorithm to improve setpoint control (redrawn from Bartolini et al. [13]).

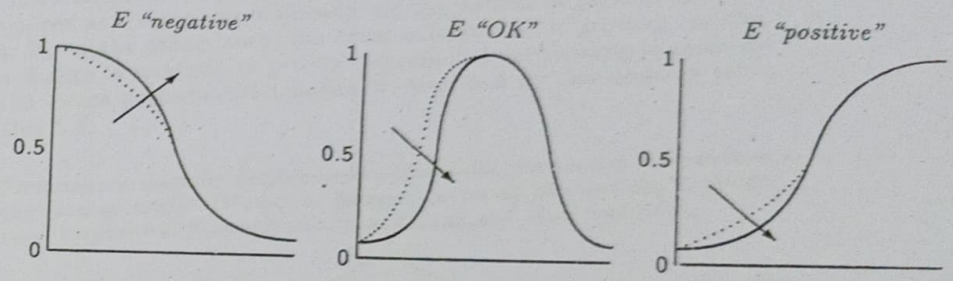


Fig. 5.9. Adaptation action (a).

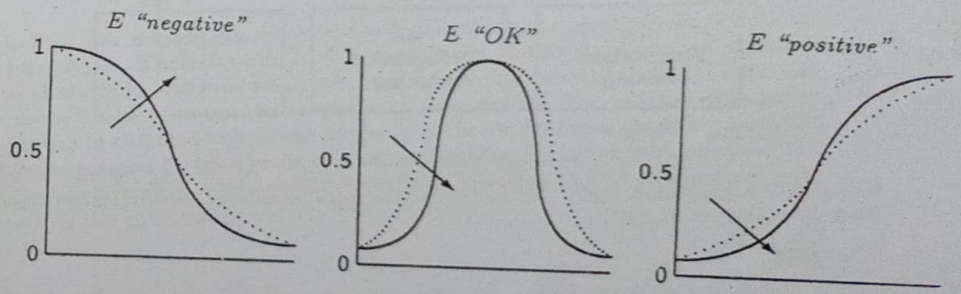


Fig. 5.10. Adaptation actions (c) and (d).

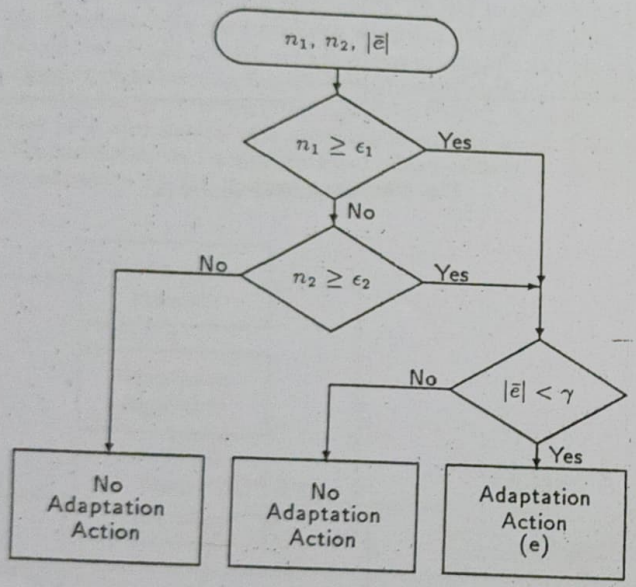


Fig. 5.11. Adaptation algorithm to reduce the number of control variations (redrawn from Bartolini et al. [13]).

et al. chose to monitor for excessive command variations are every sampling time so that this could be quickly corrected, but monitored set-point control performance at a slower rate. Adapting for set-point errors too aggressively could lead to instability.

The adaptation is done by modifying the shapes of the membership functions in proportion to the undesired effects that are being corrected. The threshold values α, β, ϵ_1, ϵ_2, and γ must be carefully chosen as they can have a large effect on the overall control system performance. The parameters α and β are concerned with specifying what levels of the set-point error criteria are acceptable before adaptation is required. If these parameters are set too low, then the controller will be constantly trying to adapt, and may never be able to satisfy the required performance. On the other hand, if the parameters are set too high, the controller may give unnecessarily bad performance as little adaptation will be carried out. Bartolini et al. demonstrated with a computer simulation of the control of a continuous casting plant that increasing β leads to an increase in the average square error. However, it also leads to a decrease in the number of command variations. Thus, choosing β involves a trade-off between competing performance criteria. The same considerations apply to α, and to the parameters controlling the adaptation to minimize command variations. The parameters, n_1 and n_2, specifying the level of command variation that is intolerable are chosen on the grounds of energy conservation requirements. Adaptation to minimize command variation only occurs when the set-point error is less than a level specified by γ, i.e., only when the error is sufficiently low that excessive control-output is not justified. If γ is too high, then too much adaptation to minimize control-output may take place, leading to an insensitive controller that provides poor set-point control. If γ is too low, then no adaptation to minimize control-output may ever take place.

The time window over which the performance indices are calculated is also important. The longer the interval, the more significant are the values of $\bar{e}$, $\bar{e}^2$, and $|e|$, but the more sluggish is the adaptation system's response. Consideration must be taken of the noisiness of the signals from which the error measurements are being made when deciding on an appropriate time window. The noisier the signals, the longer the time window should be to filter out the effects of the noise.

In summary, the choice of parameter values is process specific and requires careful matching with the controller performance objectives.

5.3.3 The Self-Organizing Controller

The major work on self-organizing controllers is that of Mamdani and his colleagues [130, 132, 169]. A detailed description of their self-organizing controller (SOC) is given in [169].

Like the controller of Bartolini et al., the self-organizing controller is also a performance-adaptive FKBC. However, in this case it is the rules themselves that are adapted, not the fuzzy set definitions, or the scaling factors. The struc-

ture of the controller is similar, but this time it also includes a model of the process. A block diagram is given in Fig. 5.12. The controller is of the standard double-input, single-output type, with the error (e) and change in error (Δe) as inputs, and process-input or control-output (u) as the output. The universes of discourse are discrete and normalized, with the appropriate scaling factors (GE, GΔE and GU, respectively) being chosen manually, i.e., they are not adapted as part of the adaptation mechanism.

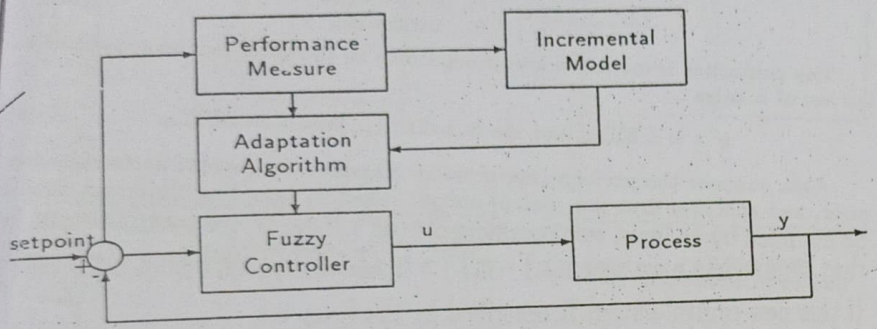


Fig. 5.12. Self-organising fuzzy controller.

Performance Monitor

The performance monitor consists of a performance measure that specifies the basic regulatory control requirements, namely a sufficiently fast approach to set-point, good damping when close to the set-point and a measure of tolerance around the set-point. The requirements with respect to this performance measure can be expressed as a set of rules that define the performance of the system at each sampling time, given the values of the error and change-of-error. For example:

if e is VERY LARGE *and* Δe is NEGATIVE VERY LARGE
then P is SATISFACTORY
if e is VERY LARGE *and* Δe is VERY LARGE
then P is VERY POOR,

where P is the performance measure.

For use in the adaptive controller, the output of the performance monitor is not given as a value of the performance, but as a value for the correction required at the process-output to obtain good performance. For example, the rules given above can be rewritten as rules describing the changes required to achieve good performance:

if e is VERY LARGE *and* Δe is NEGATIVE VERY LARGE
then C is ZERO
if e is VERY LARGE *and* Δe is VERY LARGE
then C is NEGATIVE VERY LARGE,

where C is the required correction. In other words, if the process output is a long way from set-point, but is moving rapidly towards set-point, then no change is required as the process is already moving as fast as possible towards its set-point. If, on the other hand, the error is large and is growing rapidly, then a large change is required to get the process at least moving towards its set-point. The entire performance monitor is described by the decision table shown in Table 5.2.

Table 5.2. Performance monitor decision table. The table entries are the required changes in the process-output. (Notation: Z=zero, S=small, M=medium, L=large, J=just, V=very, P=positive, N=negative) (from Procyk and Mamdani [169])

		Change in error (Δe)												
		Towards set-point							Away from set-point					
		NVL	NL	NJL	NM	NJM	NS	Z	S	JM	M	JL	L	VL
Error (e)	NVL	Z	Z	Z	Z	Z	Z	VL	VL	VL	VL	VL	VL	VL
	NL	Z	Z	Z	JM	JM	M	VL	VL	VL	VL	VL	VL	VL
	NJL	Z	Z	Z	JM	JL	L	VL	VL	VL	VL	VL	VL	VL
	NM	Z	Z	Z	JM	JM	M	JL	JL	JL	JL	L	L	VL
	NJM	Z	Z	Z	Z	Z	Z	JM	JM	JM	M	JL	L	VL
	NS	Z	Z	Z	Z	Z	Z	S	S	S	JM	M	JL	L
	NZ	Z	Z	Z	Z	Z	Z	Z	Z	Z	S	JM	M	JL
	PZ	Z	Z	Z	Z	Z	Z	Z	Z	Z	NS	NJM	NM	NJL
	S	Z	Z	Z	Z	Z	Z	NS	NS	NS	NJM	NM	NJL	NL
	JM	Z	Z	Z	Z	Z	Z	NJM	NJM	NJM	NM	NJL	NL	NVL
	M	Z	Z	Z	NJM	NJM	NM	NJL	NJL	NJL	NJL	NL	NL	NVL
	JL	Z	Z	Z	NJM	NJL	NL	NVL	NVL	NVL	NVL	NVL	NVL	NVL
	L	Z	Z	Z	NJM	NJM	NM	NVL	NVL	NVL	NVL	NVL	NVL	NVL
	VL	Z	Z	Z	Z	Z	Z	NVL	NVL	NVL	NVL	NVL	NVL	NVL

Below set-point / Above set-point

To update the controller, what is needed is the appropriate correction for the control-output. Thus the required correction in the process-output must be related to changes to the process-input, and hence the control-output. This can be done using a simple incremental model of the process that relates changes in process-inputs to changes in process-outputs. This model can simply be a matrix of coefficients related to the system Jacobian matrix. For example, suppose the process is a simple dynamical system in which changes in the process-outputs are related to the process-inputs via the equation

$$\dot{\mathbf{y}} = f(\mathbf{u}), \tag{5.28}$$

where $f(\cdot)$ is a nonlinear function. This system can be linearized around a particular operating point, $(\mathbf{y}_0, \mathbf{u}_0)$, by taking the first term of the Taylor series expansion of the above equation

$$f(\mathbf{u}_0 + \delta\mathbf{u}) \approx f(\mathbf{u}_0) + \nabla_{\mathbf{u}} f|_{\mathbf{u}_0} \delta\mathbf{u}, \tag{5.29}$$

giving

$$\delta\dot{\mathbf{y}} = f(\delta\mathbf{u}) = \nabla_{\mathbf{u}} f|_{\mathbf{u}_3} \delta\mathbf{u} = J\delta\dot{\mathbf{u}}, \tag{5.30}$$

where $\delta\mathbf{y}$ and $\delta\mathbf{u}$ are small changes in the process-outputs and inputs, respectively. The matrix J is the system's Jacobian matrix. For a two-input, two-output system

$$\begin{pmatrix} \delta\dot{y}_1 \\ \delta\dot{y}_2 \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \end{pmatrix} \begin{pmatrix} \delta u_1 \\ \delta u_2 \end{pmatrix}, \tag{5.31}$$

where $\mathbf{y} = (y_1, y_2)^T$, $\mathbf{u} = (u_1, u_2)^T$, and $f = (f_1, f_2)^T$.

The change in outputs $\Delta\mathbf{y}$ over a single sampling time, T, due to a change in inputs $\Delta\mathbf{u}$ is approximated by

$$\Delta\mathbf{y} \approx T\delta\dot{\mathbf{y}} = TJ\Delta\mathbf{u} = M\Delta\mathbf{u}. \tag{5.32}$$

The incremental model is given by the matrix of coefficients, M. For a single-input, single-output (SISO) system, the matrix M collapses to a single coefficient which, after normalization, is unity. There is no necessity for the model to be an accurate representation of the process. If it is not accurate, the adaptor will need to make repeated corrections for the same poor performance, which will slow down the learning process. However, the controller should still adapt towards improved performance.

If the performance monitor specifies, at sampling time nT, the required output corrections, $\Delta\mathbf{y}(nT) = \mathbf{c}(nT)$, the required changes in the process-inputs, or input reinforcements, are given by

$$\mathbf{r}(nT) = M^{-1}\mathbf{c}(nT). \tag{5.33}$$

For the single-input, single-output case this reduces to

$$r(nT) = c(nT), \tag{5.34}$$

i.e., the required input reinforcements are simply taken to be equal to the required output corrections.

Adaptation Algorithm

These input reinforcements are the amount that must be added to the process-inputs, i.e., control-outputs, to compensate for the current poor performance. The question remains, however, as to which control-outputs are responsible for the current poor performance. This depends on the dynamics of the process. High-order processes with large time lags will require control-outputs from a long time in the past to be adjusted. Low-order processes with short time lags will require control-outputs much nearer the present to be adjusted. Which control-outputs are corrected must be chosen in advance by the control system designer. Correcting the control-outputs in the FKBC means altering the rule-consequents of the appropriate rules.

Suppose that the current sampling time is n, and the control-output taken m sampling times in the past is the major influence on the current control performance. The numerical values of the error and change-of-error at time were $e(nT - mT)$ and $\Delta e(nT - mT)$, respectively. The control-output was $u(nT - mT)$. These values can be described by the fuzzy sets $\widetilde{LE}_{n-m}$, $\widetilde{L\Delta E}_{n-m}$, and $\widetilde{LU}_{n-m}$, defined as

$$\widetilde{LE}_{n-m}(e) = \begin{cases} 1 & \text{if } e = e(nT - mT) \\ 0, & \text{otherwise,} \end{cases} \tag{5.35}$$

$$\widetilde{L\Delta E}_{n-m}(\Delta e) = \begin{cases} 1 & \text{if } \Delta e = \Delta e(nT - mT) \\ 0, & \text{otherwise,} \end{cases} \tag{5.36}$$

$$\widetilde{LU}_{n-m}(u) = \begin{cases} 1, & \text{if } u = u(nT - mT) \\ 0 & \text{otherwise.} \end{cases} \tag{5.37}$$

The controller response m sampling times in the past can be expressed as the set of n rules

$$\textit{if } e \text{ is } LE^{(i)}_{n-m} \textit{ and } \Delta e \text{ is } L\Delta E^{(i)}_{n-m} \textit{ then } u \text{ is } LU^{(i)}_{n-m}. \tag{5.3}$$

Also, suppose the performance monitor classifies the current performance poor, and specifies that the control-output taken m sampling times ago sho be modified by an input reinforcement of $r(nT)$. That is, the control-output that time should have been $v(nT - mT) = u(nT - mT) + r(nT)$, not $u(nT - m$ If this new control-output is described by the fuzzy set

$$\widetilde{LV}_{n-m}(u) = \begin{cases} 1 & \text{if } u = u(nT - mT) + r(nT) \\ 0. & \text{otherwise,} \end{cases} \tag{5.3}$$

the correct controller response is described by the set of n rules

$$\textit{if } e \text{ is } LE^{(i)}_{n-m} \textit{ and } \Delta e \text{ is } L\Delta E^{(i)}_{n-m} \textit{ then } u \text{ is } LV^{(i)}_{n-m}. \tag{5.4}$$

To adapt the FKBC to reflect this new response, the i-th rule (a bad rule) tha specifies the control-output, $LU^{(i)}_{n-m}$, should be deleted, and the new rule (goo rule) specifying the control-output, $LV^{(i)}_{n-m}$, should be inserted in its place.

Suppose at sampling time n the complete rule set is described by the relatio matrix $\tilde{R}_n$. The meaning of the individual bad and good rules (equations (5.38) and (5.40)) can be expressed by the fuzzy relation matrices given by

$$\tilde{R}_{\text{bad}} = \widetilde{LE}_{n-m} \times \widetilde{L\Delta E}_{n-m} \times \widetilde{LU}_{n-m} \tag{5.41}$$

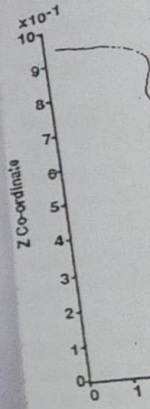
$$\tilde{R}_{\text{good}} = \widetilde{LE}_{n-m} \times \widetilde{L\Delta E}_{n-m} \times \widetilde{LV}_{n-m} \tag{5.42}$$

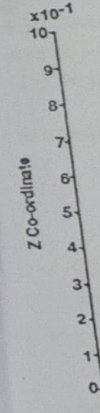
where $\times$ is the Cartesian product.

A new relation matrix, $\tilde{R}_{n+1}$, needs to be generated that no longer includes the bad relation, $\tilde{R}_{\text{bad}}$, but does include the new relation, $\tilde{R}_{\text{good}}$. This requirement can be expressed linguistically (symbolically) as

$$R_{n+1} = (R_n \text{ but not } R_{\text{bad}}) \text{ else } R_{\text{good}}. \tag{5.43}$$

This expression is not unique as a way the relation matrix can be updated. For example,





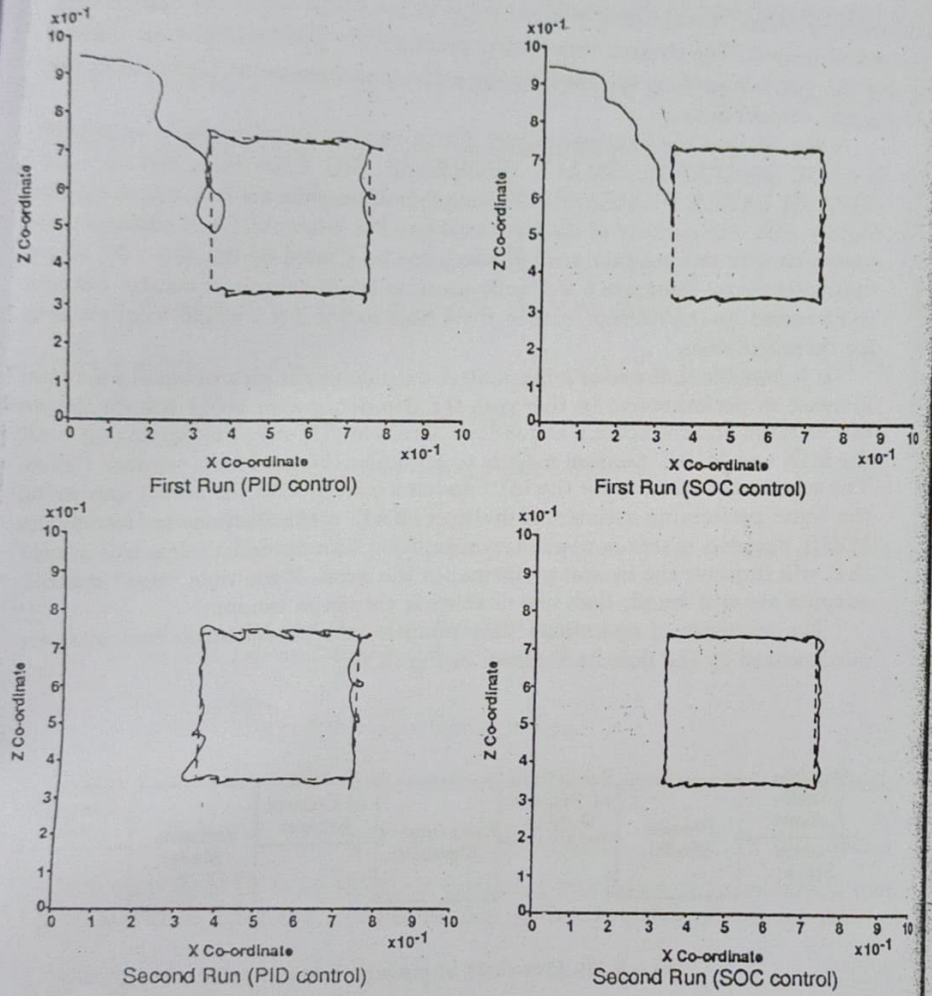


Fig. 5.16. Robot square tracking test (from Tanscheit and Scharf [211]).

5.3.4 A Model Based Controller

A self-organizing system that uses on-line identification of a fuzzy process model for adaptation has been developed by Graham and Newell [81, 82, 83]. The structure of the controller is shown in Fig. 5.17.

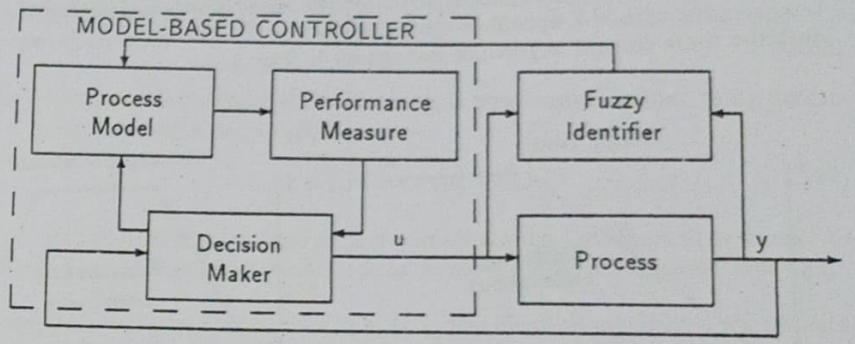


Fig. 5.17. Model-based adaptive fuzzy controller (redrawn from Graham and Newell, [82]).

The FKBC

The FKBC itself is not of the normal type, but is a model based controller (MBC). The controller was originally developed by Hendy [96] and consists of three parts:

1. a fuzzy process model,
2. a controller performance measure,
3. a decision maker.

The fuzzy process model is as described in Section 5.2.1 and consists of a set of if-then rules that are the inverse of the rules found in a Mamdani FKBC. Instead of specifying the desired control-output for a given process state, they predict the process-output to be expected in time due to the current and previous process-states and control-outputs. For example, a fuzzy model of a gas furnace may contain a rule such as:

if the current CO_2 concentration is MEDIUM
and the previous methane feedrate was LOW
then the next CO_2 concentration will be JUST HIGH.

The controller performance measure consists of a group of fuzzy sets. Each fuzzy set describes the required performance of a process variable. The performance of a particular variable is given by the degree of membership of its current value in its performance fuzzy set. For a cement kiln, for example, the performance objectives might be to maximize the amount of material processed by the kiln, i.e., the throughput, while at the same time maintaining the kiln temperature within a specified operating range. Performance fuzzy sets that describe these control objectives are shown in Fig. 5.18.

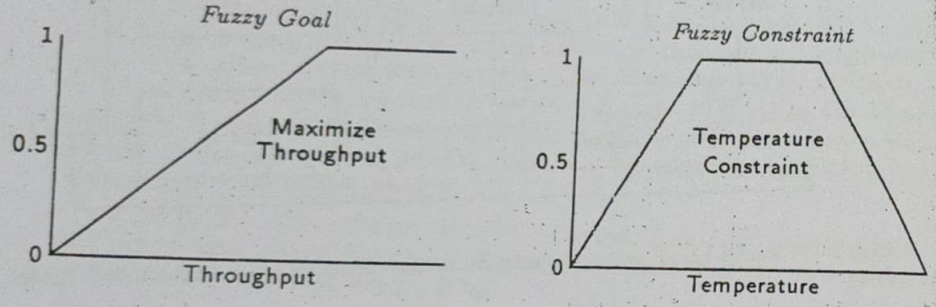


Fig. 5.18. Fuzzy performance measure goals and contraints (redrawn from Graham [81]).

The overall controller performance is given by a confluence rule that combines the individual performance values, for example:

Overall performance = Throughput performance *and*
Temperature performance.

The *and* connective might be implemented as the minimum (MIN decision maker) or product (MULT decision maker) of the individual values. The actual implementation of *and* that is used can influence controller performance. Taking the minimum of the individual performance values results in a controller that will try to improve the worst performing variable, even at the cost of degrading the performance of other variables. The multiplication of the individual values gives a controller that will try to improve the performance of all the variables at the same time. While this ensures that individual performances are generally not degraded, a badly performing variable may only be improved very slowly.

The control-output is calculated by the decision maker, using the fuzzy process model and the performance measure. The aim of the decision maker is to choose, from a predefined, finite set of control-outputs, which action will maximize the performance of the process if no future control-outputs are taken. This is achieved by applying each possible control-output to the fuzzy process

model to obtain predictions of the process-state to be expected due to each control-output. The control-output that produces the predicted process state of highest performance, as specified by the performance measure, is chosen as the current control-output.

A set of possible control-outputs for a single control-variable might be {LARGE DECREASE, SMALL DECREASE, NO CHANGE, SMALL INCREASE, LARGE INCREASE}. The number of possible control-outputs grows rapidly with the number of control-variables. For example, for a multivariable controller with two outputs, each of which can be altered by the above set of five control-outputs, there are $5 \times 5 = 25$ possible combinations of control-outputs to be tested by the decision maker. This rises to $5 \times 5 \times 5 = 125$ combinations for three variables.

It is possible that two or more control-outputs might give an equal predicted increase in performance. In this case the decision maker could simply choose one of these control actions at random. Graham [81] suggests employing both the MIN and MULT decision makers to minimize the need for a random choice. The scheme is to first apply the MIN decision maker, with the aim of improving the worst performing variable. If multiple "best" control actions are found, the MULT decision maker is applied to this "best" set to find the control-output that will improve the overall performance the most. If multiple "best" control-outputs are still found, then one of these is chosen at random.

The sequence of operations that comprise the model based controller are summarized by the flowchart shown in Fig. 5.19.

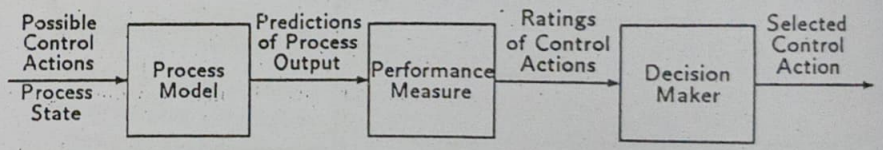


Fig. 5.19. Flowchart of model based controller.

Adaptation Mechanism

Adaptation of the controller is achieved by the use of on-line fuzzy identification of the process model. The controller may start with no model at all, or a predefined model. The on-line identification will build a model, or modify the predefined model to more closely match the process. As the identification proceeds, the performance of the controller improves, because the accuracy of the model predictions of the effect of different control-outputs improves.

The fuzzy model of the process is of the form

$$\widetilde{Ly} = \left(\widetilde{Lx} \times \widetilde{Lu}\right) \circ \tilde{R} = \widetilde{Lx} \circ \widetilde{Lu} \circ \tilde{R}, \tag{5.47}$$

where process-state $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, process-input $\mathbf{u} = (u_1, u_2, \ldots, u_m)^T$, predicted process-output $\mathbf{y} = (y_1, y_2, \ldots, y_\ell)^T$, and $\tilde{R}$ is the fuzzy relation. The aim of the fuzzy identification is to identify the relation matrix $\tilde{R}$ from a given collection of input-output data

$$[\widetilde{Ly}_k; \widetilde{Lx}_k, \widetilde{Lu}_k]_{k=1}^K = [\widetilde{LY}_{1k}, \ldots, \widetilde{LY}_{\ell k}; \widetilde{LX}_{1k}, \ldots, \widetilde{LX}_{nk}, \widetilde{LU}_{1k}, \ldots, \widetilde{LU}_{mk}]_{k=1}^K. \tag{5.48}$$

Each set of data can be written as the if-then rule

if $(x_1$ is LX_{1k} and ... and x_n is $LX_{nk})$
and $(u_1$ is LU_{1k} and ... and u_m is $LU_{mk})$
then $(y_1$ is LY_{1k} and ... and y_ℓ is $LY_{\ell k})$.

The meaning of such a rule is defined by the fuzzy relation

$$\tilde{R}_k = \widetilde{LX}_{1k} \times \ldots \times \widetilde{LX}_{nk} \times \widetilde{LU}_{1k} \times \ldots \times \widetilde{LU}_{mk} \times \widetilde{LY}_{1k} \times \ldots \times \widetilde{LY}_{\ell k}. \tag{5.49}$$

If there is a fuzzy relation $\tilde{R}_k^*$ that satisfies

$$\widetilde{Ly}_k = \widetilde{Lx}_k \circ \widetilde{Lu}_k \circ \tilde{R}_k^*, \tag{5.50}$$

then the relation $\tilde{R}_k$ given by (5.49) is a solution to (5.50).

The entire collection of input-output data can be written as the set of if-then rules:

if $\mathbf{x}$ is Lx_1 *and* $\mathbf{u}$ is Lu_1 *then* $\mathbf{y}$ is Ly_1
if $\mathbf{x}$ is Lx_2 *and* $\mathbf{u}$ is Lu_2 *then* $\mathbf{y}$ is Ly_2
⋮ ⋮ ⋮
if $\mathbf{x}$ is Lx_K *and* $\mathbf{u}$ is Lu_K *then* $\mathbf{y}$ is Ly_K.

The meaning of this set of rules is defined by the overall fuzzy relation $\tilde{R}$ given by

$$\tilde{R} = \tilde{R}_1 \cup \tilde{R}_2 \cup \ldots \cup \tilde{R}_K, \tag{5.51}$$

where each $\tilde{R}_k$ is given by (5.49). In general, a set of data collected from a real process will be noisy and incomplete, and the fuzzy relation $\tilde{R}$ will not satisfy

$$\widetilde{Ly}_k = \widetilde{Lx}_k \circ \widetilde{Lu}_k \circ \tilde{R}, \qquad k = 1, \ldots, K. \tag{5.52}$$

Hence the relation $\tilde{R}$ is only an approximate, and not necessarily optimal, fuzzy model of the process.

If a suitable set of process data has already been collected, this identification scheme can be used to identify a fuzzy model of the process off-line for use by the fuzzy model based controller. It can also form the initial model for the adaptive version of the controller.

The adaptive fuzzy model-based controller uses on-line fuzzy identification in the following algorithm to obtain controller adaptation:
At time 0:

1. An intial relation matrix $\tilde{R}_0$ is defined.
2. The initial process-state $\widetilde{Lx}_0$, process-output $\widetilde{Ly}_0$, and control-output $\widetilde{Lu}_0$ are established.

At time $k+1$:

1. The relation matrix is updated using fuzzy identification via:

$$\tilde{R}_{k+1} = \tilde{R}_k \cup \left(\widetilde{LX}_{1k} \times \ldots \times \widetilde{LX}_{nk} \times \widetilde{LU}_{1k} \times \ldots \times \widetilde{LU}_{mk} \times \widetilde{LY}_{1k} \times \ldots \times \widetilde{LY}_{\ell k}\right). \tag{5.53}$$

2. The model-based controller calculates a new control-output using predictions from the fuzzy model:

$$\widetilde{Ly} = \widetilde{Lx} \circ \widetilde{Lu} \circ \tilde{R}_{k+1}. \tag{5.54}$$

The initial relation matrix may be completely empty, i.e., $\tilde{R}_0 = \emptyset$, or it may be predefined using, say, identification from process data, as suggested above.

Example 5.4 A numerical example of the identification procedure, for a single-input, single-output (SISO) process where the fuzzy model is of the form

$$\widetilde{LY} = \widetilde{LU} \circ \tilde{R}, \tag{5.55}$$

and starting with an initially empty relation matrix, is as follows:
Step 0:

$$\widetilde{LU}_0 = \begin{bmatrix} 1.0 & 0.3 & 0.2 & 0.1 & 0.0 & 0.0 \end{bmatrix}$$
$$\widetilde{LY}_0 = \begin{bmatrix} 1.0 & 0.8 & 0.9 & 0.6 & 0.5 & 0.7 \end{bmatrix}$$
$$\tilde{R}_0 = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Step 1:

$$\widetilde{LU}_0 = \begin{bmatrix} 1.0 & 0.3 & 0.2 & 0.1 & 0.0 & 0.0 \end{bmatrix}$$
$$\widetilde{LY}_0 = \begin{bmatrix} 1.0 & 0.8 & 0.9 & 0.6 & 0.5 & 0.7 \end{bmatrix}$$
$$\tilde{R}_1 = \tilde{R}_0 \cup (\widetilde{LU}_0 \times \widetilde{LY}_0) = \begin{bmatrix} 1.0 & 0.8 & 0.9 & 0.6 & 0.5 & 0.7 \\ 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$