Fuzzy Sets and Systems 115 (2000) 327–338

FUZZY sets and systems

A self-tuning fuzzy PI controller[1]

Rajani K. Mudi [a], Nikhil R. Pal [b,*]

[a] *Department of Instrumentation Engineering, Jadavpur University, Salt-Lake Campus, Sector-III, Block LB, Calcutta 700 091, India*
[b] *Electronics & Communication Sciences Unit, Indian Statistical Institute, 203, B.T. Road, Calcutta 700 035, India*

Received March 1997; received in revised form April 1998

Abstract

A robust self-tuning PI-type fuzzy logic controller (FLC) is presented. Depending on the process trend, the output scaling factor (SF) of the controller is modified on-line by an updating factor (α). The value of α is determined from a rule-base defined on error (e) and change of error (Δe) of the controlled variable. The proposed self-tuning controller is designed using a very simple control rule-base and the most natural and unbiased membership functions (MFs) (symmetric triangles with equal base and 50% overlap with neighboring MFs). The proposed scheme is tested for a wide variety of processes including a marginally stable system with different values of dead time. Performance comparison between the conventional PI-type and proposed self-tuning FLCs is made in terms of several performance criteria such as peak overshoot, settling time, rise time, integral absolute error and integral-of-time-multiplied absolute error, in addition to the responses due to step input and load disturbance. Results for various processes show that the proposed FLC outperforms its conventional counterpart in each case.

Keywords: Fuzzy controllers; Self-tuning control; Scaling factor

1. Introduction

In process industries FLCs are becoming increasingly popular. They have been successfully used for a number of difficult processes [20], even sometimes they are proved to be more robust than conventional controllers [16]. A comprehensive review of the design and implementation of FLCs can be found in [2,7]. Various forms of self-tuning and self-organizing FLCs have also been reported [4,11,18,19]. To achieve more improved performance and increased robustness, recently neural networks and genetic algorithms are being used in designing such controllers [1,5,6,9].

Mainly three types (i.e., PI, PD and PID) of FLCs are considered for process control applications. Among them PI-type FLCs are most common and practical. The performance of PI-type FLCs is known to be quite satisfactory for linear first-order systems. But like conventional PI-controllers, performance of PI-type FLCs for higher-order systems, systems with large dead time and also for non-linear systems may be very poor due to large overshoot and excessive oscillation. For example, the peak overshoot in the step response of a system with large dead time, may be too large to be acceptable for many applications. PD-type FLCs are suitable for a limited class of systems [12]. PID-type FLCs are rarely used due to the difficulties associated with the formulation of an efficient rule-base and the tuning of its large number of parameters.

Industrial processes are usually non-linear and higher order systems with considerable dead time, and their parameters may be changed with changes in ambient conditions or with time. Thus, to have a satisfactory control performance the control action should be a non-linear function of e and Δe. In a conventional FLC this non-linearity is tried to be incorporated by a limited number of IF–THEN rules, which may not always be sufficient to generate the necessary control actions. In such a situation controllers with fixed valued SFs and simple MFs may not be enough for achieving the desired control performance. With a view to coping with such limitations many research works on tuning of FLCs have been reported [3,4,10,11,14,17,21] where either the input–output SFs or the definitions of MFs and sometimes the control rules are tuned to achieve the desired control objectives.

The tuning scheme proposed in [14] is an application of the gradient decent method for the simultaneous optimization of fuzzy antecedent and crisp consequent parts. The crisp consequent value, and the centre and width of the triangular input MFs of the controller are tuned iteratively by minimizing the square error between the FLC output and the desired output given by the training data. The tuning process continues until the change in the objective function between two successive iterations becomes sufficiently small. The usefulness of this off-line tuning method is limited due its dependency on the availability of a reliable set of training data. This method may be very good for time-invariant control systems, but not for time-varying systems. A tuning scheme for the parameters of a conventional PID controller, i.e., *proportional gain*, *integral time* and *derivative time* using fuzzy inference mechanism is found in [4]. Here the initial values of the parameters (obtained from Ziegler–Nichols tuning formula) are modified on-line by fuzzy rules defined on e and Δe. Results in [4] show that the controller significantly reduces the overshoot of second order processes with large dead time but at the cost of increased rise time.

Authors in [21] determine the input and output SFs by some empirical relations involving parameters of the identified process model (first-order with dead time). The PI-type FLC in [3] is tuned in two steps. First the input and output SFs are determined from the parameters of the identified first-order plant model. Then the crisp consequent parts are adjusted considering the peak overshoot and rise time as performance measures. Performance of these controllers [3,21] will depend on the accuracy with which the process model is identified. A method for optimal adjustment in the input SF by input–output cross-correlation function is described in [17]. An optimal input SF is found by maximizing the cross-correlation function which is a measure of the statistical dependence between input and output. More importance on the tuning of SFs than MFs or rule-base is given in [10] where the SFs are adjusted mainly through trial and error. A fuzzy rule-based scheme for the adjustment of SFs as well as for the tuning of control rules is proposed by Maeda and Murakami [11]. Here, after tuning of SFs the crisp consequent parts of the control rules are updated in each sampling time with fuzzy rules defined on differences between the desired and achieved performance indices (overshoot and rise time) and the deviation of the actual control response from a predefined target response.

Fuzzy PI controllers with resetting capability [8] are designed to eliminate large overshoot and excessive oscillation caused by the integral operation in a PI controller. Such a controller significantly improves the transient response of second order systems with integrating element. But the authors in [13] have justified and shown that the performance of the controller in [8] with resetting capability is more or less similar to that of a conventional PD-type FLC.

Our search through the literature reveals that, though many authors have tried to improve the tuning methods of FLCs, unlike conventional controllers a standard and systematic method for tuning of FLCs is yet to be developed. Moreover, most of the reported works on FLC tuning is limited only to the first-order model which is too much oversimplified for most practical processes. Developing a generalized tuning method for FLCs is a very difficult task because the computation of the optimal values of tunable parameters needs the required control objectives as well as a fixed model for the controller. In this situation FLC tuning based on experts knowledge rather than mathematical models may possibly be an appropriate step in designing a good controller.

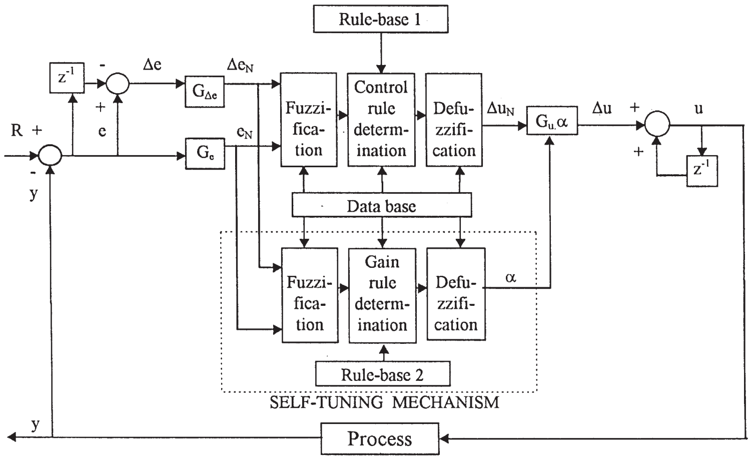


Fig. 1. Block diagram of the proposed controller (STFPIC).

The motivation of this work comes from the observation that, irrespective of the nature of the process to be controlled, a skilled human operator always tries to manipulate the process input (controller output), usually by adjusting the controller gain based on the current process states (generally e and Δe) to get the process optimally controlled. The exact manipulation strategy of a human operator is quite complex in nature and probably no mathematical model can replace it accurately. Here we propose a robust self-tuning PI-type FLC (STFPIC). We have concentrated only on the tuning of output SF (assuming that it is equivalent to the controller gain) due to its strong influence on the performance and stability of the system. The proposed FLC is tuned by dynamically adjusting its output SF in each sampling instance by a gain updating factor (α). The value of α is determined by fuzzy rules defined on e and Δe. The STFPIC is used to conduct simulation analysis for a wide range of different processes including even non-linear and marginally stable systems with different values of dead time. Results show that the proposed self-tuning controller outperforms the conventional PI-type FLC (FPIC) in each case.

The rest of the paper is divided into three sections. In Section 2, we describe the detailed design considerations, i.e., choice of MFs, selection of SFs, formulation of rule-base and the tuning mechanism of STFPIC. In Section 3, based on the simulation results for different processes, the performance of the STFPIC is compared with that of its conventional counterpart, i.e., FPIC. Finally, our conclusions are presented in Section 4.

2. The proposed controller

The block diagram of the STFPIC is shown in Fig. 1. The output SF of the controller is modified by a self-tuning mechanism which is shown by the dotted boundary. Various aspects of the design considerations are discussed in the following subsections.

2.1. Membership functions

All MFs for controller inputs, i.e. e and Δe and incremental change in controller output, i.e., Δu are defined on the common normalized domain $[-1, 1]$, whereas the MFs for α is defined on the normalized domain $[0, 1]$. We use symmetric triangles with equal base and 50% overlap with neighbouring MFs as shown in Fig. 2. This is the most natural and unbiased choice for MFs. Though the MFs in Figs. 2a and 2b are shown separately for the shake of clarity, in actual simulation only the MFs of Fig. 2a are sufficient. Because Fig. 2b can be obtained from Fig. 2a with

$$y = 0.5(x + 1). \tag{1}$$

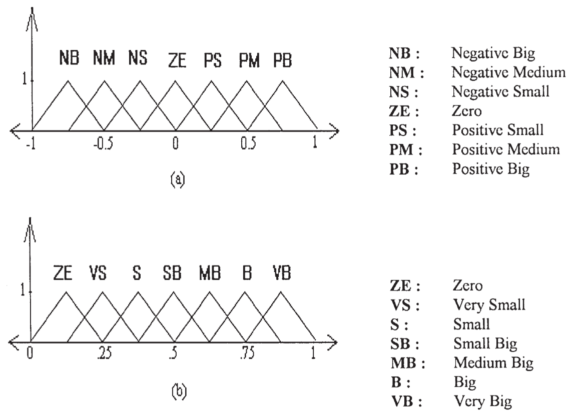
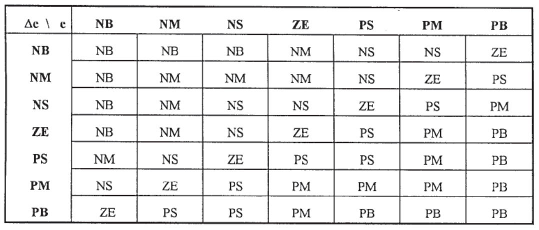


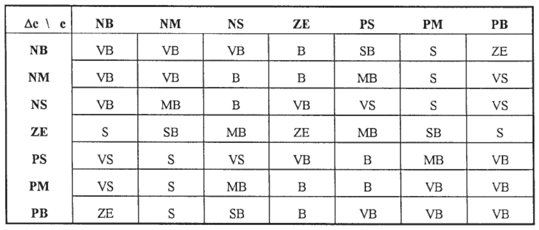
Fig. 2. Membership functions of (a) $E, \Delta E, \Delta U$ and (b) gain updating factor (α).

(a)



Δe \ e	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NM	NS	NS	ZE
NM	NB	NM	NM	NM	NS	ZE	PS
NS	NB	NM	NS	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PS	PM	PB
PM	NS	ZE	PS	PM	PM	PM	PB
PB	ZE	PS	PS	PM	PB	PB	PB

(b)



Δe \ e	NB	NM	NS	ZE	PS	PM	PB
NB	VB	VB	VB	B	SB	S	ZE
NM	VB	VB	B	B	MB	S	VS
NS	VB	MB	B	VB	VS	S	VS
ZE	S	SB	MB	ZE	MB	SB	S
PS	VS	S	VS	VB	B	MB	VB
PM	VS	S	MB	B	B	VB	VB
PB	ZE	S	SB	B	VB	VB	VB

Fig. 3. (a) Fuzzy control rules for computation of Δu. (b) Fuzzy rules for computation of α.

Here x is any point on the horizontal axis of Fig. 2a and y is the corresponding point on Fig. 2b. By this transformation MFs *NB*, *NM*, *NS*, *ZE*, *PS*, *PM* and *PB* of Fig. 2a are mapped to the MFs *ZE*, *VS*, *S*, *SB*, *MB*, *B* and *VB* respectively, of Fig. 2b. Thus the database of the FLC in Fig. 1 contains only the quantitative information about the MFs of Fig. 2a.

2.2. Scaling factors

The MFs for both normalized inputs (e_N and Δe_N) and output (Δu_N) of the controller (Fig. 1) have been defined on the common normalized domain $[-1, 1]$. The relationships between the SFs ($G_e, G_{\Delta e}$ and G_u), and the input and output variables of the STFPIC are as follows:

$$e_N = G_e \,.\, e, \tag{2}$$

$$\Delta e_N = G_{\Delta e} \,.\, \Delta e, \tag{3}$$

$$\Delta u = (\alpha \,.\, G_u) \,.\, \Delta u_N. \tag{4}$$

Unlike FPIC (which uses only G_u) the actual output (Δu) for STFPIC is obtained by using the effective SF ($\alpha.G_u$) as shown in Fig. 1. Selection of suitable values for $G_e, G_{\Delta e}$ and G_u are made based on experts knowledge about the process to be controlled, and sometimes through trial and error. We propose a scheme to compute α on-line using a model independent fuzzy rule-base defined on e and Δe.

2.3. The rule-bases

The operation of a PI-type FLC can be described by

$$u(k) = u(k-1) + \Delta u(k). \tag{5}$$

In Eq. (5) k is the sampling instance and Δu is the incremental change in controller output which is determined by the rules of the form:

R_{PI}: If e is E and Δe is ΔE then Δu is ΔU.

The rule-base for computing Δu is shown in Fig. 3a. This is a very often used rule-base designed with a two dimensional phase plane where the FLC drives the system into the so-called sliding mode [16].

The gain updating factor (α) is calculated using fuzzy rules of the form:

R_α: If e is E and Δe is ΔE then α is α.

The rule-base in Fig. 3b is used for the computation of α. This is designed in conjunction with the rule-base in Fig. 3a. We emphasize here that, the determination of rule-base for α is dependent on the controller rule-base. Fig. 3b is designed to incorporate the following important considerations keeping in view the overall control performance:

(i) To achieve a lower overshoot and reduced settling time but not at the cost of increased rise time the controller gain should be set at a small value when the error is very big. This may be achieved by a rule like, If e is *PB* and Δe is *NS* then α is *VS*. Small gain is essential to avoid the problems associated with the

integral windup (i.e., excessive accumulation of the controller output due to the integral action) to maintain the controller performance within the acceptable limit, specially when the process dead time becomes considerably large.
(ii) Depending on the situation gain around the set-point should be allowed to vary widely to prevent large overshoot and undershoot. For example, the rule, If e is *ZE* and Δe is *NM* then α is *B*, will reduce the overshoot; and large undershoot can be avoided by using a rule of the form, If e is *NS* and Δe is *PS* then α is *VS*. Due to such gain variation the convergence rate of the process output to the set point will be increased, which means the system response will be less oscillatory.
(iii) To improve the control performance under load disturbances, gain around the steady state condition is made sufficiently large (e.g. If e is *NS* and Δe is *ZE* then α is *MB*). But at steady state gain should be very small (e.g. If e is *ZE* and Δe is *ZE* then α is *ZE*).

Note that, the rule-base for α depends on the type of response the control system designer wishes to achieve. For example, the rule-base in Fig. 3b is justified and defined for the controller rule-base in Fig. 3a. And if there is any significant change in the controller rule-base then the rule-base for α may require to be changed accordingly.

2.4. Tuning of the controller

In our scheme the required non-linear controller output ($\Delta u_{\mathrm{STFPIC}}$) is generated by modifying the output of a simple FLC (Δu_{FPIC}) with the updating factor α, i.e.,

$$\Delta u_{\mathrm{STFPIC}} \propto \alpha \,.\, (\Delta u_{\mathrm{FPIC}})$$

or

$$\Delta u_{\mathrm{STFPIC}} = K \,.\, \alpha \,.\, (\Delta u_{\mathrm{FPIC}}), \tag{6}$$

where K is the proportionality constant.

From Eq. (6) we can say that the STFPIC is equivalent to a simple PI-type FLC (FPIC) with a dynamic gain. The value of α is calculated using the rule-base in Fig. 3b which is defined in e and Δe, and derived from the knowledge of control engineering with a view to mimicking an operator's strategy while running a plant.

Fig. 1 shows that the gain of our self-tuning FLC does not remain fixed while the controller is in operation, rather it is modified in each sampling time by the gain updating factor α, depending on the process trend. The reason behind this on-line gain variation is to make the controller respond according to the desired performance specifications. The highly non-linear variation of α with e and Δe is shown in Fig. 4. Fig. 4 reflects the desirable characteristics of α as a function of e and Δe. For example, if error is positive small and change of error is negative medium then there is a possibility of large overshoot, which is not desirable. To avoid this large overshoot, the value of α should be comparatively large. Fig. 4 indeed reflects this. The control surfaces, i.e., controller output (Δu) versus e and Δe for the FPIC and STFPIC are depicted in Figs. 5a and 5b respectively. Careful observation of these two figures reveals that the control surface of the STFPIC is more non-linear as well as smooth than that of FPIC (e.g., observe the third quadrant of both figures). Fig. 5a also indicates that the limited number of IF–THEN rules using simple MFs and fixed valued SFs are not sufficient to produce the necessary control action to achieve the desired performance. In the proposed scheme the controller output (Fig. 5b) is generated by the continuous and non-linear variation of α. The most important point to note is that α is not dependent in any way, on any process parameter. The value of α depends only on the instantaneous process states (e and Δe). Hence the proposed scheme is process independent. The following steps can be used for the tuning of STFPIC.

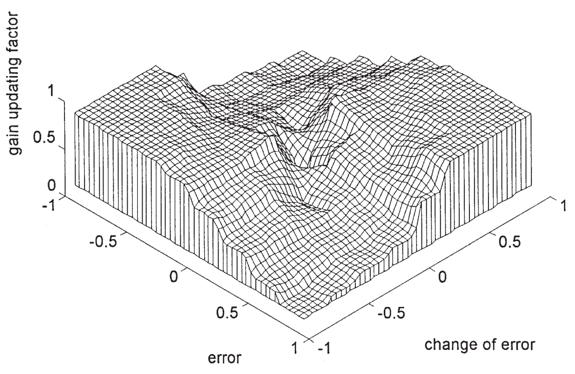
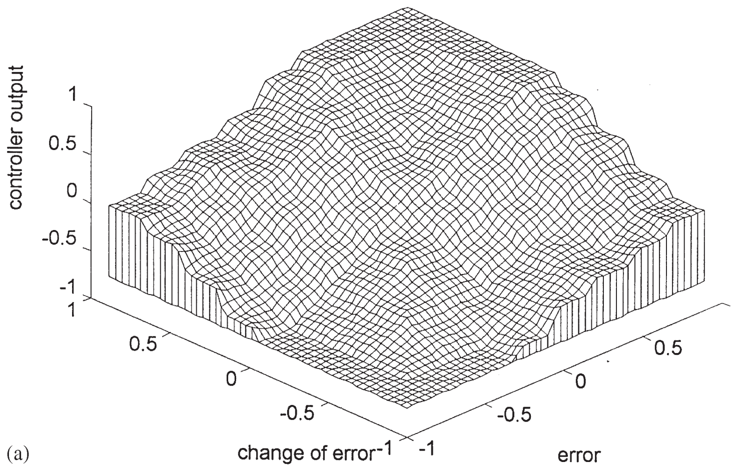


Fig. 4. Variation of gain updating factor (α) with error (e) and change of error (Δe).

Step 1: Tune the SFs of the STFPIC assuming $\alpha = 1$ (i.e., FPIC) for a given process. In doing so first, G_e should be selected in such a way that the normalized

(a)


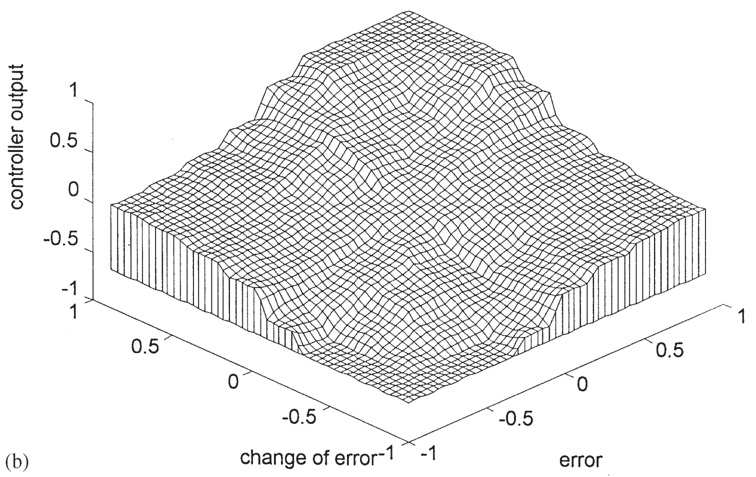
(b)


Fig. 5. (a) Control surface of the FPIC. (b) Control surface of the STFPIC.

error (e_N) almost covers the entire domain $[-1, 1]$ to make efficient use of the rule-bases. Then $G_{\Delta e}$ and G_u are to be tuned to make the transient response of the system reasonably good. At the end of this step, we get a good controller without self-tuning (FPIC) and then this controller becomes the starting point for the STFPIC in Step 2.

Step 2: Set the output SF (G_u) of the STFPIC *three times* greater than that of FPIC, keeping the values of G_e and $G_{\Delta e}$ same as those of FPIC obtained in Step 1. Observe that in this step $\alpha \neq 1$, but obtained from the rule-base in Fig. 3b. Make a small adjustment for G_u of the STFPIC, if necessary to realize almost the same rise time as that of the FPIC obtained in Step 1. Note that, in order to maintain the same rise time for STFPIC as that of FPIC, this further adjustment of G_u is required because of the factor α which lies in $(0, 1]$. Here we should mention that, instead of changing G_u, the purpose of this step can be equally served if we enhance α *three times* keeping the values of G_e, $G_{\Delta e}$ and G_u fixed at the values used for FPIC. This is better understood from Eqn. (6) with $K = 3$ in conjunction with Fig. 1.

Step 3: Fine tune the rules for α based on the considerations described in Section 2.3, depending on the type of response wanted to achieve. For example, if we want to further reduce the overshoot at the cost of increased rise time then up to the medium values of e the value of α should be kept very small and this can be achieved by a rule of the form, If e is *PM* and Δe is *NS* then α is *VS* (not *S* as shown in Fig. 3b).

To get a better insight into the proposed scheme let us consider a conventional PI controller and a FLC under the Takagi–Sugeno (TS) model. The incremental form of a conventional PI controller can be described as

$$\Delta u_c = K_p[\Delta e + (\Delta t/T_I)\cdot e], \tag{7}$$

where Δu_c is the incremental change in output, K_p is the *proportional gain*, T_I is the *integral time* and Δt is the *sampling time* of the controller. Now for the TS model, suppose the ith rule takes the form
R_i: If e is E and Δe is ΔE then $\Delta u_i = a_{i1}\cdot\Delta e + a_{i2}\cdot e$. Then the resultant output for given $(e, \Delta e)$ can be expressed as

$$\Delta u_F = \sum_i \mu_i\cdot(a_{i1}\cdot\Delta e + a_{i2}\cdot e)\Big/\sum_i \mu_i, \tag{8}$$

where μ_i is the firing strength of the ith rule. Eq. (8) can be rewritten as

$$\Delta u_F = \left[\left(\sum_i w_i\cdot a_{i1}\right)\cdot\Delta e + \left(\sum_i w_i\cdot a_{i2}\right)\cdot e\right],$$
$$\text{when } w_i = \left(\mu_i\Big/\sum_i \mu_i\right)$$

or

$$\Delta u_F = \left(\sum_i w_i\cdot a_{i1}\right)\left[\Delta e + \left(\left(\sum_i w_i\cdot a_{i2}\right)\Big/\left(\sum_i w_i\cdot a_{i1}\right)\right)\cdot e\right] \tag{9}$$

Writing $K_{PF} = \sum_i w_i\cdot a_{i1}$, and $T_{IF} = [((\sum_i w_i\cdot a_{i1})/(\sum_i w_i\cdot a_{i2}))\cdot\Delta t]$ Eq. (9) becomes

$$\Delta u_F = K_{PF}[\Delta e + (\Delta t/T_{IF})\cdot e]. \tag{10}$$

Comparing Eq. (10) with Eq. (7) we see that Eq. (10) has exactly the same form as that of Eq. (7). Thus K_{PF} and T_{IF} may be considered the respective fuzzy counterparts of K_p and T_I associated with the non-fuzzy PI controller described by Eq. (7). Since $w_i = f(e, \Delta e)$, and a_{i1} and a_{i2} are constants for the given rule-base, both K_{PF}, and T_{IF} are implicit functions of e and Δe. From Eqns. (7) and (10) we see that contrary to the non-fuzzy PI controller the output of the fuzzy PI controller is generally non-linear because both the *proportional gain* (K_{PF}) and *integral time* (T_{IF}) change with process states (i.e., e and Δe). But the non-linear output generated by the FLC with fixed valued SFs, simple MFs (regular shapes such as symmetric triangle or trapezoid with equal base) and rule-base (like in Fig. 3a designed in sliding mode principle) may not be adequate to ensure good performance for non-linear and higher order systems, specially systems with integrating element (i.e., marginally stable systems) or large dead time. In a conventional FLC (non-adaptive) this shortcoming may be eliminated by (i) changing the definitions of MFs, their degree of overlap, (ii) increasing the number of MFs or (iii) modifying the rule-base. Certainly these will make the FLC design very difficult as till-date there is no standard and systematic method to adjust them in order to achieve some desired performance. The proposed scheme is a simple attempt to realize the desired level of non-linearity into the system through a gain updating factor, computed using a rule-base defined in terms of e and Δe.

The proposed scheme uses two rule-bases both defined on e and Δe. This raises a natural question: can we combine them? Probably the answer is ‘yes’. One might think, this can be done by defining a different linguistic value for the controller output for each distinct combination of linguistic values for Δu and α as demanded by Fig. 3a and 3b. To make it clear, when e is, say, *NM*, and Δe is *PS* then Δu is *NS* (Fig. 3a) and α is *S* (Fig. 3b), so the pair (*NS*, *S*) gives a distinct combination. And we can assume a rule of the form: If e is *NM* and Δe is *PS* then Δu is *NSS*, where *NSS* is a new linguistic value. But this approach will have several problems. These linguistic values may (usually will) not have descent shapes like triangle etc. Their semantic interpretation as well as representation for implementation would be very difficult. Moreover, the highly non-linear controller output is not only dependent on this (*NS*, *S*) combination but also on its neighboring rules in both Fig. 3a and 3b. Another alternative may be to use system identification (SI) techniques through exploratory data analysis when the controller outputs for different e, Δe combinations are available. But identification of the combined system would be much more difficult than that of two separate sub-systems defined by the two rule-bases.

3. Results

The performance of the STFPIC is compared with the FPIC for different types of processes. For a clear comparison between the conventional and self-tuning FLCs several performance measures such as, peak overshoot (%OS), settling time (t_s), rise time (t_r), integral absolute error (IAE) and integral-of-time-multiplied absolute error (ITAE) [15] are used. The comparative performance of the two controllers are tabulated for each process with different values of dead time. Since peak overshoot and rise time conflict each other one cannot reduce them simultaneously. If one of them is made smaller, the other usually becomes larger. For an easy performance comparison, rise times for both FPIC and STFPIC are maintained almost at the same value. For examining the transient as well as steady state behavior of the proposed controller each process is tested with both step set-point change and load disturbance. To establish the robustness of the proposed scheme we use the same rule-bases (Fig. 3) and MFs (Fig. 2) for all processes with different values of dead time. In all cases Mamdani type inferencing [18] and height method [2] of defuzzification are used. We have also used the centre of sums method [2] of defuzzification but could not find any noticeable difference in control performance with these two defuzzification methods. We preferred the height method of defuzzification as it is very simple and faster algorithm which is an important consideration as far as real-time implementation is concerned. In the simulation fourth-order Runge-Kutta method is used for the numerical integration with an interval of 0.1 s for all processes. In all figures (Figs. 6–9), solid curves represent the responses due to the STFPIC while dashed (- -) curves depict the responses due to the FPIC. Next we present the results for different processes.

3.1. First order process

One such process is described by the following transfer function ($G_p(s)$)

$$G_p(s) = \mathrm{e}^{-LS}/(5s+1). \tag{11}$$

Three different values of process dead time (L), i.e., $L = 0.1$, $L = 0.2$ and $L = 0.3$ are considered for (11). Fig. 6 shows the response characteristics of (11) with $L = 0.1$ for both FPIC and STFPIC due to step input and load disturbance introduced at $t = 12.5$ s. Various performance indices for step response of (11) with different values of dead time are included in Table 1. From Fig. 6 we see that, STFPIC shows remarkably improved performance both in transient and steady state conditions. Results in Table 1 reveal that the performance of STFPIC is consistently better for all the three values of L than FPIC.

3.2. Second order process

Let us consider the second order process

$$G_p(s) = 5\mathrm{e}^{-LS}/(5s^2+5s+1). \tag{12}$$

For this process also we have considered three different values of L (i.e., 0, 0.1 and 0.3). Fig. 7 shows the responses of (12) under STFPIC and FPIC due to both step and load disturbances for $L = 0.1$. In Fig. 7 the load disturbance is applied at $t = 39$ s. The performance indices for different values of L are listed in Table 2. From Fig. 7 and Table 2 we again find that in each case the proposed controller outperforms its conventional counterpart.

3.3. Second order process (marginally stable system)

$$G_p(s) = \mathrm{e}^{-LS}/(s(1.5s+1)). \tag{13}$$

For this marginally stable system, presence of dead time makes it very difficult to control. Transient responses of this system with $L = 0$ for STFPIC and FPIC due to both step set-point change and load disturbance introduced at $t = 60$ s are shown in Fig. 8. Table 3, which includes various values of performance indices for different L ($L = 0$, 0.1, 0.2 and 0.3)

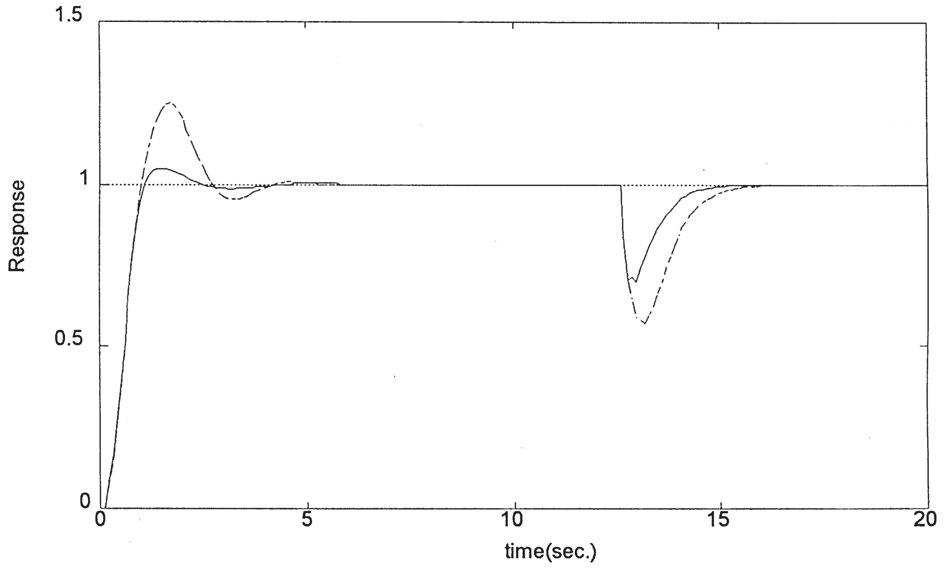


Fig. 6. Responses of the first-order process in (11) with $L = 0.1$.

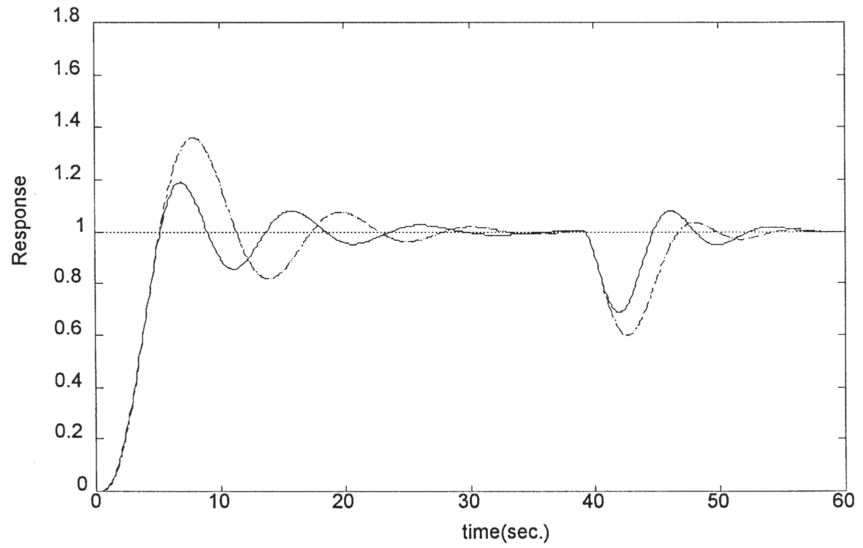


Fig. 7. Responses of the second-order process in (12) with $L = 0.1$.

shows that in each case STFPIC has remarkably reduced the %OS and t_s. Table 3 reveals that due to the self-tuning mechanism the performance of STFPIC remains within the acceptable limit even when such a difficult process (13) is associated with comparatively large dead time.

3.4. Non-linear process

Finally, we consider the non-linear process described by

$$dy/dt + 2y - y^2 = u(t - L). \tag{14}$$

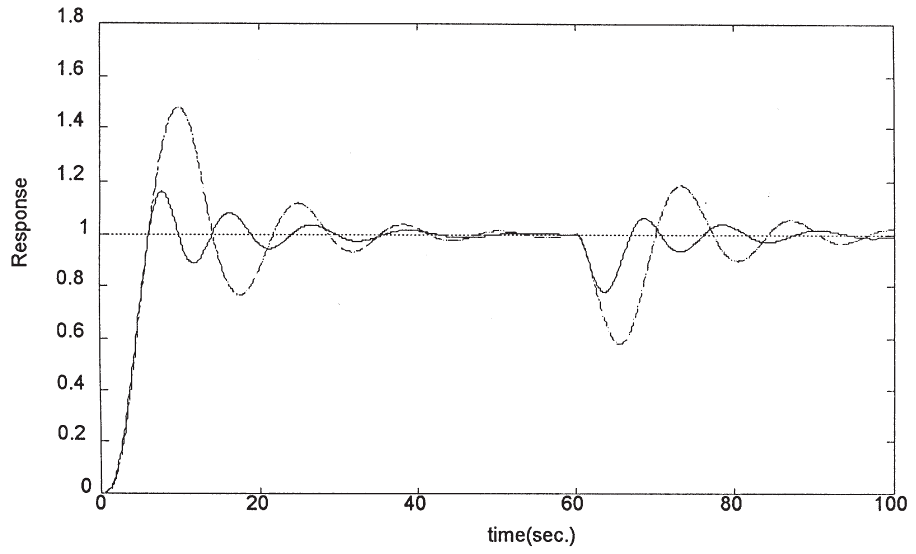


Fig. 8. Responses of the second-order (marginally stable) process in (13) with $L = 0$.

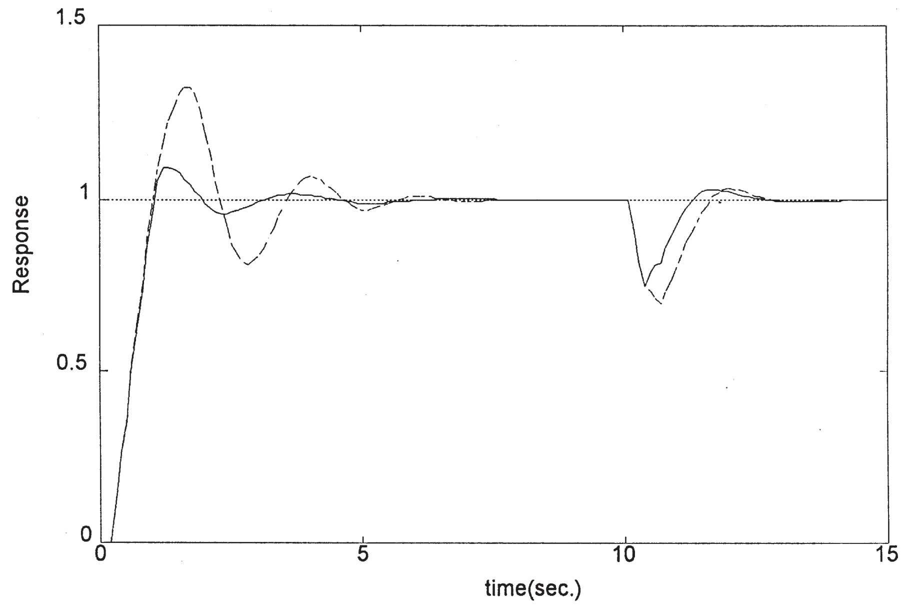


Fig. 9. Responses of the first-order non-linear process in (14) with $L = 0.2$.

Fig. 9 shows the performance comparison of STFPIC and FPIC for the process in (14) with $L = 0.2$ under step input and load disturbance applied at $t = 10$ s. And Table 4 provides the various performance indices for step response of (14) with different values of dead time (i.e., $L = 0.1$, 0.2 and 0.3). From the results (Table 4 and Fig. 9) here also we find that STFPIC shows much improved performance than FPIC for all three values of dead time.

Table 1
Performance analysis for the first-order process in (11)

L	FLC	%OS	$t_s(s)$	$t_r(s)$	ITAE	IAE
0.1	FPIC	25.2	2.7	1.0	0.75	0.84
	STFPIC	5.1	1.1	1.0	0.32	0.62
0.2	FPIC	37.4	4.5	1.1	1.70	1.22
	STFPIC	10.9	2.2	1.1	0.59	0.78
0.3	FPIC	47.3	6.5	1.2	3.29	1.66
	STFPIC	19.8	3.6	1.2	1.17	1.08

Table 2
Performance analysis for the second-order process in (12)

L	FLC	%OS	$t_s(s)$	$t_r(s)$	ITAE	IAE
0	FPIC	34.3	18.5	4.5	28.19	4.76
	STFPIC	16.1	14.8	4.5	18.35	3.76
0.1	FPIC	36.1	21.2	5.0	36.22	5.42
	STFPIC	19.1	17.2	5.0	24.20	4.37
0.3	FPIC	40.4	23.9	5.4	54.50	6.59
	STFPIC	26.6	21.7	5.4	42.48	5.48

Table 3
Performance analysis for the second-order process (marginally stable in (13)

L	FLC	%OS	$t_s(s)$	$t_r(s)$	ITAE	IAE
0	FPIC	48.0	33.0	6.0	80.3	7.91
	STFPIC	16.1	17.7	6.0	32.5	4.93
0.1	FPIC	53.9	39.4	6.6	130.5	10.06
	STFPIC	20.2	21.0	6.6	43.5	5.70
0.2	FPIC	58.2	51.7	7.0	188.7	12.16
	STFPIC	22.9	28.4	7.0	52.86	6.29
0.3	FPIC	61.1	54.7	7.2	227.2	13.44
	STFPIC	25.2	30.6	7.2	62.5	6.80

Table 4
Performance analysis for First-order non-linear process in (14)

L	FLC	%OS	$t_s(s)$	$t_r(s)$	ITAE	IAE
0.1	FPIC	14.3	2.6	0.8	0.27	0.56
	STFPIC	3.5	1.7	0.8	0.19	0.50
0.2	FPIC	32.2	5.4	1.0	1.25	1.02
	STFPIC	9.48	2.9	1.0	0.49	0.69
0.3	FPIC	51.1	10.5	1.2	5.73	2.20
	STFPIC	21.2	6.7	1.2	1.41	1.00

The preceding results for different types of processes reveal that, just like non-fuzzy PI controllers, conventional fuzzy PI controllers also are not suitable for higher order and non-linear processes. This fact is clearly justified from the various performance indices listed in Tables 3 and 4 for the second-order process with integrating element in (13) and the non-linear process in (14), respectively. But the proposed self-tuning controller, i.e., STFPIC shows excellent performance in such situations.

4. Conclusion

We proposed a robust self-tuning PI-type fuzzy logic controller. The proposed controller was tuned on-line by adjusting its output SF depending on the process trend. The output SF was dynamically modified by a single parameter α which was determined by fuzzy rules defined on e and Δe. The most important feature of the proposed scheme is that it does not depend on the process being controlled (i.e., process independent). The proposed controller was used to conduct simulation analysis for a wide range of different processes, and in each case the performance of STFPIC with respect to both transient and steady state conditions was compared with that of FPIC. From the results for different processes, STFPIC was found to exhibit remarkably improved performance. Robustness of the proposed scheme was established by using the same rule-bases and MFs for all processes with different values of dead time.

This same technique may possibly be applied for the tuning of PD-type FLCs and also for the tuning of input or both input and output SFs simultaneously which may lead to achieve fuzzy controllers with more improved performances. Present investigation used 49 rules for the tuning of output SF. We believe, that the size of rule-base can be significantly reduced. We are currently investigating on these possibilities.

References

[1] C.L. Chen, W.C. Chen, Fuzzy controller design by using neural network techniques, IEEE Trans. Fuzzy Systems 2 (3) (1994) 235–244.

[2] D. Dirankov, H. Hellendron, M. Reinfrank, An Introduction to Fuzzy Control, Springer, New York, 1993.

[3] S. Hayashi, Auto-tuning fuzzy PI controller, Proc. IFSA, 1991, pp. 41–44.

[4] S.Z. He, S. Tan, F.L. Xu, P.Z. Wang, Fuzzy self-tuning of PID controller, Fuzzy Sets and Systems 56 (1993) 37–46.

[5] A. Homaifar, E. McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, IEEE Trans. Fuzzy Systems 3 (2) (1995) 129–139.

[6] C.L. Karr, E.J. Gentry, Fuzzy control of PH using genetic algorithm, IEEE Trans. Fuzzy Systems 1 (1) (1993) 46–53.

[7] C.C. Lee, Fuzzy logic in control systems: fuzzy logic controller – part I, II, IEEE Trans. Systems Man Cybernet. 20 (2) (1990) 404–435.

[8] J. Lee, On methods for improving performance of PI-type fuzzy logic controllers, IEEE Trans. Fuzzy Systems 1 (4) (1993) 298–301.

[9] W. Li, A method for design of a hybrid neuro-fuzzy control system based on behaviour modeling, IEEE Trans. Fuzzy Systems 5 (1) (1997) 128–137.

[10] H.X. Li, H.B. Gatland, Conventional fuzzy control and its enhancement, IEEE Trans. Systems Man Cybernet. 26 (5) (1996) 791–797.

[11] M. Maeda, S. Murakami, A self-tuning fuzzy controller, Fuzzy Sets and Systems 51 (1992) 29–40.

[12] H.A. Malki, H. Li, G. Chen, New design and stability analysis of fuzzy proportional-derivative control systems, IEEE Trans. Fuzzy Systems 2 (4) (1994) 245–254.

[13] R. Mudi, N.R. Pal, A note on fuzzy PI-type controllers with resetting action, Fuzzy Sets and Systems, accepted.

[14] H. Nomura, I. Hayashi, N. Wakami, A self-tuning method of fuzzy control by decent method, Proc. IFSA, 1991, pp. 155–158.

[15] K. Ogata, Modern Control Engineering, Prentice-Hall, Englewood Cliffs, NJ, 1970.

[16] R. Palm, Sliding mode fuzzy control, Proc. Fuzz IEEE, San Diego, 1992, pp. 519–526.

[17] R. Palm, Scaling of fuzzy controller using the cross-correlation, IEEE Trans. Fuzzy Syst. 3 (1) (1995) 116–123.

[18] T.J. Procyk, E.H. Mamdani, A linguistic self-organising process controller, Automatica 15 (1) (1979) 53–65.

[19] S. Shao, Fuzzy self-organing control and its application for dynamical systems, Fuzzy Sets and Systems 26 (1988) 151–164.

[20] M. Sugeno, Industrial Applications of Fuzzy Control, Elsevier Science, Amsterdam, 1985.

[21] M. Yoshida, Y. Tsutsumi, T. Ishida, Gain tuning method for design of fuzzy control systems, Proc. Internat. Conf. on Fuzzy Logic and Neural Networks, 1990, pp. 405–408.