ELSEVIER

Journal of Systems Architecture 47 (2001) 663–673

JOURNAL OF SYSTEMS ARCHITECTURE

www.elsevier.com/locate/sysarc

Autotuning a PID controller: A fuzzy-genetic approach

R. Bandyopadhyay [a], U.K. Chakraborty [b,*], D. Patranabis [a]

[a] *Department of Instrumentation Engineering, Jadavpur University, Salt Lake Campus, Calcutta 700 098, India*
[b] *Department of Mathematics and Computer Science, University of Missouri, St. Louis, MO 63121, USA*

Abstract

A new method for tuning the parameters of the proportional integral derivative (PID) controller is presented in this paper. The technique adopted in this proposition is based on the format of dead-beat control. Fuzzy inference mechanism has been used here for predicting the future values of the controller output while crisp consequent values of the rulebase of the Takagi–Sugeno model are optimized using a genetic algorithm. The proposition is an extension of the work in R. Bandyopadhyay, D. Patranabis (A new autotuning algorithm for PID controllers using dead-beat format, ISA Trans., accepted for publication) where the rulebase was prepared based on the knowledge of process experts. The use of genetic algorithm for optimizing the crisp values of the rulebase has considerably improved the performance of the PID autotuner. The proposed algorithm seems to be a complete and generalized PID autotuner as can be seen from the simulated and experimental results. In all the cases the method shows substantial improvement over the controller tuned with Ziegler–Nichols formula and the PID controller proposed in (loc cit). © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Control system design; PID control; Autotuner; Dead-beat control; Genetic algorithm; Fuzzy logic

1. Introduction

The proportional integral derivative (PID) controller is by far the most commonly used controller in process control applications. Almost all control problems can be solved by these controllers and they are found in large numbers in all process industries. They have survived many changes in technology. The early controllers were based on relays and pneumatic systems. These systems were then replaced by electronic circuits and lately, by microprocessors. The PID controllers perform several important functions, two very important ones being the elimination of steady-state offset and anticipation of deviation and generation of adequate corrective signals through the derivative action. Together with combinational logic, sequential machines and simple function blocks, these PID controllers are increasingly being used to build the automation system for industries. All the modern instrumentation and control systems like Programmable Logic Controllers and Distributed Control Systems provide a block for the PID controller.

In the early development of automatic control, PID control generated much interest. But unfortunately for a long time, researchers paid very little attention to it. The main reason behind this is the practical difficulty in tuning the three parameters by trial and error. After the widespread application of microprocessors, there has been a resurgence of interest in PID control. Even though

* Corresponding author.
E-mail address: uday@cs.umsl.edu (U.K. Chakraborty).

these PID controllers are very common and well known, they are often not tuned properly resulting in poor control quality. Since almost all PID controllers are now implemented in software, there is opportune scope to incorporate complex algorithms in these controllers. Autotuning is one such feature which is now being extensively used in commercially available PID controllers [1,2]. Autotuners offer several advantages. Since automatic tuning is faster than manual tuning, the commissioning time for installation of new processes can be decreased. In manual tuning, trial and error is the most common procedure; otherwise a step response method or ultimate cycle method of Ziegler and Nichols [3] is used. Automatic tuning also ensures systematic tuning, even for the simplest control loops.

The programmable feature of currently available autotuners offers numerous possibilities for improving the performance of the control loops. In a previous work [4], an attempt was made to improve the performance of the proportional integral type of controllers by incorporating an auto-tuning strategy. Subsequently, the algorithm was extended [5] to include the derivative term to make a complete PID autotuner so that disturbances with relatively high rate and reasonable magnitude could be taken care of. In the present work, the identification mechanism for the self-tuning strategy is implicit, and error and its derivative have been taken into consideration for predicting the parameters of the controller. The objective here is to make the error zero at the next sampling instant as dictated by the format of dead-beat control [6] and the predicted value of the controller output is obtained by a fuzzy inference mechanism. The fuzzy inference mechanism is based on the Takagi–Sugeno model [7] and the rulebase is derived with the help of a genetic algorithm [8,9]. The proposed approach of autotuning is similar in spirit to the approaches of He et al. [10] and Xu et al. [11] with the conventional PID control mechanism replaced by soft computing techniques. The advantage of such an approach is that since the controller output is not directly obtained from the inference engine, the method is less sensitive to the knowledge of process experts.

The controller parameters are set to their initial values when the process is in steady state. As soon as the error exceeds a predefined limit (the limit is set depending on measurement noise), the controller parameters are varied in such a way that the process reaches steady state minimizing the integral absolute error (IAE). Although the steady-state values are not obtained on-line in this study, the on-line identification algorithm employing the relay-feedback method [12] can be incorporated in the proposed autotuner.

The proposed controller tuning strategy has been applied to several simulated process models and results for the first-order plus dead-time (FOPDT) and second-order plus dead-time (SOPDT) process models are presented. Comparisons with the conventionally tuned Ziegler–Nichols controller as well as the previously designed PID controller [5] show marked improvement for the proposed autotuner taking IAE as the performance criterion. The method has also been tested on a laboratory-type temperature rig PT-326 of FEEDBACK. Superior performance of the proposed algorithm has been demonstrated through simulation as well as kit level test in the laboratory.

2. The tuning procedure

As mentioned earlier, the algorithm has been developed with the objective of achieving dead-beat control and the controller parameters are tuned in such a way that the error at the next sampling instant becomes zero. The basic equation of a PID controller in discrete domain is given by

$$m_n = K_{\mathrm{C}_{n-1}} \cdot e_n + \frac{T}{2}\sum_{i=1}^{n}\frac{K_{\mathrm{C}_{i-1}}}{T_{\mathrm{R}_{i-1}}}(e_i + e_{j-1}) + \frac{K_{\mathrm{C}_{n-1}} T_{\mathrm{D}_{n-1}}}{T}(e_n - e_{n-1}) + m_0, \tag{1}$$

where m is the manipulated variable, the controller output; K_{C} is the proportional action gain; e is the error; T is the sampling time; T_{R} is the reset time; T_{D} is the derivative time and suffixes denote the sampling instants.

At the $(n+1)$th instant, Eq. (1) is modified as

$$m_{n+1} = K_{\mathrm{C}_n} \cdot e_{n+1} + \frac{T}{2} \sum_{i=1}^{n+1} \frac{K_{\mathrm{C}_{i-1}}}{T_{\mathrm{R}_{i-1}}} (e_i + e_{j-1}) + \frac{K_{\mathrm{C}_n} T_{\mathrm{D}_n}}{T} (e_{n+1} - e_n) + m_0. \tag{2}$$

Subtracting Eq. (1) from Eq. (2), the velocity form algorithm of the PID controller is derived:

$$\begin{aligned} m_{n+1} - m_n = {} & K_{\mathrm{C}_n} \cdot e_{n+1} - K_{\mathrm{C}_{n-1}} \cdot e_n \\ & + \frac{T}{2} \left\{ \frac{K_{\mathrm{C}_n}}{T_{\mathrm{R}_n}} (e_{n+1} + e_n) \right\} \\ & + \frac{K_{\mathrm{C}_n} \cdot T_{\mathrm{D}_n}}{T} (e_{n+1} - e_n) \\ & - \frac{K_{\mathrm{C}_{n-1}} \cdot T_{\mathrm{D}_{n-1}}}{T} (e_n - e_{n-1}). \end{aligned} \tag{3}$$

Dead-beat control requires e_{n+1} to be zero and the proposed controller tuning method modifies the controller parameters to achieve this goal. Thus making $e_{n+1} = 0$, Eq. (3) is modified as

$$K_{\mathrm{C}_n} \left(\frac{T}{2T_{\mathrm{R}_n}} - \frac{T_{\mathrm{D}_n}}{T} \right) = \left\{ m_{n+1} - m_n + K_{\mathrm{C}_{n-1}} \cdot e_n + \frac{K_{\mathrm{C}_{n-1}} \cdot T_{D_{n-1}}}{T} (e_n - e_{n-1}) \right\} \Big/ e_n. \tag{4}$$

The right-hand side of Eq. (4) is to be evaluated for obtaining the new values of $K_{\mathrm{C}}, T_{\mathrm{R}}$ and T_{D}. All the quantities in the above equation, except m_{n+1}, are known at the nth sampling instant. The evaluation of m_{n+1}, however, is done through a rule-based prediction mechanism as was done in [5]. The procedure, however, is a little more elaborate because of the additional parameter T_{D}. The technique is explained in the following section.

2.1. Prediction mechanism of controller output (m_{n+1}) using fuzzy logic

The linear extrapolation technique has been adopted for the evaluation of the predicted controller output and is given as

$$m_{n+1} = \alpha \cdot m_n. \tag{5}$$

The parameter α may be termed as the predictive scaling factor and is chosen in such a way that the predicted value of the controller output always leads the process variable to the desired state via optimal trajectory. Even though the objective of the proposed algorithm is to achieve dead-beat control, i.e. to make the process variable align with the set-point at the next sampling instant, it is practically impossible to achieve perfection because of the inherent dead-time and process time-constant. The approach, however, leads to improvement in the performance of the control loop and is thus justified. In order to implement the algorithm, α is selected depending upon the process states (i.e. the values of e and $\dot{e}$) using an inference engine and rulebase where the rules are optimized using genetic algorithm. Transition of the process from one state to another should be such that the predicted controller output remains smooth. Fuzzy inference has been used here because its inherent interpolative nature makes such a transition possible. The inference mechanism to obtain α is shown in Fig. 1 and follows the Takagi–Sugeno model [7].

In line with the fuzzy logic control mechanism, the error and its derivative are normalized to [−1, +1] which becomes their domain for fuzzification. The fuzzifier block fuzzifies the two crisp input variables e and $\dot{e}$ into fuzzy variables A and B, respectively. As usual, the term sets for both the variables are taken as {NB, NM, NS, ZE, PS, PM,

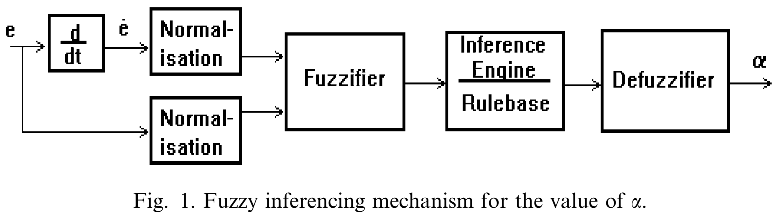

Fig. 1. Fuzzy inferencing mechanism for the value of α.

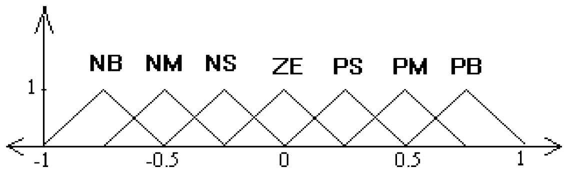

Fig. 2. Membership functions of both e and $\dot{e}$.

PB}. Their membership functions are shown in Fig. 2.

The generated rules are of the form:

$$R_i: \quad \text{if } e \text{ is } A_i \text{ and } \dot{e} \text{ is } B_i, \quad \text{then } \alpha \text{ is } C_i. \tag{6}$$

As per the Takagi–Sugeno model [7], the antecedent has fuzzy variables A and B and the consequent is a crisp variable C. Proper assignment of rule values is critical to the success of the entire scheme. The usual policy is to follow the procedure adopted in [5]. In the present paper, a genetic algorithm is used to optimize the rule values. The GA-based values lead to significant improvement in the performance of the PID autotuner. The rulebase is formed in such a way that denormalization for the value of α is not required. The value of α is obtained by the defuzzification method which employs the centre of height algorithm [13].

2.2. Implementation of the genetic algorithm

The outline of the genetic algorithm used in this paper is given below:

```
t = 0;
initialize population (t);
evaluate chromosomes in population (t);
while predetermined termination condition not satisfied
  {
    t = t + 1;
    select population(t) from population(t − 1);
    apply crossover and mutation to chromosomes in population(t);
    insert the best chromosome of population(t − 1) into population(t);
    evaluate chromosomes in population(t);
  }
```

2.2.1. Encoding

In the present implementation, a chromosome is a string of 52 floating-point variables. The first 49 variables represent the entries in the 7×7 matrix C, the 50th stands for the normalization factor for the error derivative and the 51st and 52nd variables denote the two constraint limits as discussed later in Section 2.3.

2.2.2. Initialization

The population size used here is 6. The initial population is created by setting the value of each variable in each chromosome to a uniform random number from the domain of the variable.

2.2.3. Fitness function

The fitness of a chromosome is calculated from the IAE. The problem has been formulated as one of minimizing the IAE.

2.2.4. Selection

Truncation selection [14] has been used where the best $T\%$ individuals at any generation are deterministically selected to produce offspring for the next generation. The truncation threshold has been taken as 40%.

2.2.5. Crossover

Two parent chromosomes are crossed to produce one child. The two parents are chosen from the mating pool (of size $T \times N/100$) by uniform sampling with replacement. The probability of crossover is unity. Arithmetic crossover [15] is used: the value of a variable in a child is obtained as the arithmetic mean of the values of that variable in its parents.

2.2.6. Mutation

Each variable of a chromosome undergoes mutation with mutation probability $= 0.25$. Let val_i be the current value of the ith variable ($i = 1, 2, \ldots, 52$) in a chromosome. Let $\text{range}_i = \text{high}_i - \text{low}_i$ represent the range of possible values of the ith variable. The changed value, val'_i of the ith variable is obtained by the following mechanism:

```
do {
  change_i = uniform-random
  (−range_i/10, +range_i/10);
```

```
    }   while   ($\text{val}_i + \text{change}_i < \text{low}_i$ or $\text{val}_i + \text{change}_i > \text{high}_i$);
$\text{val}'_i = \text{val}_i + \text{change}_i$;
```

2.2.7. Experiments

The elitist strategy is employed where the best-so-far individual is never lost from the population. The results presented in this paper for each case of each model (Section 3) are obtained as the best result out of five independent GA runs. No parameter tuning of the genetic algorithm was attempted.

2.3. New values of K_C, T_R and T_D

After obtaining the value of m_{n+1} using Eq. (5) and the rulebase, the right-hand side of Eq. (4) is evaluated. Let the value be denoted as ρ. Also let

$$K_{C_{SS}}\left(\frac{T}{2T_{R_{SS}}} - \frac{T_{D_{SS}}}{T}\right) = \beta, \tag{7}$$

where the suffix 'SS' stands for steady-state or initial values.

The value of ρ may be greater or less than β. When $\rho > \beta, K_C$ and T_D are to be increased and T_R is to be decreased. Similarly, when $\rho < \beta, K_C$ and T_D are to be decreased and T_R increased. The reason for increasing contributions of all the action modes for $\rho > \beta$, is only natural but as would be seen from the left-hand side of Eq. (4), increasing D action does provide slight redressing. The same argument holds for $\rho < \beta$.

These increments and decrements, if imposed without any constraint or modification, result in large oscillations thereby defeating the purpose of the proposed algorithm. It is true that the controller parameters are to be changed many times to meet the stringent requirement of dead-beat control. But these abrupt changes would result in unduly large oscillations. There should, therefore, be some constraint without largely affecting the principle of dead-beat control modes. The approach adopted here is that the tuning of the controller parameters is only grossly guided by the dead-beat technique and some pertinent modifications as well as constraints are imposed alongwith. All the controller parameters are allowed to vary within a prescribed range. The optimal range is found out by the genetic algorithm. If such a constraint *alone* is imposed, it would be observed that the controller parameters are likely to be always at the two extremities, which obviously is not desired for satisfactory performance. To counter this, the following method has been devised through which a smooth variation of the controller parameters has been ensured. Unless constraints are judiciously chosen, the algorithm of any self-tuning regulator, under normal circumstances, tends to drive off in the wrong direction and convergence of tuning may fail [16]. The presented method thus chooses the best possible values for these constraints using GA.

```
Case I: When $\rho > \beta$.
    Step 1. Let $\gamma = \rho - \beta$ and initialize $r_1$
    Step 2. If $\gamma > \gamma_{\max}$, then $\gamma_{\max} \leftarrow \gamma$
    Step 3. Let $\phi \leftarrow \gamma/\gamma_{\max}$
    Step 4. Obtain $K_C \leftarrow K_{C\,SS}\sqrt{(1 + r_1\phi)}$
                $T_R \leftarrow T_{R\,SS}/\sqrt{(1 + r_1\phi)}$
            and $T_D \leftarrow T_{D\,SS}\sqrt{(1 + r_1\phi)}$
Case II: When $\rho < \beta$.
    Step 1. Let $\psi = \beta - \rho$ and initialize $r_2$
    Step 2. If $\psi > \psi_{\max}$, then $\psi_{\max} \leftarrow \psi$
    Step 3. Let $\phi \leftarrow \psi/\psi_{\max}$
    Step 4. Obtain $K_C \leftarrow K_{C\,SS}/\sqrt{(1 + r_2\phi)}$
                $T_R \leftarrow T_{R\,SS}\sqrt{(1 + r_2\phi)}$
            and $T_D \leftarrow T_{D\,SS}/(1 + r_2\phi)$
```

Initially $\gamma_{\max}$ and $\psi_{\max}$ are set to 0. The value of ϕ is always between 0 and 1. It is nearer to 0 when the change sought for the controller parameters is small and is near 1 when the change required is more. The values of r_1 and r_2 are obtained as the 51st and the 52nd variables of the GA-based optimization scheme discussed in Section 2.2.

3. Simulation results

The proposed algorithm for autotuning of PID controllers has been applied to a number of process models. The responses for the first-order dead-time process model and the second-order dead-time process models are presented here since they encompass almost all industrial processes. It would be seen that the proposed method is better

e \ $\dot{e}$	NB	NM	NS	ZE	PS	PM	PB
NB	0.07375	-7.31564	-4.23187	-4.65937	-7.86651	-3.99180	3.68063
NM	-5.51577	0.07690	-6.19469	-4.58432	-5.53269	4.08432	4.30825
NS	-5.04078	-2.82924	0.00054	-4.64163	3.80239	3.56959	4.82629
ZE	-6.33535	-3.64308	-2.51039	0.21888	5.41272	5.26642	6.30143
PS	-5.88522	-2.04923	3.73623	2.78875	-0.71778	4.80270	6.23517
PM	-4.95947	5.18720	1.70715	3.78954	6.32118	-0.04711	4.61373
PB	6.29557	3.57108	2.94978	4.98727	4.04014	4.85271	0.20763

Fig. 3. Rules for the value of α for the FOPDT model with $\tau = 1.0$ and $\theta = 0.2$.

than the Ziegler–Nichols tuning method as well as the autotuning method in [5].

3.1. First-order plus dead-time model

The same model as used in [5] is chosen here for ease of comparison.

The proposed PID algorithm was tested on $T(s) = e^{-\theta \cdot s}/(\tau \cdot s + 1)$.

For this FOPDT model, simulation results are shown for $\theta = 0.2$ s and $\tau = 1.0$ s. The sampling time is taken as 0.05 s. The 49 values of the rules are obtained using the GA procedure discussed in Section 2.2 and the values obtained are shown in Fig. 3.

With the same GA, the optimum value for the normalization factor of error derivative is obtained as 7.63 and the controller parameters are allowed to vary by +30% and −70% of their initial values. The simulation results are shown in graphs in Fig. 4. The disturbance to the process is assumed to be a unit step at the set-point applied at $t = 0$ and a step of magnitude 0.5 to the load applied at $t = 5$ s. Comparison of the responses indicated by the curves in Fig. 4 shows that both with set-point change and load change, the proposed autotuner performs better than the Ziegler–Nichols tuner as well as the PID autotuner in [5] in so far as the IAE values are concerned. The proposed PID autotuner has less overshoot than

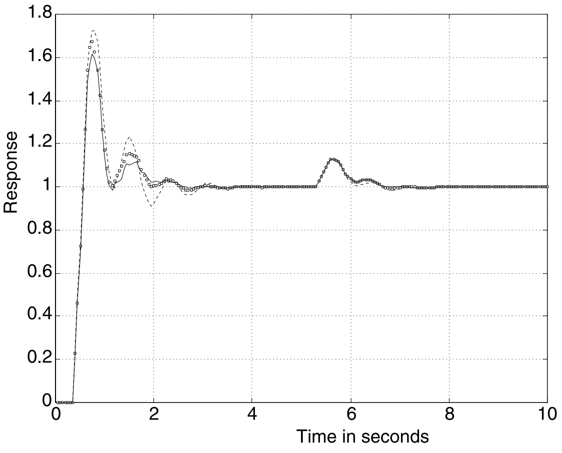

Fig. 4. Response of an FOPDT process model with $\tau = 1.0$ s, dead-time $(\theta) = 0.2$ s, sampling time $T = 0.05$ s. (i) ----: Ziegler–Nichols tuning (ii) ∘∘∘∘: PID autotuner in [5]. (iii) ———: Proposed fuzzy-genetic PID autotuner.

the other ones (≈60%). Both the rise-time and the settling time are reduced as compared to the other two methods. With load change, however, peak deviation decreases, but no substantial change in rise-time is observed and the improvement, therefore, is less. It is to be stressed that reductions in rise-time and peak deviation/overshoot are the primary requisites for improved performance. However, the actual reduction in IAE would also depend on the values of θ, τ and T as well.

Table 1
Performance analysis for the FOPDT process model

θ	Scheme[a]	%os	t_r	T_s	IAE
0.2	ZN	72.82	0.55	3.04	0.8436
0.2	PID	66.32	0.51	1.96	0.7877
0.2	FGPID	61.17	0.50	1.92	0.7321
0.3	ZN	57.36	0.75	2.90	1.0722
0.3	PID	55.28	0.70	2.85	1.0115
0.3	FGPID	52.76	0.70	2.70	0.8954
0.4	ZN	72.28	0.88	4.32	1.5152
0.4	PID	68.04	0.82	3.84	1.4214
0.4	FGPID	63.21	0.81	3.62	1.2351

[a] (i) ZN – Ziegler–Nichols PID controller. (ii) Fuzzy logic-based PID autotuner in [5]. (iii) Genetic algorithm and fuzzy logic-based proposed PID autotuner.

A comparison of performance is given in Table 1 for this FOPDT process model. The performance criteria for this purpose are taken as the percentage overshoot (%os), rise-time (t_r), settling time (t_s) and the IAE when the process is subjected to a disturbance of unit step at the set-point at $t = 0$ and a step disturbance of magnitude 0.5 at $t = 5$ s. The effect of different dead-times is considered to demonstrate the applicability of the proposed method in different processes.

3.2. Second-order plus dead-time models

(i) A second-order monotone process model with dead-time specified by

$$T(s) = \frac{e^{-\theta s}}{(\tau_1 s + 1)(\tau_2 s + 1)}$$

has next been taken for simulation study. As in the FOPDT model, the 49 values of the rules are obtained using the genetic algorithm and the values obtained are shown in Fig. 5.

The optimum value for the normalization factor of error derivative in this case is obtained as 10.9388 and the controller parameters are allowed to vary by +21% and −73% of their initial values as dictated by the genetic optimization algorithm. In this case too, the proposed algorithm performs better than the conventional PID controller tuned with Ziegler–Nichols formulae and the PID

$e \backslash \dot{e}$	NB	NM	NS	ZE	PS	PM	PB
NB	0.92387	-0.93000	-2.49864	-7.82982	-0.04046	-9.75625	9.10181
NM	-5.53107	-0.91296	-5.90272	-5.54487	-2.00077	0.90685	-1.14603
NS	3.68332	-9.39988	-1.96226	-8.04717	1.39896	-5.91325	3.89235
ZE	-3.38744	-9.58229	-6.95404	-2.53693	7.91210	4.03202	3.95296
PS	-5.72395	-5.52198	3.23575	8.37184	-0.54904	8.74446	9.03413
PM	-8.40410	7.77779	4.71582	8.13242	6.43713	0.93588	6.26733
PB	-0.47854	3.86555	9.26837	4.41472	7.93876	7.82138	-0.44876

Fig. 5. Rules for the value of α for the monotone SOPDT model with $\tau_1 = 0.4$ s, $\tau_2 = 0.5$ s and dead-time (θ) = 0.3 s.

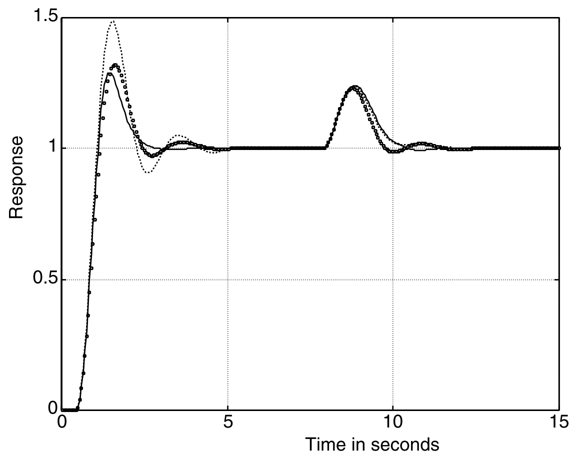

Fig. 6. Response of a SOPDT process model with $\tau_1 = 0.4$ s, $\tau_2 = 0.5$ s, dead-time $(\theta) = 0.3$ s, sampling time $T = 0.05$ s. (i) - - - -: Ziegler–Nichols tuning (ii) ○ ○ ○○ PID autotuner in [5]. (iii) ——— Proposed fuzzy-genetic PID autotuner.

autotuner proposed earlier [5]. The time-constants τ_1 and τ_2 have been taken as 0.4 and 0.5 s, respectively, the dead-time as 0.3 s and the sampling time as 0.05 s. The disturbance, as in the previous case, is a unit step at the set-point at $t = 0$, and after the output settles down at $t = 5$ s, the load disturbance of magnitude 0.5 is applied. As seen in the simulation plots in Fig. 6, there is considerable reduction in the peak deviation of the response curve obtained using the proposed PID autotuner as compared with the response curve obtained through Ziegler–Nichols tuner and, although there is no substantial improvement in the rise-time, the settling time and the IAE are reduced considerably with the proposed method. However, with load disturbance, the proposed autotuner makes the response settle faster compared to the other two controllers and hence the performance improves.

In Table 2, the comparison of performance with the SOPDT monotone process model is shown. Different values of process time-constants and the dead-time are chosen and it is observed that in all the cases the proposed technique produces better results than the other two methods. As in the FOPDT process model, here also the performance criteria chosen are %os, t_r, t_s and the IAE.

(ii) The second second-order process model taken for testing the proposed algorithm is given by

$$T(s) = \frac{e^{-\theta s}}{s(s+1)}.$$

This is a second-order oscillatory process model with considerable dead-time. This process model has been chosen because this type of process is difficult to control because of its inherent oscillatory nature. As in the other simulation

Table 2
Performance analysis for the monotone SOPDT process model

τ_1	τ_2	θ	Scheme[a]	%os	t_r	t_s	IAE
0.4	0.5	0.3	ZN	50.5	0.99	4.50	1.221
0.4	0.5	0.3	PID	46.5	1.05	3.85	1.149
0.4	0.5	0.3	FGPID	45.1	0.95	3.24	0.989
0.4	0.5	0.1	ZN	61.3	0.59	3.37	0.863
0.4	0.5	0.1	PID	56.6	0.58	3.31	0.813
0.4	0.5	0.1	FGPID	53.4	0.58	3.02	0.714
0.4	0.8	0.3	ZN	47.5	1.14	4.46	1.274
0.4	0.8	0.3	PID	43.4	1.16	3.89	1.218
0.4	0.8	0.3	FGPID	40.2	1.13	3.51	1.081
0.4	0.8	0.1	ZN	63.4	0.71	3.96	0.958
0.4	0.8	0.1	PID	62.1	0.65	3.83	0.898
0.4	0.8	0.1	FGPID	58.4	0.65	3.32	0.764

[a] (i) ZN – Ziegler–Nichols PID controller. (ii) Fuzzy logic-based PID autotuner in [5]. (iii) Genetic algorithm and fuzzy logic-based proposed PID autotuner.

e \ $\dot{e}$	NB	NM	NS	ZE	PS	PM	PB
NB	0.72383	0.35116	4.45008	2.78120	-11.9403	-0.95068	9.09643
NM	-1.26021	3.28147	-0.46632	-3.08439	-5.38439	4.92399	4.63662
NS	-1.51058	-2.01559	-0.68936	17.6800	15.6270	1.39058	6.03109
ZE	13.11984	-2.52018	-0.82939	-1.89050	0.33249	1.21259	1.38239
PS	24.7036	-13.7379	6.61990	7.52004	-2.5136	8.18128	4.96074
PM	-6.65028	-7.24682	-5.14442	6.24623	3.98471	-0.85491	1.07848
PB	6.47207	-4.92975	1.31929	1.211947	0.91149	4.27328	0.16714

Fig. 7. Rules for the value of α for the oscillatory SOPDT model with dead-time $(\theta) = 0.3$ s.

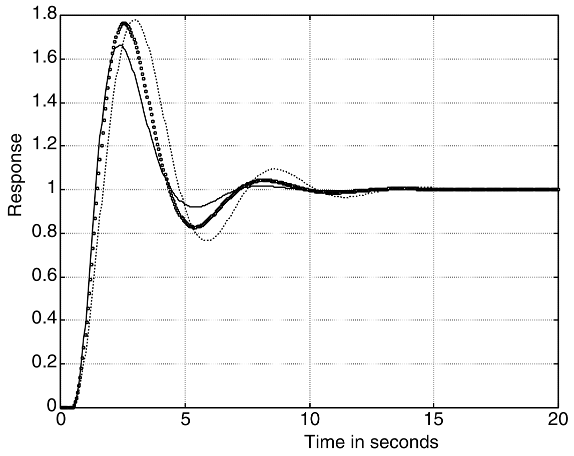

Fig. 8. Responses of a second-order oscillatory process model with dead-time $(\theta) = 0.2$ s, sampling time $T = 0.05$ s. (i) - - - -: Ziegler–Nichols tuning. (ii) ○ ○ ○○ PID autotuner in [5]. (iii) ———: Proposed fuzzy-genetic PID autotuner.

models, the 49 values of the rules are presented in Fig. 7.

The value for the normalization factor of error derivative used here is 6.480005 and the controller parameters are allowed to vary within +99% and −78% of their initial values as obtained by the GA-based optimization algorithm. Fig. 8 shows the simulation results with a unit step disturbance at the set-point. It is clearly seen that the proposed method reduces the rise-time and settling time, and the overshoot is also decreased substantially resulting in large reduction in IAE.

In Table 3, performance comparison of the three methods is shown. It is observed that the improvement in IAE is more when the process has larger dead-time. This clearly establishes the efficacy of the proposed PID autotuner for almost all types of processes.

Table 3
Performance analysis for the SOPDT oscillatory process model

θ	Scheme[a]	%os	t_r	t_s	IAE
0.3	ZN	78.81	1.75	13.50	3.34
0.3	PID	74.65	1.62	12.60	3.01
0.3	FGPID	63.21	1.56	11.27	2.59
0.4	ZN	77.0	2.60	16.36	4.66
0.4	PID	73.4	2.21	15.97	4.30
0.4	FGPID	56.47	2.11	12.75	3.15

[a] (i) ZN – Ziegler–Nichols PID controller. (ii) Fuzzy logic-based PID autotuner in [5]. (iii) Genetic algorithm and fuzzy logic-based proposed PID autotuner.

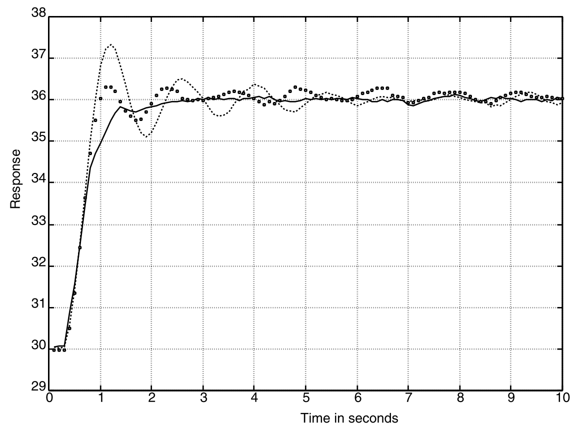

Fig. 9. Experimental Results with PT326 thermal process from FEEDBACK. (i) ----: Ziegler–Nichols tuning. (ii) ∘ ∘ ∘ PID autotuner in [5]. (iii) ———: Proposed fuzzy-genetic PID autotuner.

4. Experimental results

The proposed PID autotuning algorithm has been tested on a laboratory type thermal process module PT326 procured from FEEDBACK. The response of this algorithm is compared with the Ziegler–Nichols PID controller and the PID autotuner presented in [5]. The responses are shown in Fig. 9 where the sampling time has been taken as 0.1 s. In order to obtain the initial controller parameters, the ultimate cyclic method of Ziegler and Nichols was chosen, from which the ultimate gain was found to be 1.1 and the ultimate period 0.5 s. The disturbance to the process is a step input at the set-point from 30°C to 36°C. Here the improvement in IAE over the previously proposed PID autotuner is 10.2% and that over the Ziegler–Nichols tuner is 18.8%.

5. Conclusion

A fuzzy-genetic technique of autotuning a PID controller has been presented in this paper. It is an extension of the method proposed previously in [5] where a rulebase was used and the rule values were derived from the knowledge of process experts. There was, however, ample scope of improving the performance of the autotuner by optimizing the values in the rulebase. Standard optimization techniques such as the method of Nelder and Mead [17] used in the optimization toolbox of MATLAB [18] failed to provide good solution perhaps because of the presence of a number of local optima. The present approach has, therefore, been followed which utilized the technique of evolutionary computation for optimization of 52 floating point variables. A notable feature of the present method is that autotuning is based on the format of dead-beat control, which ensures tighter control as the output response aligns faster with the set-point. The predicted controller output is obtained via fuzzy inference mechanism, which makes the variation of the controller output smooth. Hence the method can be applied to processes having mechanical actuators where jerks from the controller would cause damage to the actuators.

Simulation results with FOPDT and SOPDT process models have been presented along with the results obtained with a controller tuned with Ziegler–Nichols formulae and an algorithm proposed in [5]. The proposed PID autotuner has outperformed the other two controllers. For monotone processes, the improvement appears to be marginal for set-point change, but for reasonably frequent load disturbances the performance is superior to the other two. For the inherently oscillatory processes with large dead-time, the performance of the proposed method is far better than the others. Finally, the experimental results obtained with a thermal process kit establish the superiority of the algorithm.

The algorithm is generalized in the sense that it tunes all the parameters of the PID controllers online and it is expected that it would perform better in almost all types of applications specially in processes where disturbances are likely to occur frequently and in processes with a pole at the origin as in servo-type processes. Simulation results and practical tests have confirmed this.

References

[1] T.W. Kraus, T.J. Myron, Self-tuning PID controller uses pattern recognition approach, Control Eng. (1984) 106–111.

[2] H.S. Hoopes, W.M. Hawk Jr., R.C. Lewis, A self-tuning controller, ISA Trans. 22 (1983) 49–58.

[3] J.G. Ziegler, N.B. Nichols, Optimum settings for automatic controllers, Trans. ASME 64 (1942) 759–768.

[4] R. Bandyopadhyay, D. Patranabis, A fuzzy logic based PI autotuner, ISA Trans. 37 (1998) 227–235.

[5] R. Bandyopadhyay, D. Patranabis, A new autotuning algorithm for PID controllers using dead-beat format, ISA Trans. 40 (3) (2001) 255–266.

[6] P.B. Deshpande, R.H. Ash, Elements of computer process control with advanced control applications, Instr. Soc. Am., 1981.

[7] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cybern. 15 (1) (1985) 116–132.

[8] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[9] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

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

[11] J.-X. Xu, C. Liu, C.C. Hang, Tuning of fuzzy PI controllers based on gain/phase margin specifications and ITAE index, ISA Trans. 35 (1996) 79–91.

[12] K.J. Astrom, T. Hagglund, Automatic tuning of PID controllers, Instrum. Soc. Am., 1988.

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

[14] H. Muehlenbein, D. Schlierkamp-Voosen, The science of breeding and its application to the breeder genetic algorithm, Evol. Comput. 1 (4) 335–360.

[15] Z. Michalewicz, C. Janikow, Genetic algorithms for numerical optimization, Statist. Comput. 1 (1) (1991).

[16] O.L.R. Jacobs, P.F. Hewkian, C. While, On line computer control of pH in an industrial process, Proc. IEE 127 (4) (1980) 161–168.

[17] J.A. Nelder, R. Mead, A simplex method for function minimization, Comput. J. 7 308–313.

[18] MATLAB Optimization Toolbox User's Guide, Mathworks Inc., 1998.

R. Bandyopadhyay is a Reader in the Department of Instrumentation Engineering, Jadavpur University, Kolkata, India. An M.E. in Electronics and Tele-communication Engineering, Jadavpur University, he worked in the CAD Center of the same university for the first three years of his professional career. He has recently been awarded the Ph.D. degree and is currently interested in the areas of Intelligent Process Control.

U.K. Chakraborty is an Assistant Professor in Computer Science at University of Missouri, St. Louis. He has held positions at CMC Ltd. (India), Jadavpur University (India), GMD-Forschungszentrum Informationstechnik (Germany) and Carnegie Mellon University, Pittsburgh. His research interests are in evolutionary computation and software engineering. He has published one book and forty research articles. He is a member of the editorial board of the Journal of Computing and Information Technology.

D. Patranabis is a Professor in the Department of Instrumentation Engineering, Jadavpur University, Kolkata, India. An M.Sc. (Tech.) and Ph.D. from the University of Calcutta, he taught Physics for a short while, worked for Damodar Valley Corporation as Electrical Engineer and with Guest Keen Williams Ltd. as Instrument Engineer before joining Jadavpur University in 1966. He was the honorary editor IIST (India) for six years, honorary summary writer and editor for Zentralblatt fuer Mathematik (Springer) for about seven years in Control Group. He has authored five books published by Tata McGraw-Hill and Wheeler Publishing, and over 100 technical research papers. His current research interest is in the areas of Process Control Systems and Sensors and Transducers. He is a member of IE(I) and IETE(I).