ELSEVIER

Fuzzy Sets and Systems 87 (1997) 141–154

FUZZY sets and systems

A PD-like self-tuning fuzzy controller without steady-state error

Chun-Tang Chao*, Ching-Cheng Teng

Institute of Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan

Received January 1995; revised January 1996

Abstract

This paper presents a PD-like self-tuning fuzzy controller (STFC) based on the tuning of scaling factors. A two-stage tuning method including a direct tuning stage and indirect tuning stage is proposed. For most linear controlled plants, application of the direct tuning stage alone is enough to yield satisfactory system responses. The proposed STFC can automatically detect the operating ranges of input variables and then adjust the scaling factors. In the indirect tuning stage the back-propagation algorithm can be directly applied to fine-tune the scaling factors when a more complicated plant is controlled. Furthermore, a simple but efficient method for fully compensating for steady-state error, which is the main problem in a PD-like fuzzy logic control system, is proposed so that tuning for fuzzy logic rule bases is unnecessary. Simulation results with linear time-invariant systems and a nonlinear unstable system as the controlled plants show that the proposed technique yields zero steady-state error responses very quickly without overshoot or oscillatory behavior. © 1997 Elsevier Science B.V.

Keywords: Control theory; PD controller; Fuzzy controller; Scaling factor; Steady-state error

1. Introduction

The concept of a fuzzy set can be directly attributed to the seminal work of Zadeh in 1965 [24]. Inspired by Zadeh, Mamdani and his associates [13] used a rule-based system with fuzzy parameters to construct a controller that emulated the performance of a human operator. Sugeno, who appreciated the power of this new fuzzy rule-based paradigm for building controllers, began developing applications for this new methodology [19]. The main advantage of the fuzzy logic controller (FLC) is that it can be applied to plants that are difficult to model mathematically, and the controller can be designed to apply heuristic rules that reflect the experience of human experts.

The first step in the design of an FLC is to determine the input and output variables of the FLC [22]. In this step, the designers determine whether they will be using a PI-like, PD-like, or PID-like FLC. Since it is unrealistic to expect that an operator or expert can formulate reasonable control rules, considering third and higher dimensions, most common FLCs are PI-like or PD-like controllers [22]. Though the FLC exhibits superior applicability to the traditional PID controller [5, 20] and is highly robust [3], PI-like and PD-like FLCs possess mainly the same characteristics as traditional PI and PD controllers, respectively. That is, the PI-like FLC adds damping to a system and reduces steady-state error, but yields penalized rise time and settling time. The PD-like FLC adds damping and reliably predicts large overshoots, but does not improve the steady-state response.

* Corresponding author.

PII S0165-0114(96)00022-X

Much research on the PD-like FLC has either considered only simulation examples with no steady-state error problem [4] or reduced the steady-state error by fine tuning the rule bases, performing parameter optimization, and increasing the number of rules [8]. On the other hand, though the PI-like FLC can solve the steady-state error problem, techniques such as scaling-factor adjustment, rule modification [12], and membership-function shifting [25] are required in order to reduce the rise time and improve oscillatory behavior in the step response.

Once the membership functions and the rule-base of the FLC are set up, the next problem related to its implementation is the issue of tuning. The scaling factors (SF) are the main parameters used for tuning the FLC. If we demonstrate the virtual PI (or PD) approximation of the PI-like (or PD-like) FLC, we can find that variations in the SFs result in modification of the poles of the overall transfer function and its zero [7]. This is the reason that changes in the SFs have a dramatic influence on the overall dynamics of the closed loop system. There is still no standard method for tuning the SFs. Thus an FLC that incorporates a systematic method for adjusting the SFs is urgently needed. Tuning rules for tuning the FLC by manipulating the SFs have been proposed in [6, 12, 15]. In these methods, however, the SFs can be tuned simultaneously only after information such as rise-time, overshoot, degree of oscillation, settling time, and steady-state error is obtained. In [4], many tuning tables, in numerical form, were constructed for real-time simultaneous tuning of SFs. However, care must be taken in determining several parameters in [4].

In recent years there have been considerable developments in the tuning of parameters in fuzzy logic systems [9, 10, 21] using the gradient-descent-based back-propagation (BP) algorithm [16], similar to methods in neural networks [14]. However, to use the BP method in the fuzzy logic systems, most authors construct the so-called neural-network-based fuzzy logic systems [10, 11, 21]. We can find that it is time-consuming to take hundreds of learning epochs to train so many parameters in the fuzzy neural networks.

In this paper we favor the PD-like FLC, because it yields quick response with less oscillation than the PI-like FLC. Moreover, the problem of compensating for steady-state error in the PD-like FLC can be solved by a simple method in the proposed system. Owing to the advantages of the proposed controller, the tuning procedure for SF tuning is so simple that a direct tuning-stage alone is sufficient for the control of most linear plants. In the direct tuning-stage, the operating ranges of the input variables of the FLC can be detected automatically. When the controlled plant is more complicated, on the other hand, the BP algorithm is applied to adaptively tune the SFs in the indirect tuning stage. We do not need to construct a fuzzy-neural-network structure that there are only three parameters to be directly tuned in this way. Furthermore, since the indirect tuning stage is based on the results obtained in the above direct tuning stage, only few learning steps are required. Simulation results indicate that the proposed self-tuning fuzzy controller (STFC) is indeed efficient.

The network structure and the tuning methods of the proposed STFC will be described in Section 2. A simple method for compensating for the steady-state error in a PD-like FLC is presented in Section 3. Section 4 describes the proposed two-stage tuning method. The results of simulations conducted to evaluate the proposed STFC are presented in Section 5. Section 6 concludes the paper.

2. The proposed STFC

In this section, several tables are constructed that will be used in the tuning procedures in the proposed STFC. The procedure for detecting the operating ranges of the input variables and the supervised learning used for adaptively tuning the SFs will also be described.

2.1. Modified decision table

A block diagram of a basic feedback control system is shown in Fig. 1. The purpose of the feedback controller under consideration is to maintain the output $y(k)$ close to the set point sp. This is the so-called *regulation* problem. The definitions for the error $e(k)$ and *error change* $\Delta e(k)$ are

$$e(n) = sp - y(n), \tag{1}$$

$$\Delta e(n) = e(n) - e(n-1) = -(y(n) - y(n-1)), \tag{2}$$

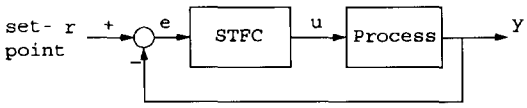


Fig. 1. Block diagram of a basic feedback control system.

Table 1
The control rules

	E^*						
ΔE^*	NL	NM	NS	Z	PS	PM	PL
NL	NL	NL	NM	NM	NS	NS	Z
NM	NL	NM	NM	NS	NS	Z	PS
NS	NM	NM	NS	NS	Z	PS	PS
Z	NM	NS	NS	Z	PS	PS	PM
PS	NS	NS	Z	PS	PS	PM	PM
PM	NS	Z	PS	PS	PM	PM	PL
PL	Z	PS	PS	PM	PM	PL	PL

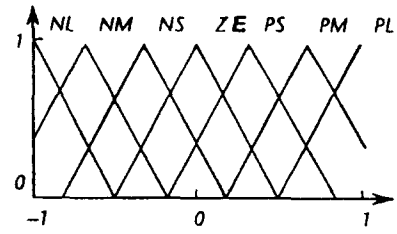


Fig. 2. Membership functions applied for the control rules.

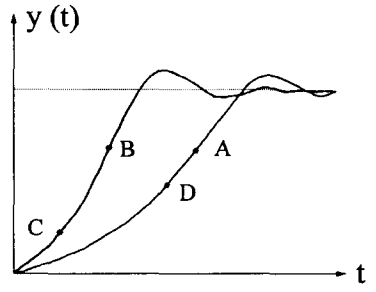


Fig. 3. Responses of process.

where indices n and $n-1$ indicate the present state and the previous state of the system.

The control rules with two inputs and a single output fuzzy variables E^*, ΔE^*, and U^*, representing $e^*(k)$, $\Delta e^*(k)$, and $u^*(k)$ of the controller, are shown in Table 1. We consider an FLC with the input and output SFs. The control rules in Table 1 are based on the characteristics of the step response [18]. The term sets of E^*, ΔE^*, and U^* include the linguistic labels "positive large" (PL), "positive medium" (PM), "positive small" (PS), "zero" (Z), "negative small" (NS), "negative medium" (NM), and "negative large" (NL). The membership functions of the respective reference fuzzy sets are plotted in Fig. 2. We combine the well-known *min–max* inference method cooperating with the popular center-of-area (COA) defuzzification procedure to produce the look-up table in Table 2, which will be referred to below as the decision table (DT).

The DT will be modified slightly so as to satisfy the two criteria shown below, where fixed SFs are considered.

1. For two points in the system response with the same e^*, the point with the greater Δe^* should have the greater u^* in control action.

2. For two points in the system response with the same Δe^*, the point with the greater e^* should have the greater u^* in control action.

We need to note that the word "greater" means "greater in signed value", e.g., 5 is greater than 3 and -3 is greater than -5. It is reasonable that a PD-like FLC should obey these two criteria. The reasoning behind these criteria is as follows. When plants are feedback-controlled, we can imagine obtaining the two responses shown in Fig. 3. Point A has the same error value as point B, but point A has a greater Δe than point B. A reasonable control output for point A should be greater than that for point B. On the other hand, suppose points C and D have the same error change, while the e for point C is significantly greater than that for point D. Then the control output for point C should be greater than that for point D. In the above cases, we have $e > 0$ and $\Delta e < 0$. The modified decision table (MDT) constructed by applying the above criteria is shown in Table 3, where the elements in bold-face have been modified from the original DT.

Instead of quantizing the scaling results as the indices to map an element in the MDT, the mapped element is obtained by performing *interpolation*. From the MDT, the approximate equations for calculating

Table 2
The decision table (DT)

Δe^* \ e^*	−1.2	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.2
−1.2	−0.95	−0.95	−0.92	−0.80	−0.75	−0.67	−0.54	−0.50	−0.41	−0.25	−0.17	−0.12	−0.00
−1.0	−0.95	−0.85	−0.84	−0.82	−0.68	−0.56	−0.56	−0.47	−0.30	−0.15	−0.07	−0.00	0.12
−0.8	−0.92	−0.84	−0.75	−0.75	−0.68	−0.52	−0.47	−0.44	−0.24	−0.09	0.00	0.07	0.17
−0.6	−0.80	−0.82	−0.75	−0.56	−0.56	−0.52	−0.34	−0.24	−0.12	−0.00	0.09	0.15	0.25
−0.4	−0.75	−0.68	−0.68	−0.56	−0.40	−0.40	−0.26	−0.09	−0.00	0.12	0.24	0.30	0.41
−0.2	−0.67	−0.56	−0.52	−0.52	−0.40	−0.20	−0.09	0.00	0.09	0.24	0.44	0.47	0.50
0.0	−0.54	−0.56	−0.47	−0.34	−0.26	−0.09	0.00	0.09	0.26	0.34	0.47	0.56	0.54
0.2	−0.50	−0.47	−0.44	−0.24	−0.09	0.00	0.09	0.20	0.40	0.52	0.52	0.56	0.67
0.4	−0.41	−0.30	−0.24	−0.12	−0.00	0.09	0.26	0.40	0.40	0.56	0.68	0.68	0.75
0.6	−0.25	−0.15	−0.09	−0.00	0.12	0.24	0.34	0.52	0.56	0.56	0.75	0.82	0.80
0.8	−0.17	−0.07	0.00	0.09	0.24	0.44	0.47	0.52	0.68	0.75	0.75	0.84	0.92
1.0	−0.12	−0.00	0.07	0.15	0.30	0.47	0.56	0.56	0.68	0.82	0.84	0.85	0.95
1.2	−0.00	0.12	0.17	0.25	0.41	0.50	0.54	0.67	0.75	0.80	0.92	0.95	0.95

Table 3
The modified decision table (MDT)

Δe^* \ e^*	−1.2	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	0.2	0.4	0.6	0.8	1.0	1.2
−1.2	−0.95	−0.95	−0.92	−0.80	−0.75	−0.67	−0.54	−0.50	−0.41	−0.25	−0.17	−0.12	−0.00
−1.0	−0.95	−0.85	−0.84	−0.82	**−0.69**	−0.56	**−0.53**	−0.47	−0.30	−0.15	−0.07	−0.00	0.12
−0.8	−0.92	−0.84	**−0.76**	−0.75	−0.68	−0.52	−0.47	−0.44	−0.24	−0.09	0.00	0.07	0.17
−0.6	−0.80	−0.82	−0.75	**−0.57**	−0.56	**−0.50**	−0.34	−0.24	−0.12	−0.00	0.09	0.15	0.25
−0.4	−0.75	**−0.69**	−0.68	−0.56	**−0.41**	−0.40	−0.26	−0.09	−0.00	0.12	0.24	0.30	0.41
−0.2	−0.67	−0.56	−0.52	**−0.50**	−0.40	−0.20	−0.09	0.00	0.09	0.24	0.44	0.47	0.50
0.0	−0.54	**−0.53**	−0.47	−0.34	−0.26	−0.09	0.00	0.09	0.26	0.34	0.47	0.56	0.54
0.2	−0.50	−0.47	−0.44	−0.24	−0.09	0.00	0.09	0.20	0.40	**0.50**	0.52	**0.57**	0.67
0.4	−0.41	−0.30	−0.24	−0.12	−0.00	0.09	0.26	0.40	**0.41**	0.56	**0.66**	0.68	0.75
0.6	−0.25	−0.15	−0.09	−0.00	0.12	0.24	0.34	0.50	0.56	**0.57**	0.75	0.82	0.80
0.8	−0.17	−0.07	0.00	0.09	0.24	0.44	0.47	0.52	**0.66**	0.75	**0.76**	0.84	0.92
1.0	−0.12	−0.00	0.07	0.15	0.30	0.47	0.56	**0.57**	0.68	0.82	0.84	0.85	0.95
1.2	−0.00	0.12	0.17	0.25	0.41	0.50	0.54	0.67	0.75	0.80	0.92	0.95	0.95

$\partial u^*/\partial e^*$ and $\partial u^*/\partial \Delta e^*$ shown below can be obtained; these equations will be used in the indirect tuning stage:

$$\frac{\partial u^*}{\partial e^*} = \frac{MDT(\Delta e^*, e^* + I(e^*)) - MDT(\Delta e^*, e^*)}{I(e^*)}, \quad \text{for } I(e^*) \to 0 \tag{3}$$

$$\frac{\partial u^*}{\partial \Delta e^*} = \frac{MDT(\Delta e^* + I(\Delta e^*), e^*) - MDT(\Delta e^*, e^*)}{I(\Delta e^*)}, \quad \text{for } I(\Delta e^*) \to 0, \tag{4}$$

where $I(x)$ is a function to take little increment on x. In the proposed system, we assign $I(e^*) = I(\Delta e^*) = 0.01$. We also find that the inferred $\partial u^*/\partial e^*$ and $\partial u^*/\partial \Delta e^*$ are positive, as required by criterion 2 and criterion 1, respectively. Moreover, Eqs. (3) and (4) indicate that a DT or MDT with interval $[-1.2, 1.2]$ is convenient for programming.

2.2. Layered operation

The proposed four-layer network structure of the STFC is illustrated in Fig. 4. In this subsection, we

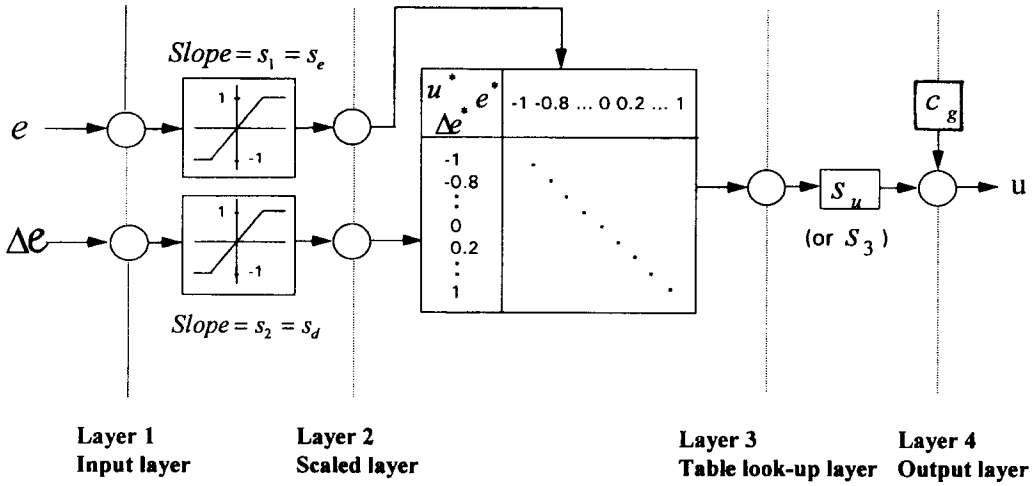


Fig. 4. The network structure of the proposed STFC.

shall describe the signal propagation and the basic function of the nodes in each layer. We use net_j and f_j to denote the summed net input and activation function of node j, respectively, and the superscript denotes the layer number. Moreover, x_j^k and y_j^k denote the input and output vector of the jth node in layer k, respectively.

Layer 1: *input layer.* For the jth, $j = 1, 2$, node of layer one, the net input and the net output are

$$net_j^1 = x_j^1 \quad \text{and} \quad y_j^1 = f_j^1(net_j^1) = net_j^1, \tag{5}$$

where $x_1^1 = e$ and $x_2^1 = \Delta e$.

Layer 2: *scaled layer.* In this layer, the operating ranges of the measured variable $e(k)$ and $\Delta e(k)$ are transformed to the normalized universe $[-1, 1]$ by the scaling factors s_e and s_d, respectively. Before the layer two operation, an *SF-adjustment* procedure is performed:

$$s_e = \begin{cases} 1/e & \text{if } es_e > 1 \\ -1/e & \text{if } es_e < -1 \end{cases} \tag{6}$$

$$s_d = \begin{cases} 1/\Delta e & \text{if } \Delta es_d > 1 \\ -1/\Delta e & \text{if } \Delta es_d < -1 \end{cases} \tag{7}$$

The SF-adjustment procedure is crucial for the direct tuning stage, as will be explained in Section 4. We can find that the input scaling factors are changed only when the $e_{\max}$ or $\Delta e_{\max}$ is obtained. Thus, it seems that we can predict the operating ranges OR_e and OR_d of e and Δe, respectively, by

$$OR_e = [-e_{\max}, e_{\max}] = [-1/s_e, 1/s_e],$$
$$OR_d = [-\Delta e_{\max}, \Delta e_{\max}] = [-1/s_d, 1/s_d].$$

After the SF-adjustment procedure, the operation in this layer is

$$net_j^2 = s_j * x_j^2 \quad \text{and} \quad y_j^2 = f_j^2(net_j^2) = net_j^2. \tag{8}$$

We note that $s_1 = s_e$, $s_2 = s_d$, $y_1^2 = e^*$, and $y_2^2 = \Delta e^*$.

Layer 3: *table look-up layer.* In this layer, we perform MDT-mapping for the scaling results e^* and Δe^*. Thus we have

$$net^3 = MDT(x_2^3, x_1^3) \quad \text{and}$$
$$y^3 = u^* = f^3(net^3) = net^3. \tag{9}$$

Layer 4: *output layer.* The mapped element in the last layer is also scaled by an SF, say s_u (or s_3), and is added by a constant gain c_g to arrive at the desired control signal. The final output of the network is

$$net^4 = s_u * x^4 + c_g \quad \text{and}$$
$$y^4 = u = f^4(net^4) = net^4. \tag{10}$$

The constant gain c_g is crucial for zeroing steady-state error, as will be explained in the next section.

2.3. Supervised gradient descent learning

Since the proposed STFC is a four-layer feedforward network, it is a straightforward task to apply the gradient-descent-based BP algorithm [16] to adaptively adjust the SFs. The goal is to minimize a cost function, E, so that training pattern k is proportional to the square of the difference between the set point sp and the plant output $y(k)$. Let E be defined by

$$E = \tfrac{1}{2}(sp - y(k))^2. \tag{11}$$

If s_i, $i = 1,2,3$, is the adjusted SF, then the learning rule is

$$s_i(k+1) = s_i(k) - \eta_i \frac{\partial E}{\partial s_i} + \alpha_i \Delta s_i(k) \tag{12}$$

and

$$\Delta s_i(k) = s_i(k) - s_i(k-1), \tag{13}$$

where η_i is the learning rate and α_i, $0 < \alpha_i < 1$, is the momentum parameter. We will now to derive the learning law for each layer in the feedback direction.

Layer 4: The gradient of error in (11) with respect to an arbitrary weighting vector $W \in \mathbb{R}^n$ is

$$\frac{\partial E}{\partial W} = e(k)\frac{\partial e(k)}{\partial W} = -e(k)\frac{\partial y(k)}{\partial W}$$

$$= -e(k)\frac{\partial y(k)}{\partial u(k)}\frac{\partial u(k)}{\partial W} = -e(k)y_u(k)\frac{\partial O(k)}{\partial W}, \tag{14}$$

where $O(k)$ is the output of the STFC and $S = y_u(k) = \partial y(k)/\partial u(k)$ is the plant sensitivity. From (14), we can derive the propagating error term given by the output node

$$\delta^4 = \frac{-\partial E}{\partial net^4} = e(k)y_u(k)\frac{\partial y^4}{\partial net^4} = e(k)y_u(k). \tag{15}$$

Then, we have

$$-\frac{\partial E}{\partial s_3} = -\frac{\partial E}{\partial y^4}\frac{\partial y^4}{\partial net^4}\frac{\partial net^4}{\partial s_3} = \delta^4 y^3 = \delta^4 u^*. \tag{16}$$

Hence, by (12), the scaling factor s_3 is updated by

$$s_3(k+1) = s_3(k) + \eta_3 \delta^4(k) y^3(k) + \alpha_3 \Delta s_3(k). \tag{17}$$

Layer 3: The error term δ^3 is derived as follows:

$$\delta^3 = \frac{-\partial E}{\partial net^3} = \frac{-\partial E}{\partial y^3}\frac{\partial y^3}{\partial net^3} = \frac{-\partial E}{\partial y^3} \tag{18}$$

$$= \frac{-\partial E}{\partial y^4}\frac{\partial y^4}{\partial y^3} = \delta^4 s_3. \tag{19}$$

Layer 2: First, the error term is computed:

$$\delta_j^2 = \frac{-\partial E}{\partial net_j^2} = \frac{-\partial E}{\partial y_j^2}\frac{\partial y_j^2}{\partial net_j^2}$$

$$= \frac{-\partial E}{\partial y^3}\frac{\partial y^3}{\partial y_j^2} = \delta^3 \frac{\partial y^3}{\partial y_j^2}$$

$$= \begin{cases} \delta^3 \dfrac{\partial u^*}{\partial e^*} & \text{for } j = 1, \\ \delta^3 \dfrac{\partial u^*}{\partial \Delta e^*} & \text{for } j = 2. \end{cases} \tag{20}$$

We can then derive

$$-\frac{\partial E}{\partial s_j} = -\frac{\partial E}{\partial y_j^2}\frac{\partial y_j^2}{\partial s_j} = \delta_j^2 \frac{\partial y_j^2}{\partial s_j} = \delta_j^2 \cdot x_j^1$$

$$= \begin{cases} \delta_j^2 x_j^1 & \text{for } -1 < y_j^2 < 1, \\ 0 & \text{for } y_j^2 = 1 \text{ or } y_j^2 = -1. \end{cases} \tag{21}$$

Thus, the update rules for s_j, $j = 1,2$, are

$$s_j(k+1) = s_j(k) + \eta_j \delta_j^2 x_j^1 + \alpha_j \Delta s_j(k). \tag{22}$$

3. A simple method for zeroing steady-state error

PD control can reliably predict and correct large overshoots, but the derivative control will affect the steady-state error of a system only if the steady-state error varies with time. If the steady-state error of a system is constant with respect to time, the time derivative of the error will be zero, and derivative control will have no effect on the steady-state error. In this paper we consider the steady-state error for a step input in a control system with PD control. We try to fully compensate for the steady-state error by a proposed simple method.

Basically, the proposed STFC in Fig. 4 is composed of a fuzzy PD controller and an added constant gain. In the following, we will show how to fully compensate for the steady-state error by this structure. From the operations in the table look-up layer, we shall demon-

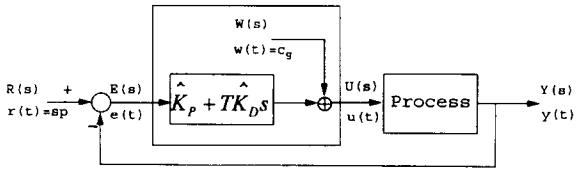


Fig. 5. Control system with PD control.

strate the effect of the normalization on the virtual PD approximation as

$$u^*(k) = \bar{K}_P e^*(k) + \bar{K}_D \Delta e^*(k). \tag{23}$$

Thus, the following equality will be true for the real values:

$$\begin{aligned} u(k) &= s_u u^*(k) + c_g \\ &= \bar{K}_P(s_u s_e) e(k) + \bar{K}_D(s_u s_d)\Delta e(k) + c_g \\ &= \hat{K}_P e(k) + \hat{K}_D \Delta e(k) + c_g. \end{aligned} \tag{24}$$

We can transform the above discrete expression to its continuous form

$$u(t) = \hat{K}_P e(t) + T\hat{K}_D \dot{e}(t) + c_g, \tag{25}$$

where T is the sampling period.

A control system with fuzzy PD control, added by a constanr gain, is shown in Fig. 5. Let $G_p(s)$ denote the transfer function of the linear controlled plant. If we first ignore the constant gain c_g, the DC gain from set point sp to the plant output y_p is

$$G(0) = \frac{\hat{K}_P G_p(0)}{1 + \hat{K}_P G_p(0)}, \tag{26}$$

where $G(s)$ is the transfer function of the overall system and $G_p(0)$, $G_p(0) > 0$, is the DC gain of the plant. From (26), we can see that steady-state error due to a step-function input will always be present if $G_p(0) \to \infty$ is not satisfied. From the viewpoint of the FLC, this means a trajectory converging to the origin ($e = \Delta e = 0$) in linguistic space [1], with fuzzy variables E and ΔE as its axes, is impossible unless $G_p(0) \to \infty$ is satisfied. When $G_p(0)$ does not approach infinity, the linguistic trajectory of the overall control response will eventually reach a stable point in the neighborhood of the origin with a certain degree of steady-state error.

We will make the linguistic trajectory converges exactly to the origin with the help of the constant gain c_g. Considering the equilibrium point at $\boldsymbol{x} = [x_1(t)\ x_2(t)]^T = [e(t)\ \dot{e}(t)]^T = \mathbf{0}$, where the system output response is maintained at set point sp and the output of the fuzzy PD controller is zero, we have

$$c_g G_p(0) = sp. \tag{27}$$

We find that the value of c_g has no relationship with the control constant gains $\hat{K}_P$ or $\hat{K}_D$. Also, it seems that if we set

$$c_g = \frac{sp}{G_p(0)}, \tag{28}$$

we can solve the steady-state error problem. To show this, we can derive the output transfer function

$$\begin{aligned} Y(s) = &\frac{G_p(s)(T\hat{K}_D s + \hat{K}_P)}{G_p(s)(T\hat{K}_D s + \hat{K}_P) + 1} R(s) \\ &+ \frac{G_p(s)}{G_p(s)(T\hat{K}_D s + \hat{K}_P) + 1} W(s), \end{aligned} \tag{29}$$

where we can find that the step signal $w(t)$ does not influence the stability of the overall system. By substituting sp/s and c_g/s, respectively, for $R(s)$ and $W(s)$ in (29), we have the steady-state output

$$\begin{aligned} \lim_{t\to\infty} y_p(t) &= \frac{G_p(0)\hat{K}_P}{G_p(0)\hat{K}_P + 1} sp \\ &\quad + \frac{G_p(0)}{G_p(0)\hat{K}_P + 1} \frac{sp}{G_p(0)} \\ &= sp, \end{aligned} \tag{30}$$

with steady-state error $e_{ss} = 0$.

4. Two-stage tuning method

In Sections 2 and 3, we obtained an important fact that variations in the SFs implies the variations in the operating ranges OR_e, OR_d, and OR_u. It means that if we want to fine-tune SFs, the remaining problem is how to automatically detect the operating ranges and then adjust the SFs. For brevity, we neglect the magnitude-constraint on the controller

output $u(k)$. Instead of tuning the SFs through trial-and-error, the proposed system employs a systematic approach:

1. In the *direct tuning stage*, the SFs are directly and iteratively adjusted to reduce the error between the plant output and the set point. The supervised learning described in Section 2.3 is not used in this stage.

2. In the *indirect tuning stage*, the BP algorithm is applied to fine-tune the SFs obtained in the above stage. This tuning stage can be omitted if the desired control result is achieved in the direct tuning stage.

In the direct tuning-stage, we first initialize s_e and s_d with large values and then increase the value of s_u, starting from a low value, in each iteration. Generally speaking, when the value of s_3 increases, the operating range of $e(k)$ and $\Delta e(k)$ will expand since the variation in system response is greater. The SF-adjustment procedure in (6) and (7) will detect this situation and reduce the value of s_e and s_d. Thus, though only the layered operation is performed in this stage, the SFs will be automatically adjusted. Of course, s_e will be adjusted as

$$s_e = \frac{1}{sp} \tag{31}$$

in this tuning stage for most cases, except when the step response exhibits a small negative undershoot near the time index $k = 0$. Note that if the initial values s_e and s_d remain unchanged after the first iteration, this means that their values should be further enlarged. For most of the linear controlled plants studied in our simulations, this simple direct tuning stage is sufficient to provide the desired output. The user can record the performance indices such as the rise time, reach time, overshoot, and settling time after each iteration to determine whether the desired system response has been achieved.

Furthermore, in the first iteration of the direct tuning stage, we can assume that $G_p(0) \to \infty$ and let $c_g = 0$. If we obtain the system output with nonzero steady-state error, we then do approximation on

$$G_p(0) = \frac{y_{ss}}{u_{ss}} \tag{32}$$

by the steady-state input u_{ss} and output y_{ss} of the plant in the following tuning iterations. In this way, the proposed compensation method will be robust to varying sizes of DC disturbances of the controlled plant. This approach is adopted in the simulation examples in Section 5.

If the desired control result is not achieved in the above stage, supervised learning can be applied in the indirect tuning stage, and a plant-identifier can be employed to identify the plant sensitivity y_u to implement the indirect adaptive FLC. We refer readers to reference [2], in which a FNN identifier is applied. It is not practical to simultaneously tune SFs s_e, s_d, and s_u to obtain a desired response with freely chosen initial values, because the process may be trial-and-error and learning convergence problems may arise. It is feasible to perform supervised learning based on the SFs obtained in the first tuning stage, however, because in this case the tuning method is likely to converge to a desired solution. In the proposed system, *micro-tuning* of the results of the first stage is performed in the second stage if we set all learning parameters to smaller values. On the other hand, varying the SFs by setting all learning parameters to larger values can result in considerable nonlinearity of the fuzzy controller, but convergence is hard to be guaranteed in this approach. Simulation results indicate that the two-stage tuning method is feasible and efficient.

5. Simulation examples

In the following simulation examples, suppose each learning cycle takes *te* seconds with a step size of *ts* seconds.

Example 1 (*An under-damped system*). The plant to be controlled is described by the Laplace transfer function [8]

$$G_p(s) = \frac{20}{s^2 + 8s + 20}.$$

Let $te = 10$, $ts = 0.02$, and $sp = 4$, we initialize $s_e = s_d = 1000$ and $s_u = 20$. The value of s_u is increased by one in each iteration. Supervised learning is not applied here, so no plant-identifier is required. When steady-state error compensation method is applied, Table 4 shows values of the main parameters in the first five iterations. We find that the SFs are automatically tuned by the proposed SF-adjustment

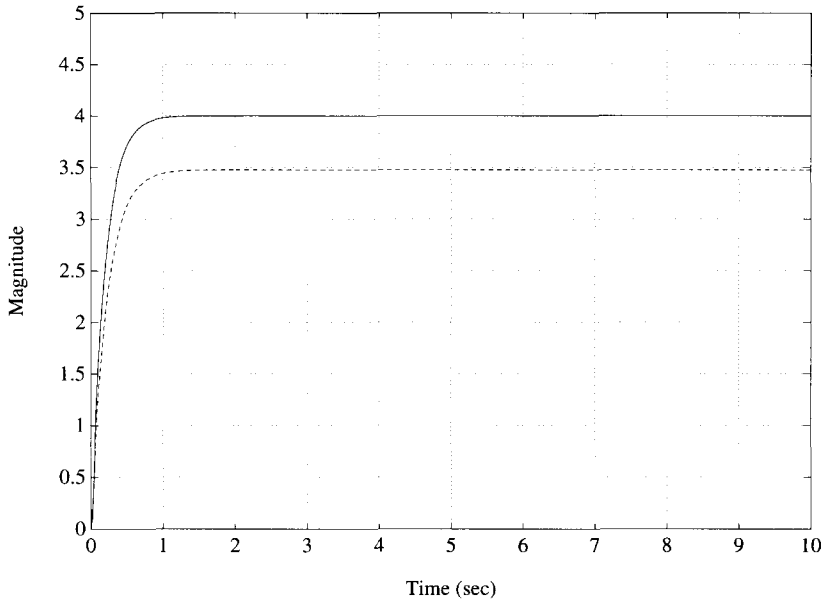


Fig. 6. Step response of the process with $s_u = 60$ in Example 1. The dashed line indicates the system response before steady-state error compensation ($s_e = 0.25$ and $s_d = 2.77$); the solid line indicates the response after steady-state error compensation ($s_e = 0.25$ and $s_d = 2.357$).

Table 4
Values of the main parameters in the first five iterations

Iteration	s_e	s_d	s_u	c_g	Predicted OR_e	Predicted OR_d	y_{ss}
1	0.25	7.35	20.0	0.00	[−4.0,4.0]	[−0.136, 0.136]	2.934
2	0.25	5.57	21.0	4.00	[−4.0,4.0]	[−0.180,0.180]	4.000
3	0.25	5.02	22.0	4.00	[−4.0,4.0]	[−0.199,0.199]	4.000
4	0.25	4.78	23.0	4.00	[−4.0,4.0]	[−0.209,0.209]	4.000
5	0.25	4.62	24.0	4.00	[−4.0,4.0]	[−0.216,0.216]	4.000

procedure. To illustrate the efficiency of the direct tuning stage, Fig. 6 shows the step responses before and after steady-state error compensation when $s_u = 60$. We find that the original steady-state error of 0.53 quickly decreases to zero. We also find that the proposed direct tuning method can automatically adjust s_e and s_d by detecting the operating ranges of $e(k)$ and $\Delta e(k)$.

Example 2 (*A damped oscillation system*). The controlled system is a second-order system modeled as follows:

$$\ddot{y}(t) + 0.40\dot{y}(t) + 0.54y(t) = 19.54u(t).$$

This is a variation of a vehicle speed control system with a potential disorder [12]. Let $te = 6$, $ts = 0.02$, and $sp = 60$. Then we can control the plant as well as that in Example 1. If the proposed compensation method is not applied, most designers would increase the value of s_u to reduce the steady-state error. Fig. 7 presents an example which shows that the steady-state

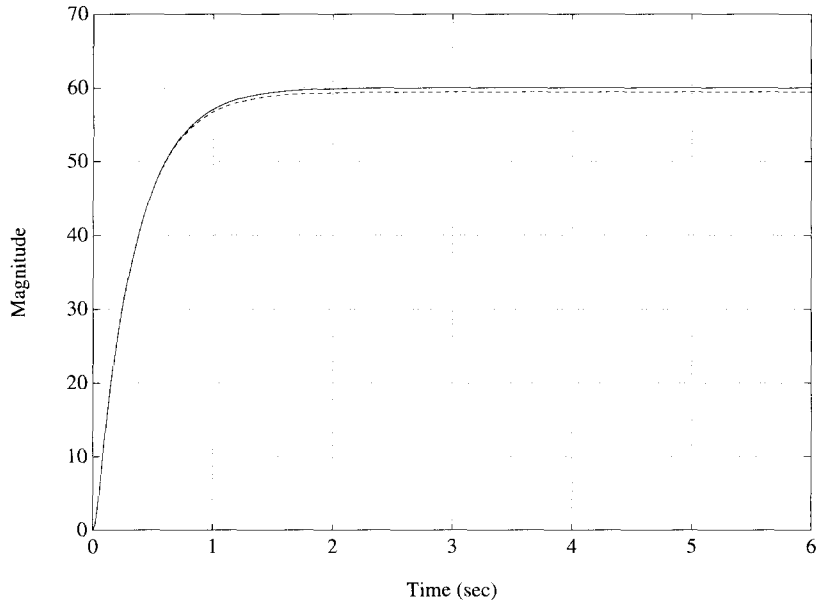


Fig. 7. Step response of the process with $s_u = 400$ in Example 2. The dashed line indicates the system response before steady-state error compensation ($s_e = 0.017$ and $s_d = 0.294$); the solid line indicates the response after steady-state error compensation ($s_e = 0.017$ and $s_d = 0.290$).

error cannot be completely eliminated even when s_u is increased to 400. When the proposed method is applied, however, the steady-state error is very quickly reduced from 0.56 to zero.

Example 3 (*A time-delay system*). The transfer function of the controlled plant is

$$G_p(s) = \frac{1}{s+1}\exp(-0.5s),$$

which involves a time delay. This example corresponds to situations often encountered in industrial processes. Let $t_e = 20$, $t_s = 0.02$, and $sp = 1$. As in the above examples, we initialize $s_e = s_d = 1000$ and $s_u = 1$ and then increase the value of s_u by one in each iteration. Fig. 8 shows the step response before and after steady-state error compensation when s_u is 19. The system response with zero steady-state error again confirms the applicability and efficiency of the proposed system.

Example 4 (*A nonlinear unstable system*). The plant is described by the differential equation [23]

$$\dot{y} = \tfrac{1}{2}y^2 - y + u.$$

In this case we let $te = 7.5$, $ts = 0.015$, and $sp = 2.0$. In this example, the suggested two-stage tuning method is introduced because of the complexity of the controlled plant. In the direct tuning stage we initialize $s_e = s_d = 1000$ and $s_u = 1$ and then increase the value of s_u by one in each iteration. The mean-square error of the response gradually decreases as s_u increases until $s_u = 39$, at which time we have $s_e = 0.5$ and $s_d = 2.79$. The step response is indicated in Fig. 9 by the dashed line. Using the above result, we begin to perform micro-tuning of SFs in the indirect tuning stage. Note that the plant sensitivity y_u is calculated by numerical approximation. We set $\eta_i = \alpha_i = 0.001$ for $i = 1$ to 3 and perform supervised learning. After 50 learning epochs, we obtain the desired system response shown in Fig. 9 by the solid line. The respective learning trajectories of s_e, s_d, and s_u in the 50th learning epoch are shown in Fig. 10.

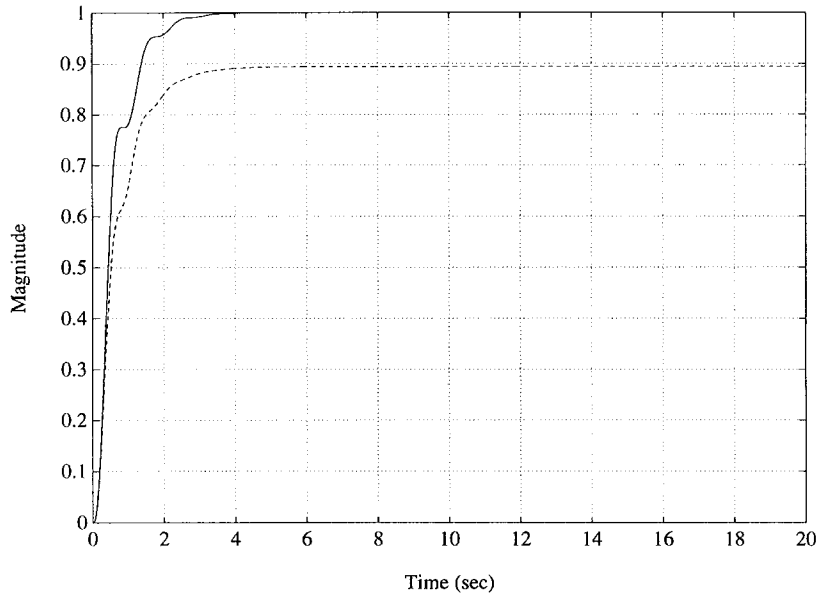


Fig. 8. Step response of the process with $s_u = 19$ in Example 3. The dashed line indicates the system response before steady-state error compensation ($s_e = 1.0$ and $s_d = 34.229$); the solid line indicates the response after steady-state error compensation ($s_e = 1.0$ and $s_d = 29.492$).

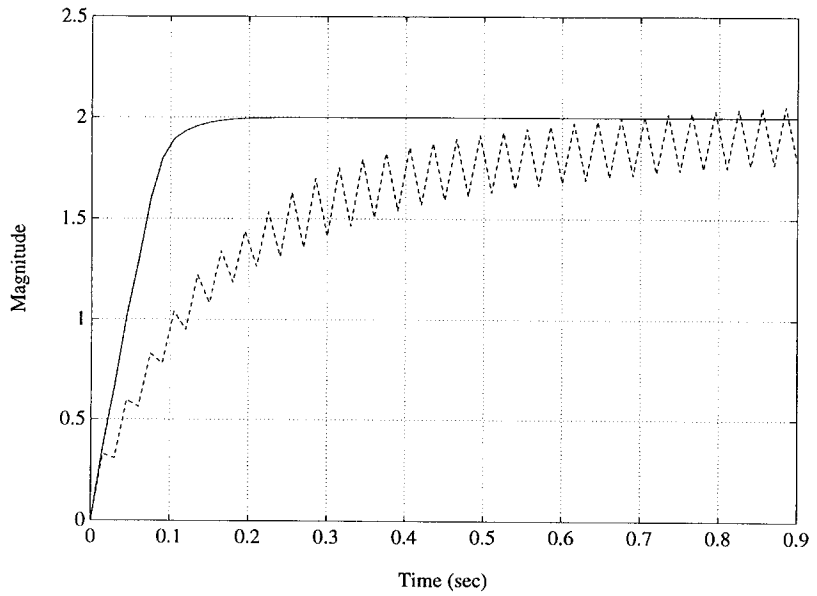
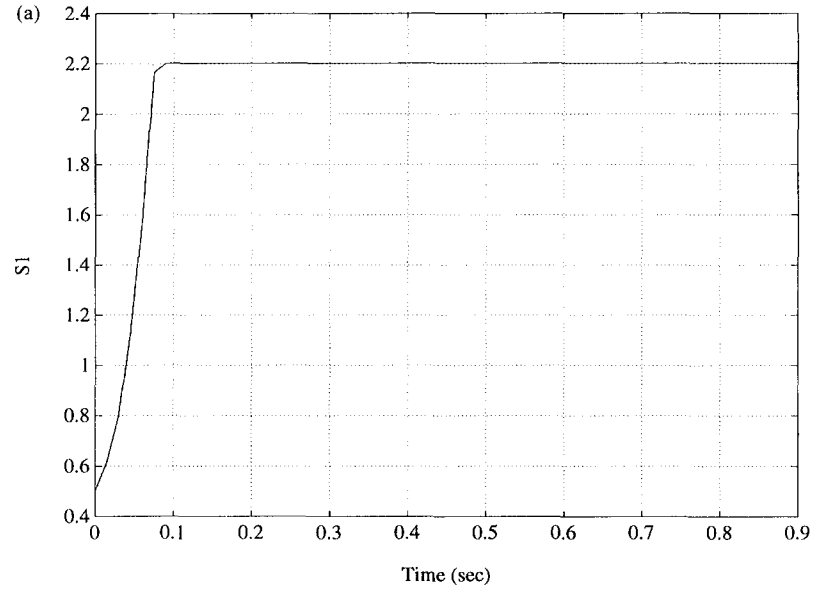


Fig. 9. Final system response for Example 4. The response after direct tuning stage ($s_e = 0.5$, $s_d = 2.79$, and $s_u = 39$) is indicated by the dashed line, that after the indirect tuning stage indicated by the solid line.



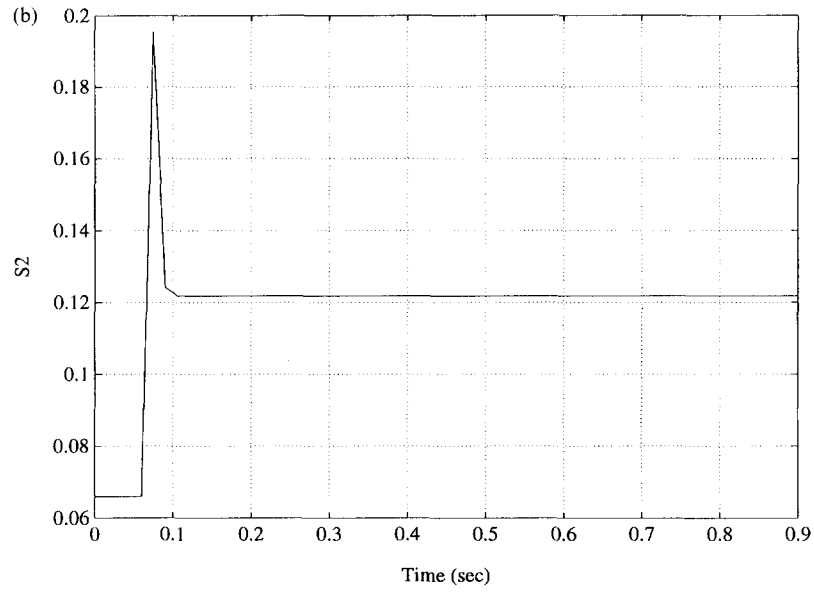


Fig. 10. The learning trajectories of the scaling factors in Example 4. (a) s_e, (b) s_d, and (c) s_i

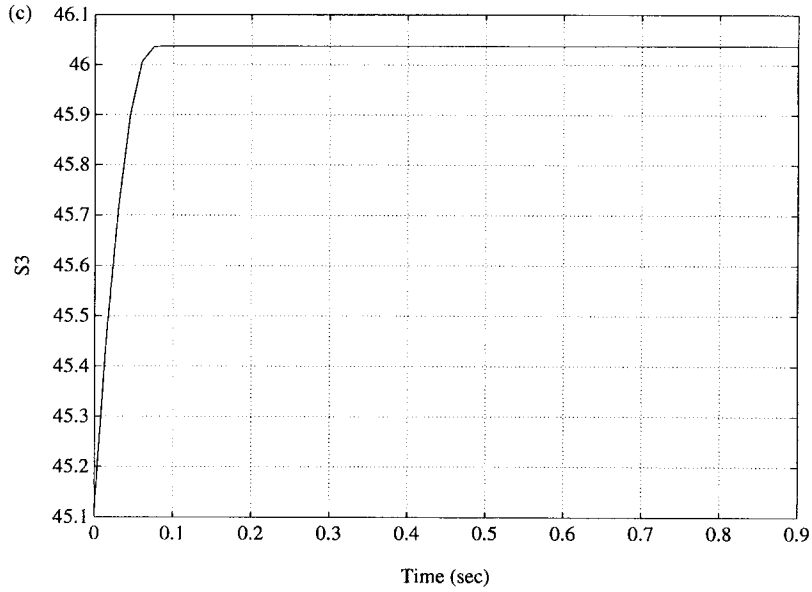


Fig. 10. Continued.

6. Conclusion

In this paper, we have described a PD-like self-tuning fuzzy controller (STFC) based on the tuning of scaling factors. We have solved the steady-state error problem in a PD-like FLC system by a simple but efficient method. Furthermore, to avoid tuning the SFs through trial-and-error, we have proposed a systematic two-stage tuning method. In the direct tuning stage, the proposed STFC automatically detects the operating ranges of the input variables and then adjusts them. In the indirect tuning stage the design results of the direct tuning stage are used to adaptively fine-tune the SFs if the desired result was not obtained in the previous stage. Simulation results for linear time-invariant systems and a nonlinear unstable system show the proposed technique produces zero steady-state error responses very quickly without overshoot or oscillatory behavior.

References

[1] M. Braae and D.A. Rutherford, Selection of parameters for a fuzzy logic controller, *Fuzzy Sets and Systems* **2** (1979) 185–199.

[2] Y.C. Chen and C.C. Teng, A model reference control structure using a fuzzy neural network, *Fuzzy Sets and Systems* **73** (3) (1995) 291–312.

[3] C.H. Chou and H.C. Lu, Real-time fuzzy controller design for hydraulic servo system, *Int. J. Comput. Ind.* **22** (1993) 129–142.

[4] C.H. Chou and H.C. Lu, A heuristic self-tuning fuzzy controller, *Fuzzy Sets and Systems* **61** (1994) 249–264.

[5] S. Daley and K.F. Gill, Comparsion of a fuzzy logic controller with a P+D control law, *Trans. ASME J. Dyn. Systems, Measurement Control* **111** (1989) 128–137.

[6] W.C. Daugherity, B. Rathakrishnan and J. Yen, Performance evaluation of a self-tuning fuzzy controller, *Proc. 1st IEEE Internat. Conf. on Fuzzy Systems*, San Diego (1992) 389–397.

[7] D.P. Filev and R.R. Yager, On the analysis of fuzzy logic controllers, *Fuzzy Sets and Systems* **68** (1994) 39–66.

[8] H.B. Gürocak and A. de Sam Lazaro, A fine tuning method for fuzzy logic rule bases, *Fuzzy Sets and Systems* **67** (1994) 147–161.

[9] C.J. Harris, C.G. Moore and M. Brown, *Intelligent Control: Aspects of Fuzzy Logic and Neural Nets* (World Scientific, Singapore, 1993).

[10] J.S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, *IEEE Trans. Systems Man Cybernet.* **23** (3) (1993) 665–684.

[11] C.T. Lin and C.S.G. Lee, Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Comput.* **40** (12) (1993) 1320–1336.

[12] M. Maeda and S. Murakami, A self-tuning fuzzy controller, *Fuzzy Sets and Systems* **51** (1992) 29–40.
[13] E.H. Mamdani and S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, *Int. J. Man-Machine Studies* **8** (1975) 1–13.
[14] K.S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Networks* **1** (1) (1990) 4–27.
[15] T.J. Procyk and E.H. Mamdani, A linguistic self organizing process controller, *Automatica* **15** (1979) 15–30.
[16] D.E. Rumelhart and J.L. McClelland (eds.), *Parallel Distributed Processing*, Vol. 1 (MIT Press, Cambridge, MA, 1986).
[17] J.-J.E. Slotine and W. Li, *Applied Nonlinear Control* (Prentice-Hall, Englewood Cliffs, NJ, 1991).
[18] M. Sugeno, An introductory survey of fuzzy control, *Inform. and Sci.* **36** (1985) 59–83.
[19] M. Sugeno, *Industrial Applications of Fuzzy Control* (North-Holland, Amsterdam, 1985).
[20] K.L. Tang and R.J. Mulholland, Comparing fuzzy logic with classical controller designs, *IEEE Trans. Systems Man Cybernet.* **17** (1987) 1085–1087.
[21] L.X. Wang, *Adaptive Fuzzy systems and Control: Design and Stability Analysis* (Prentice Hall, Englewood Cliffs, NJ, 1994).
[22] R.R. Yager and D.P. Filev, *Essentials of Fuzzy Modeling and Control* (Wiley, New York, 1994).
[23] H. Ying, W. Siler and J.J. Buckley, Fuzzy control theory: a nonlinear case, *Automatica* **26**(3) (1990) 513–520.
[24] L.A. Zadeh, Fuzzy sets, *Inform. Control* **8** (1965) 338–353.
[25] L. Zheng, A practical guide to tune of proportional and integral (PI) like fuzzy controller, *Proc. 1st IEEE Internat. Conf. on Fuzzy Systems*, San Diego (1992) 633–640.