ELSEVIER

Fuzzy Sets and Systems 101 (1999) 21–30

FUZZY sets and systems

Tuning of PID controllers based on gain and phase margin specifications using fuzzy neural network

Sheng-Yuan Chu, Ching-Cheng Teng*

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

Received October 1996; received in revised form January 1997

Abstract

We propose a new PID tuning method using fuzzy neural networks for a given gain and phase margin specifications (FNGP). We use fuzzy neural networks to determine the PID controller parameters. Because the definitions of gain and phase margin equations are complex, an analytical tuning method for achieving specified gain and phase margins is not available yet. In this paper, a fuzzy neural modeling method is first proposed to identify the relationship between the gain-phase margin specifications and the PID controller parameters. Then, the FNGP is used to automatically tune the PID controllers parameter for different gain and phase margin specifications so that neither numerical methods nor graphical methods need be used. Simulation results show that the FNGP can achieve the specified values much more efficiently than other methods. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Fuzzy neural network; Process control; PID tuning; Gain and phase margin

1. Introduction

Over the past 50 years, several methods for determining PID controller parameters have been developed. Some employ information about open-loop step response, for example, the Coon–Cohen reaction curve method [3]; other methods use knowledge of the Nyquist curve, e.g., the Ziegler–Nichols frequency-response method. However, these tuning methods use only a small amount of information about the dynamic behavior of the system, and often do not provide good tuning.

Gain margin and phase margin have always served as important measures of robustness. It is also known from classical control theories that phase margin is related to the damping of the system, and therefore also serves as a performance measurement. Controllers designed to satisfy gain margin and phase margin (GPM) criteria are not new approaches. However, their solutions are normally obtained numerically or graphically by trial-and-error use of Bode plots. Such methods are certainly not suitable for use in adaptive control and auto-tuning.

Tuning of PID controllers based on gain and phase margin specifications (GPM) was proposed by Hang [4]. This method uses linear equations to approximate the arctan function in the gain-phase margin definition,

* Corresponding author.

PII: S0165-0114(97)00039-0

formulas are then derived to design the PI and PID controllers to meet user-specified gain margins and phase margins. The disadvantage of the GPM method is that the transfer function of the controlled process is restricted to the first-order-plus-time-delay type. This approach is certainly not suitable for use in adaptive control and auto-tuning.

Although various kinds of fuzzy logic controllers (FLCs) [8] are widely used nowadays and have certain advantages over conventional PID controllers, relatively few theoretical analysis that explain why they can achieve better performance are available.

In this paper, we present a new tuning method that uses a fuzzy neural network based on gain and phase margin specifications (FNGP), to tune the PID controller parameters efficiently and meet user-specified gain and phase margins exactly. This approach enjoys the advantage of functionally mapping the fuzzy neural network, and gives better performance than GPM.

The arrangement of this paper is as follows. In Section 2, we briefly introduce gain and phase margins. Section 3 proposes the fuzzy neural network structure. Section 4 describes the proposed tuning method for a fuzzy neural controller based on gain and phase margin specifications (FNGP). Section 5 gives the simulation results and discusses the advantages of the proposed approach as compared with other methods. Finally, conclusions are summarized in Section 6.

2. Gain margin and phase margin

Denote the process and the controller transfer function by $G_p(s)$ and $G_c(s)$, as shown in Fig. 1, and the specified gain and phase margins by A_m and ϕ_m, respectively. The formulas for gain margin and phase margin are as follows:

$$\arg[G_c(jw_p)G_p(jw_p)] = -\pi, \tag{1}$$

$$A_m = \frac{1}{|G_c(jw_p)G_p(jw_p)|}, \tag{2}$$

$$|G_c(jw_g)G_p(jw_g)| = 1, \tag{3}$$

$$\phi_m = \arg[G_c(jw_g)G_p(jw_g)] + \pi, \tag{4}$$

where the gain margin is defined by Eqs. (1) and (2), and the phase margin by Eqs. (3) and (4). The frequency w_p at which the Nyquist curve has a phase of $-\pi$ is known in classical terminology as the phase crossover frequency, and the frequency w_g at which the Nyquist curve has an amplitude of 1 as the gain crossover frequency.

The PI controller is given by

$$G_c(s) = K_c\left(1 + \frac{1}{sT_I}\right)$$

and the process is given by

$$G_p(s) = \frac{K_p(1 + w_{n1}s)^{n1}(1 + w_{n2}s)^{n2}\cdots(1 + w_{nq}s)^{nq}}{(1 + w_{d1}s)^{d1}(1 + w_{d2}s)^{d2}\cdots(1 + w_{dp}s)^{dp}}\,e^{-Ls}.$$

The loop transfer function is obtained from

$$G_c(s)G_p(s) = \frac{K_cK_p(1 + sT_I)(1 + w_{n1}s)^{n1}(1 + w_{n2}s)^{n2}\cdots(1 + w_{nq}s)^{nq}}{sT_I(1 + w_{d1}s)^{d1}(1 + w_{d2}s)^{d2}\cdots(1 + w_{dp}s)^{dp}}\,e^{-Ls}.$$

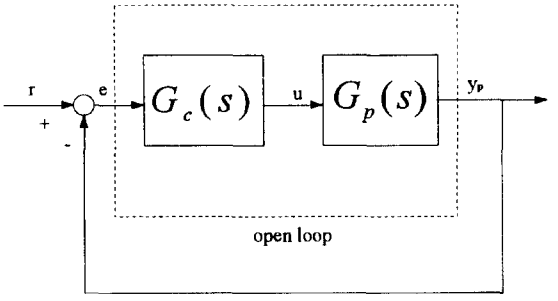

Fig. 1. The overall system.

Substituting the above equation into Eqs. (1)–(4), we have

$$\tfrac{1}{2}\pi + \tan^{-1}(w_p T_I) + n_1 \tan^{-1}(w_p w_{n1}) + \cdots + n_q \tan^{-1}(w_p w_{nq}) - w_p L - d_1 \tan^{-1}(w_p w_{d1}) - d_2 \tan^{-1}(w_p w_{d2}) - \cdots - d_p \tan^{-1}(w_p w_{dp}) = 0, \tag{5}$$

$$A_m K_c K_p = w_p T_I \frac{\sqrt{1 + w_p^2 T_I^2}\sqrt{(1 + w_p^2 w_{n1}^2)^{n1}} \cdots \sqrt{(1 + w_p^2 w_{nq}^2)^{nq}}}{\sqrt{(1 + w_p^2 w_{d1}^2)^{d1}}\sqrt{(1 + w_p^2 w_{d2}^2)^{d2}} \cdots \sqrt{(1 + w_p^2 w_{dp}^2)^{dp}}}, \tag{6}$$

$$K_c K_p = w_g T_I \frac{\sqrt{(1 + w_g^2 w_{d1}^2)^{d1}}\sqrt{(1 + w_g^2 w_{d2}^2)^{d2}} \cdots \sqrt{(1 + w_g^2 w_{dp}^2)^{dp}}}{\sqrt{(1 + w_g^2 T_I^2)}\sqrt{(1 + w_g^2 w_{n1}^2)^{n1}} \cdots \sqrt{(1 + w_g^2 w_{nq}^2)^{nq}}}, \tag{7}$$

$$\phi_m = \tfrac{1}{2}\pi + \tan^{-1}(w_g T_I) + n_1 \tan^{-1}(w_g w_{n1}) + \cdots + n_q \tan^{-1}(w_g w_{nq}) - w_g L - d_1 \tan^{-1}(w_g w_{d1}) - d_2 \tan^{-1}(w_g w_{d2}) - \cdots - d_p \tan^{-1}(w_g w_{dp}). \tag{8}$$

For a given process $(K_p, w_{n1}, \ldots, w_{nq}, w_{d1}, \ldots, w_{dp}, L)$ and specifications (A_m, ϕ_m), Eqs. (5)–(8) can be solved for the PI controller parameters (K_c, T_I) and crossover frequencies (w_g, w_p) numerically but not analytically because of the presence of the arctan function. Controllers such as the IMC [2] and GPM [4] that are based on gain and phase margins cannot efficiently meet specifications within a 10% error margin owing to approximation of the arctan function. Therefore, another approach using a fuzzy neural network is considered here.

3. Structure of a controller that uses a fuzzy neural network based on gain and phase margins (FNGP)

The overall system block diagram is shown in Fig. 1. To get parameters (K_c, T_I) for a PI controller more exactly, analytically avoid the presence of arctan functions, and a lack of solutions for the nonlinear Eqs. (5)–(8), we use fuzzy neural networks [6, 9] based on gain and phase margins (FNGP) to model these equations analytically.

Considering the nonlinear coupled Eqs. (5)–(8), we find there are four parameters (w_p, w_g, K_c, T_I) in those four equations. If we are given gain margin and phase margin specifications (A_m, ϕ_m), it may not be possible to solve the four parameters because the equations are nonlinear.

Now, let us consider another aspect of these equations. First, we could put random controller parameters (K_c, T_I) into those equations. Using Eq. (5), we could solve for w_p then substitute it into Eq. (6) to get A_m.

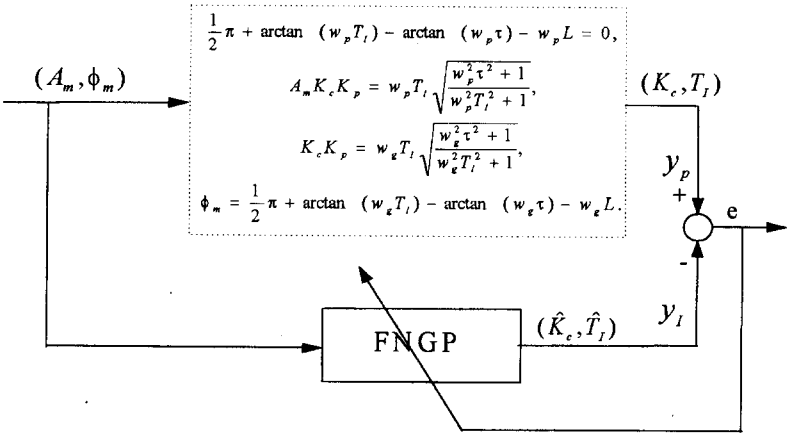

Fig. 2. Block diagram of function mapping using FNGP.

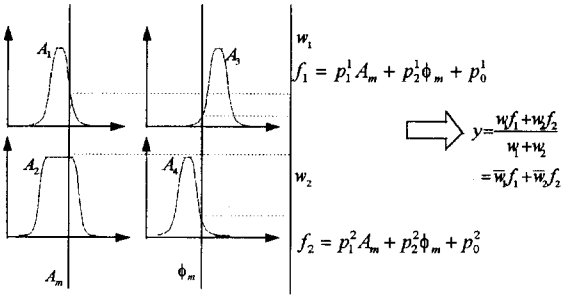

Fig. 3. A two-input first-order Sugeno fuzzy model.

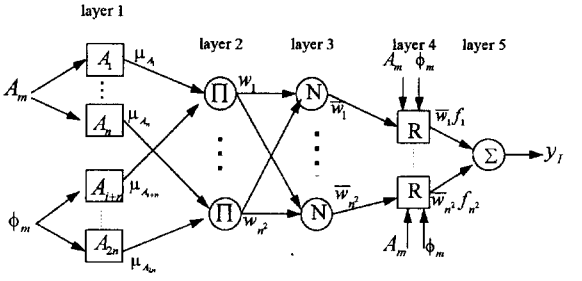

Fig. 4. The FNGP networks architecture.

And using Eq. (7), we could calculate w_g then substitute it into Eq. (8) to obtain ϕ_m. Hence we could obtain the parameters (w_p, w_g, A_m, ϕ_m) that correspond to the random controller parameters (K_c, T_I) respectively. For example, in preparation for training the FNGP, we gather 200 points (K_c, T_I) and corresponding (A_m, ϕ_m) points, respectively, and put them into Fig. 2 as the training data. This approach avoids the problem of no solution being yielded by nonlinear Eqs. (5)–(8), and reduces the overall task dimensions.

Fig. 2 illustrates the block diagram of the function mapping of Eqs. (5)–(8) using FNGP with a Sugeno fuzzy model. A simple two-input first-order Sugeno fuzzy model is shown in Fig. 3.

The FNGP architecture is shown in Fig. 4. Suppose we are given (A_m, ϕ_m) and have R^i $(i = 1, \ldots, n^2)$ implications, then the value of $y \in \{K_c, T_I\}$ is implied as follows.

Layer 1: Every node i in this layer is an adaptive node with a node output defined by (Here we denote the output node i in layer l as $O_{l,i}$.)

$$O_{1,i} = \mu_{A_i}(A_m), \quad \text{for } i = 1, \ldots, n,$$

$$O_{1,i+n} = \mu_{A_{i+n}}(\phi_m),$$

where x_i is the input to the node and A_i is a fuzzy set associated with this node. In other words, outputs from this layer are the membership values of the premise part. Here the membership function can be characterized

by the generalized bell-shaped function:

$$\mu_{A_i}(x) = \frac{1}{1 + [(x_i - c_i)^2/a_i^2]^{b_i}},$$

where $\{a_i, b_i, c_i\}$ is in the parameter set S_1. Parameters in this layer are referred to as *premise parameters.*

Layer 2: Every node in this layer is a fixed node labeled Π, which multiplies the incoming signals and outputs the product,

$$O_{2,k} = w_k = \mu_{A_i}(A_m) \times \mu_{A_j}(\phi_m), \quad i,j = 1,\ldots,n, \quad k = 1,\ldots,n^2.$$

Each output of the node represents the firing strength of a rule. (In fact, any other T-norm operators that perform fuzzy AND operations can be used as the node functions in this layer.)

Layer 3: Every node in this layer is a fixed node labeled N. The ith node calculates the ratio of the ith rule's firing strength to the sum of all rule's firing strengths:

$$O_{3,i} = \bar{w}_i = \frac{w_i}{w_1 + \cdots + w_{n^2}}, \quad i = 1,\ldots,n^2.$$

For convenience, the outputs from this layer are called *normalized firing strengths.*

Layer 4: Every node i in this layer is an adaptive node with a node function

$$O_{4,i} = \bar{w}_i f_i = \bar{w}_i(p_1^i A_m + p_2^i \phi_m + p_0^i), \quad i = 1,\ldots,n^2,$$

where $\bar{w}_i$ is the output of layer 3 and $\{p_0^i, p_1^i, p_2^i\}$ is in the parameter set S_2. Parameters in this layer are referred to as *consequent parameters.*

Layer 5: The single node in this layer is a fixed node labeled $\sum$ that computes the overall outputs as the summation of all incoming signals:

$$y_I = O_{5,1} = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}, \quad i = 1,\ldots,n^2.$$

4. Tuning of the FNGP

We noted that when the values of the premise parameters are fixed, the overall output $y_I \in \{K_c, T_I\}$ can be expressed as a linear combination of consequent parameters. In symbols, the output y_I in Fig. 4 can be written as

$$\begin{aligned} y_I &= \frac{w_1}{w_1 + \cdots + w_{n^2}} f_1 + \cdots + \frac{w_{n^2}}{w_1 + \cdots + w_{n^2}} f_{n^2} = \bar{w}_1 f_1 + \cdots + \bar{w}_{n^2} f_{n^2} \\ &= (\bar{w}_1 A_m) p_1^1 + (\bar{w}_1 \phi_m) p_2^1 + (\bar{w}_1) p_0^1 + \cdots + (\bar{w}_{n^2} A_m) p_1^{n^2} + (\bar{w}_{n^2} \phi_m) p_2^{n^2} + (\bar{w}_{n^2}) p_0^{n^2}, \end{aligned}$$

which is linear in the consequent parameters $\{p_0^1, p_1^1, p_2^1, \ldots, p_0^{n^2}, p_1^{n^2}, p_2^{n^2}\}$. Note that if a fuzzy neural network output or its transformation is linear in some of the network's parameters, then we can identify these linear parameters using the well-known linear least-squares method [5].

4.1. Off-line learning

For simplicity, we assume that the fuzzy neural network under consideration has only one output

$$\text{output} = F(\vec{I}, S), \tag{9}$$

where $\bar{I}$ is the input variable vector and S is the set of parameters. If there exists a function H such that the composite function $H \circ F$ is linear in some of the elements of S, then these elements can be identified by the least-squares method. More formally, if the parameter set S can be decomposed into two sets

$$S = S_1 \oplus S_2,$$

where $\oplus$ represents *direct sum*, such that $H \circ F$ is linear over the elements of S_2, then upon applying H to Eq. (9), we have [6]

$$H(\text{output}) = H \circ F(\bar{I}, S), \tag{10}$$

which is linear over the elements of S_2. Now, given values of elements of S_1, we can put M training data into Eq. (10) and obtain the matrix equation:

$$\begin{aligned} &XP = Y, \qquad Y = [y_1, \ldots, y_2]^{\mathrm{T}}, \\ &P = [p_0^1, \ldots, p_0^n, \ldots, p_1^1, \ldots, p_1^n, \ldots, p_k^1, \ldots, p_k^n]. \end{aligned} \tag{11}$$

Here, P is an unknown vector whose elements are parameters in S_2. This equation represents the standard, linear least-squares problem and the best solution for P, that minimizes $\|XP - Y\|^2$, is the least-squares estimator (LSE) P^*:

$$P^* = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}Y,$$

where X^{T} is the transpose of X and $(X^{\mathrm{T}}X)^{-1}$ is the pseudo-inverse of $X^{\mathrm{T}}X$. The parameter vector P can be calculated using a stable-state Kalman filter, which is an algorithm for calculating the parameters of a linear algebraic equation that gives the least squares of errors. Of course, we can also employ the recursive LSE formula [5]. Specifically, let the ith row vector of matrix X defined in Eq. (11) be x_i^{T}, and the ith element of Y be y_i^{T}; P can then be calculated recursively as follows:

$$\begin{aligned} &P_{i+1} = P_i + Q_{i+1}x_{i+1}(y_{i+1} - x_{i+1} \cdot P_i), \\ &Q_{i+1} = Q_i - \frac{Q_i x_{i+1} x_{i+1}^{\mathrm{T}} Q_i}{1 + x_{i+1}^{\mathrm{T}} Q_i x_{i+1}}, \quad i = 0, 1, \ldots, M-1, \\ &P = P_M, \end{aligned}$$

where the initial values of P_0 and Q_0 are set as follows:

$$P_0 = 0, \qquad Q_0 = \alpha \cdot I,$$

where α is a large number and I is the identity matrix.

For given fixed values of parameter in S_1, the parameters in set S_2 thus found are guaranteed to be global optimum points in set S_2 parameter space because of the choice of the squared-error measure. Not only can this hybrid learning rule decrease the dimensions of the search space in gradient descent method, but, in general, it will also substantially reduce the time needed to reach convergence.

For the recursive least-squares formula to account for the time-varying characteristics of the incoming data, the effects of old data pairs must decay as new data pairs become available. One simple method for ensuring this is to formulate the squared error measure as a weighted version that gives higher weighting factors to more recent data pairs. This amounts to addition of a forgetting factor λ to the original recessive formula:

$$P_{i+1} = P_{i+1} + Q_{i+1}x_{i+1}(y_{i+1}^{\mathrm{T}} - x_{i+1}^{\mathrm{T}}P_{i+1}), \qquad Q_{i+1} = \frac{1}{\lambda}\left[Q_{i+1} - \frac{Q_{i+1}x_{i+1}x_{i+1}^{\mathrm{T}}Q_i}{\lambda + x_{i+1}^{\mathrm{T}}Q_i x_{i+1}}\right],$$

where typical value for λ in practice is between 0.9 and 1. The smaller λ is, the faster the effects of old data decay. A small λ sometimes causes numerical instability.

Table 1

	Forward pass	Backward pass
Premise parameters	Fixed	Backpropagation
Consequent parameters	Least-squares method	Fixed

4.2. On-line learning

If the parameters are updated after each data presentation, we have an on-line learning scheme. This learning strategy [1] is vital to on-line parameter identification by systems with changing characteristics. In this learning scheme, we use back-propagation learning [8] to modify the premise parameters $\{a_i, b_i, c_i\}$ in parameter set S_1.

Let the cost function, E_l, of the training pattern k be proportional to the square of the difference between the plant output $y_p(k)$ in the frequency domain and the actual output $y_l(k)$ of NFGP in Fig. 2, and let E_l be defined by

$$E_l = \tfrac{1}{2}[y_p(k) - y_l(k)]^2. \tag{12}$$

The gradient of error in Eq. (11) with respect to the parameter weighting vector S_1 in Fig. 4 then becomes

$$\frac{\partial E_l}{\partial S_1} = e_l(k)\frac{\partial e_l(k)}{\partial S_1} = -e_l(k)\frac{\partial y_l(k)}{\partial S_1} = -e_l(k)\frac{\partial O_l(k)}{\partial S_1}.$$

The weights can be adjusted using the following gradient descent method:

$$S_1(k+1) = S_1(k) + \Delta S_1(k) = S_1(k) + \eta_l\left(-\frac{\partial E_l}{\partial S_1}\right),$$

where η_l is a learning rate.

Thus, the learning process is as listed in Table 1.

5. Simulation

The approach based on gain and phase margins as important measures is attractive not only in performance, but also in robustness. In this section, we give a specific performance comparison with GPM because both were designed based on gain and phase margin specifications.

Example 1 (*High-order process*). The process is given as follows:

$$G_p(s) = \frac{1}{(1+s)^5}.$$

Since the process is not a first-order type, the GPM cannot be applied. Therefore, a first-order type with time-delay model is used to approximate a high-order type. Various gain and phase margins are specified for this model in Table 2. It can be seen that the FNGP performs much better than the GPM design. A performance and robustness comparison is shown in Fig. 5.

Example 2 (*First-order with time-delay process*). The process is given as

$$G_p(s) = \frac{e^{-0.1s}}{1+s}, \quad L/\tau = 0.1 < 1.$$

Table 2
Different PI controllers for $G_p(s) = 1/(1+s)^5$

Tuning method	High order	Specifications		Resultant						Error	
	$1/(1+s)^5$	A_m	ϕ_m°	K_c	T_I	w_g	w_p	A_m^*	ϕ_m^*	A_m'	ϕ_m'
GPM	$e^{-2.93s/(1+2.73s)}$	3	60	0.4879	2.7300	0.1771	0.4828	3.8528	59.026	28.4%	1.62%
		2.5	70	0.6350	3.7335	0.1823	0.5319	3.9115	65.728	56.5%	6.10%
FNGP	$1/(1+s)^5$	3	60	0.5559	2.7281	0.2109	0.5621	3.0051	60.355	0.17%	0.59%
		2.5	70	0.8723	4.6649	0.2456	0.6297	2.5141	69.705	0.56%	0.42%

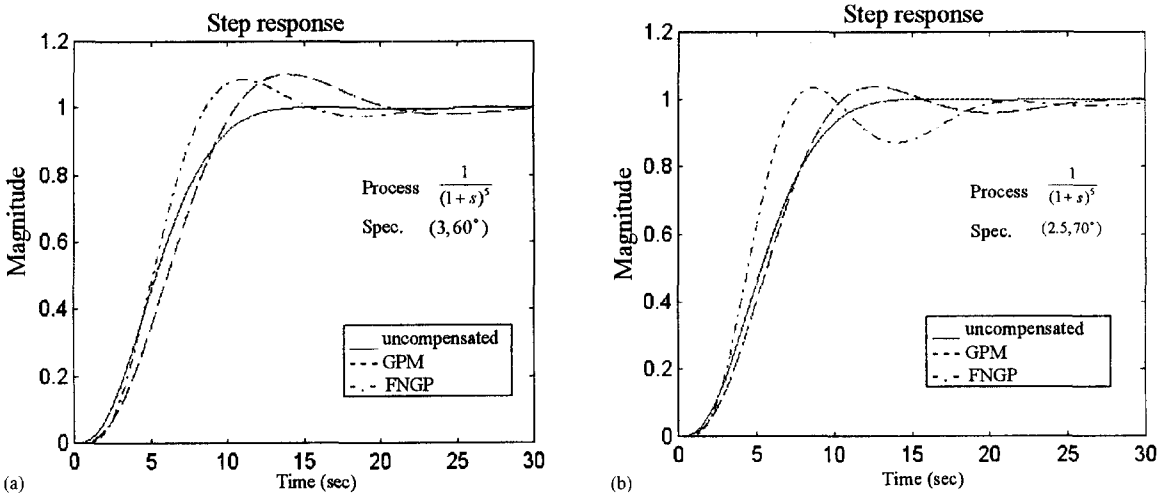

Fig. 5. (a) Step response of FNGP and GMP with Spec. $(3, 60°)$ in Example 1. (b) Step response of FNGP and GPM with Spec. $(2.5, 70°)$ in Example 1.

Table 3
Different PI controllers for $G_p(s) = e^{-0.1s}/(1+s)$

Tuning method	First-order time delay	Specifications		Resultant						Error	
	$e^{-0.1s}/(1+s)$	A_m	ϕ_m°	K_c	T_I	w_g	w_p	A_m^*	ϕ_m^*	A_m'	ϕ_m'
GPM	$e^{-0.1s}/(1+s)$	3	45	4.9087	0.3520	5.3984	12.780	3.3075	40.308	10.2%	10.4%
		5	60	3.0543	0.5410	3.3413	13.534	5.9904	58.140	19.8%	3.10%
FNGP	$e^{-0.1s}/(1+s)$	3	45	6.1984	0.5626	6.2430	13.586	2.9786	45.411	0.71%	0.91%
		5	60	3.7851	0.6601	3.9214	13.777	5.0292	60.039	0.58%	0.07%

The results for different specifications in this example are illustrated in Table 3, which shows that even when the plant is first-order with time-delay, the proposed FNGP approach has better performance than GPM. FNGP

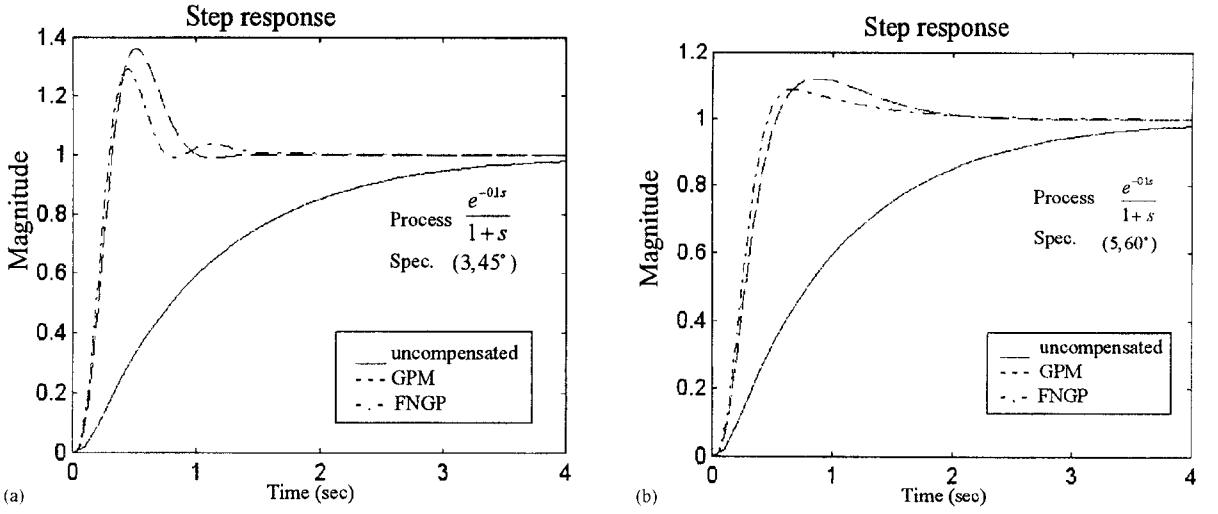

Fig. 6. (a) Step response of FNGP and GPM with Spec. (3,45°) in Example 2. (b) Step response of FNGP and GPM with Spec. (5,60°) in Example 2.

yields less than a 1% error but GMP's is greater than 10%. FNGP also has such a shorter rise time and lower overshoot than GPM, as shown in Fig. 6.

6. Conclusions

This paper has investigated fuzzy neural controllers based on gain and phase margins (FNGP). The proposed FNGP is a new approach to tuning controller parameters according to gain margin and phase margin specifications.

There are two advantages of using the proposed FNGP for formulating gain and phase margin problems. First, the trained FNGP automatically tunes PI controller parameters for different gain and phase margin specifications so that neither numerical methods nor graphical methods need be used. Second, the FNGP can also find the relationship between PI controllers (K_c, T_I) and specifications (A_m, ϕ_m) in the weighting parameters in the networks. Therefore, the proposed method is simple and systematic in reducing the dimensions of the problem presented in this paper.

PID controller tuning also employs the same approach in meeting user-specified gain and phase margin specifications. The PID tuning formulas are detailed in [4].

References

[1] A.G. Barto, R.S. Sutton, C.W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, IEEE Trans. Systems Man Cybernet. SMC-13 (1983) 834–846.

[2] I.-L. Chien, P.S. Fruehauf, Consider IMC tuning to improve controller performance, Chem. Eng. Progr. 86(10) (1990) 33–41.

[3] G.H. Cohen, G.A. Coon, Theoretical consideration of retarded control, Trans. ASME 75 (1953) 827–834.

[4] W.K. Ho, C.C. Hang, L.S. Cao, Tuning of PID controllers based on gain and phase margin specification, Automatica 31 (1995) 497–502.

[5] T.C. Hsia, System Identification: Least-Squares Methods, Heath, New York, 1977.

[6] J.-S.R. Jang, C.T. Sun, Neuro-Fuzzy Modeling and Control, Proc. IEEE 83 (1995) 378–405.

[7] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed: Explorations in Microstructure of Cognition, vol. 1, MIT Press, Cambridge, MA, 1986, Chap. 8, pp. 318–362.
[8] J.X. Xu, C. In, C.C. Hang, Tuning and analysis of a fuzzy logic controller based on gain and phase margins, Proc. American Control Conf., 1995, pp. 3234–3238.
[9] R.R. Yager, D.P. Filev, Essentials of Fuzzy Modeling and Control, Wiley, New York, 1994.