Available online at www.sciencedirect.com
SCIENCE DIRECT®
Fuzzy Sets and Systems 143 (2004) 311–333
FUZZY sets and systems
www.elsevier.com/locate/fss

A GA-based neural fuzzy system for temperature control

Cheng-Jian Lin*

Department of Information and Communication Engineering, Chao-Yang University of Technology, Gifeng E. Road, 168 Taichung County, Wufeng, Taiwan 413, ROC

Received 18 September 2000; received in revised form 19 December 2002; accepted 11 March 2003

Abstract

This paper addresses the structure and an associated learning algorithm of a feedforward multilayered connectionist network for realizing the basic elements and functions of a traditional fuzzy logic controller. The genetic algorithm-based neural fuzzy system (GA-NFS) is based on Takagi–Sugeno–Kang (TSK) type model possessing a neural network's learning ability. A hybrid learning algorithm is proposed for parameters learning. The proposed algorithm combines the genetic algorithm (GA) and the least-squares estimate (LSE) method to construct the GA-NFS. The genetic algorithm is used to tune membership functions at the precondition part of fuzzy rules while the LSE method is used to tune parameters at the consequent part of fuzzy rules. The performance of the GA-NFS is compared to that of the traditional PID controller and fuzzy logic controller on the water bath temperature control system.

Keywords: Fuzzy system; Temperature control; TSK fuzzy rules; Genetic algorithm; Least-squares estimate

1. Introduction

Temperature control is an important factor in many process control system [12,28]. If the temperature is too high or too low, the final product is seriously affected. Therefore, it is necessary to reach some desired temperature points quickly and avoid large overshoot. Since the process-control system are often nonlinear and tend to change in an unpredictable way, they are not easy to control accurately.

Fuzzy logic has been mainly applied to control problems with fuzzy if–then rules [15,24]. In most fuzzy control systems fuzzy if–then rules were derived from human experts. Recently, several approaches have been proposed for generating fuzzy if-then rules from numerical data [25,27,30].

* Fax: +886-43742375.
E-mail address: cjlin@mail.cyut.edu.tw (C.J. Lin).

0165-0114/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0165-0114(03)00126-X

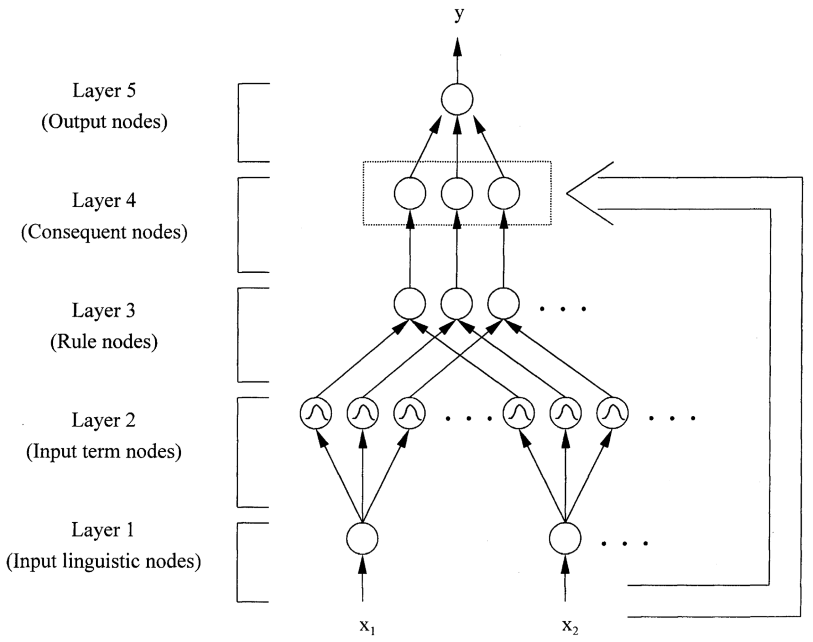

Fig. 1. Proposed GA-based neural fuzzy system (GA-NFS).

Self-learning methods have been also proposed for adjusting membership functions of fuzzy set in fuzzy if–then rules [2,7,9,14,18,19,21,26,31,33].

A genetic algorithm (GA) [3] is a parallel, global search technique that emulates operators. Because it simultaneously evaluates many points in the search space, it is more likely to converge toward the global solution. A GA applies operators inspired by the mechanics of natural selection to a population of binary string encoding the parameter space at each generation, it explores different areas of the parameter space, and then directs the search to regions where there is a high probability of finding improved performance. By working with a population of solutions, the algorithm can effectively seek many local minima, thereby increasing the likelihood of finding the global minimum.

GAs have been also employed for generating fuzzy rules and adjusting membership functions of fuzzy sets. The pioneer was Karr [10,11] who used GAs to adjust membership functions. Nomura et al. [22] used GAs to determine the fuzzy partition of input spaces. Hence, both the number of fuzzy sets and the membership function of each fuzzy set were determined. In Thrift [29], an appropriate fuzzy set in the consequent part of each fuzzy rule was selected. He examines the feasibility of using GAs to find fuzzy rules. Lee and Takagi [16] have also used GA to approach simultaneous membership function and rule set design. The GA was given more flexibility in designing the rule set and membership functions for some difficult nonlinear control problems. Ishibuchi [6] proposed a genetic-based method for selecting a small number of significant fuzzy rules to construct a compact fuzzy classification system with high classification power. Homaifar [4] examined the applicability of GAs in the simultaneous design of membership functions and rule sets for fuzzy logic controllers.

In this paper, a GA-based neural fuzzy system (GA-NFS) is based on Takagi–Sugeno–Kang (TSK) type model possessing a neural network's learning ability (see Fig. 1). The structure of the GA-NFS

model is completely determined in advance by determining the number of membership functions along each axis and choosing a grid-type partition of the premise space initially. Associated with the GA-NFS is a hybrid parameter learning algorithm that combines the GA and the least-squares estimate (LSE) method to identify parameters of the fuzzy logic rules. Inspired by the strong search ability of GAs, we let a GA to find the precondition part of fuzzy rules and leave the consequent part of fuzzy rules to the conventional optimization methods. The parameters at consequent part of fuzzy rules are the estimation of the parameters. Note in this case the estimation problem becomes linear in the parameters, so some well-established linear search algorithms will be described in [8,9]. Here the LSE method is preferred because of its recursive and fast convergent properties. Thus the GA is used to tune membership functions at the precondition part of fuzzy rules while the LSE method is used to tune parameters at consequent part of fuzzy rules.

This paper is organized as follows: Section 2 describes the structure of the GA-NFS model. The hybrid learning algorithm which combines GAs and LSE method is presented in Section 3. In Section 4, the GA-NFS model is used to control water bath temperature to demonstrate its learning capability. Comparisons with PID controller and fuzzy controller are also made. Section 5 describes the features of the proposed hybrid learning algorithm. We summarize the features of the proposed GA-NFS model in Section 5. Conclusions are summarized in the last section.

2. The structure of the GA-NFS model

In this section, we shall describe the structure and functions of the proposed GA-NFS model. The GA-NFS (see Fig. 1) has five layers with node and link numbering defined by the brackets on the right-hand side of the figure. Nodes at layer 1 are input nodes (linguistic nodes) that represent input linguistic variables. Layer 5 is the output layer. Nodes at layer 2 are term nodes that act as membership functions to represent the terms of the respective linguistic variable. Each node at layer 3 is a rule node, which represents one fuzzy logic rule. Thus all layer three nodes form a fuzzy rule base. In layer 4 nodes, the consequent part of the Takagi–Sugeno–Kang (TSK) model [27] is used. The TSK model can represent a complex system in terms of fewer rules than the ordinary Mamdani-type fuzzy model [27]. Layer 3 links define the preconditions of the rule nodes. The links at layer 2 are fully connected between linguistic nodes and their corresponding term nodes.

A typical neural network consists of nodes with some finite number of fan-in connections from other nodes represented by weight values, and fan-out connections to other nodes. Associated with the fan-in of a node is an integration function f which combines information, activation, or evidence from other nodes, and provides the net input; i.e.,

$$\text{net-input} = f(z_1^{(k)}, z_2^{(k)}, \ldots, z_p^{(k)}; w_1^{(k)}, w_2^{(k)}, \ldots, w_p^{(k)}), \tag{1}$$

where $z_i^{(k)}$ is the ith input to a node in layer k, and $w_i^{(k)}$ is the weight of the associated link. The superscript in the above equation indicates the layer number. This notation will be also used in the following equations. Each node also outputs an activation value as a function of its net-input,

$$\text{output} = a(f(\cdot)), \tag{2}$$

where $a(\cdot)$ denotes the activation function. We shall next describe the functions of the nodes in each of the five layers of the GA-NFS. Assume that the dimension of the input space is n, and that of the output space is m.

Layer 1: Each node in this layer is called an input linguistic node and corresponds to one input linguistic variable. Layer-1 nodes just transmit input signals to the next layer directly. That is,

$$f(x_i) = x_i \quad \text{and} \quad a(f(\cdot)) = f(\cdot). \tag{3}$$

From the above equation, the link weight in layer 1 $()w_i^{(1)}$ is unity.

Layer 2: Nodes in this layer are called input term nodes and each represents a term of an input linguistic variable. In other words, the membership value that specifies the degree to which an input value belongs to a fuzzy set is calculated in Layer 2. With the use of Gaussian membership function, the operation performed in this layer is

$$f(z_{ij}^{(2)}) = \exp\left\{ -\left(\frac{z_{ij}^{(2)} - m_{ij}}{\sigma_{ij}} \right)^2 \right\} \quad \text{and} \quad a(f(\cdot)) = f(\cdot), \tag{4}$$

where m_{ij} and σ_{ij} are the center (or mean) and the width (or variance) of the Gaussian membership function of the jth input term node of the ith input linguistic node.

Layer 3: Nodes in this layer are called rule nodes and each represents one fuzzy logic rule. The links in layer 3 are used to perform precondition matching of fuzzy logic rules. Hence the rule nodes perform the fuzzy AND (or product) operation,

$$f(z_i^{(3)}) = \prod_{i=1}^{n} z_i^{(3)} \quad \text{and} \quad a(f(\cdot)) = f(\cdot), \tag{5}$$

where $z_i^{(3)}$ is the ith input to a node in layer 3 and the product is over the inputs of this node. The link weight in layer 3 $(w_i^{(3)})$ is then unity.

Layer 4: Nodes in this layer are called the consequent nodes. The input to a node of Layer 4 is the output delivered from Layer 3, and the other inputs are the input variables from Layer 1 as depicted in Fig. 1. For this kind of node, we have

$$f(z_i^{(4)}, x_j) = \left(\sum_j p_{ji} x_j \right) \cdot z_i^{(4)} \quad \text{and} \quad a(f(\cdot)) = f(\cdot), \tag{6}$$

where the summation is over all the inputs and p_{ji} is the corresponding parameter of consequent part.

Layer 5: Each node in this layer corresponds to one output variable. The node integrates all the actions recommended by Layers 3 and 4 and acts as a defuzzifier with

$$f(z_i^{(5)}) = \sum_i z_i^{(5)} \quad \text{and} \quad a(f(\cdot)) = \frac{f(\cdot)}{\sum_j z_j^{(4)}}. \tag{7}$$

Based on the above structure, a hybrid learning algorithm will be proposed to determine the proper parameters for each node in layers 2 and 4.

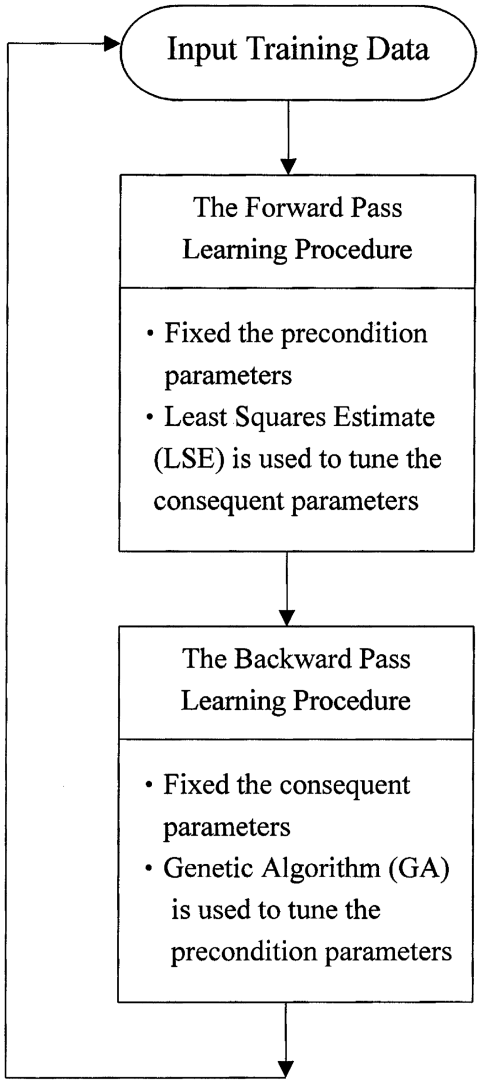

Fig. 2. The flowchart of the hybrid learning algorithm for the GA-NFS model.

3. Learning algorithms for GA-NFS model

In this section, we present a hybrid learning scheme for the proposed GA-NFS model. Fig. 2 illustrates the flowchart of the hybrid learning algorithm for the GA-NFS model. The proposed hybrid learning algorithm performs only parameter learning of the fuzzy model. The structure of the GA-NFS controller is completely determined in advance by determining the number of memberships along each axis and choosing a grid-type partition of the premise space. That is, the number of rules is the product of the memberships along the input variables (full interconnection between layers 2 and 3). This hybrid learning algorithm combines GA and LSE method to identify parameters of

the fuzzy logic rules. The GA is used to tune membership functions at the precondition part of fuzzy rules, while the LSE method is used to tune parameters at consequent part of fuzzy rules. The hybrid learning algorithm is composed of a forward pass learning procedure and a backward pass learning procedure. In the forward pass learning procedure, we supply input data and functional signals go forward to calculate each node output and the parameters at the consequent part of fuzzy rules are tuned by LSE method. After identifying parameters at the consequent part of rules, the functional signals keep going forward till the root-mean-square error is calculated. In the backward pass learning procedure, the root-mean-square error is then converted to a fitness function, and the membership functions at the precondition part of fuzzy rules are updated by the GA.

3.1. The forward pass learning procedure

Even though such a basic fuzzy model [24] can be used directly for system modeling, a large number of rules are necessary for modeling sophisticated system under a tolerable modeling accuracy. To cope with this problem, we adopt the spirit of TSK model [27] into the GA-NFS. In the TSK model, each consequent part is represented by a linear equation of the input variables. It is reported in [27] that the TSK model can model a sophisticated system using a few rules. During the first pass the consequent parameters are estimated by LSE while the first pass GA-based corrections are applied to premise parameters. However, the GA determines the premise parameters since these parameters are only encoded in the chromosome of the genotypes. Then, given the premise parameter for all the individuals, the consequent coefficients are attained through LSE method. The consequent parameters thus identified are optimal under the condition that the precondition parameters are fixed. The parameters p_{ji} in Layer 4 are tuned by recursive squares algorithm [9].

$$P(t+1) = P(t) + S(t+1)Z^{(4)}(t+1)[y^{\mathrm{d}}(t) - y(t)],$$
$$S(t+1) = \frac{1}{\lambda}\left[S(t) - \frac{S(t)Z^{(4)^{\mathrm{T}}}(t+1)Z^{(4)}(t+1)S(t)}{\lambda + Z^{(4)^{\mathrm{T}}}(t+1)S(t)Z^{(4)}(t+1)}\right], \tag{8}$$

where $0<\lambda\leqslant 1$ is the forgetting factor, $Z^{(4)}$ is the current input vector, P is the corresponding parameter vector, S is the covariance matrix, y^{d} is the desired output, and y is the current output. If the precondition parameters (i.e., membership functions) are fixed and only the consequent parameters are adjusted, the GA-NFS can be viewed as a functional-link network [13].

3.2. The backward pass learning procedure

In this learning scheme, we fixed the consequent parameters of fuzzy rules, and then the GA is used to find proper membership functions according to the desired input and output pair. The following steps are employed to tune the membership functions by the GA.

Step 1. *Initialization*: The first step in GAs is coding which maps a finite-length string to the searched parameters. We first generate an initial population containing N_{pop} strings, where N_{pop} is the number of strings in each population. Each string is an individual point in the search space. The coding of the precondition parameter in a chromosome is shown in Fig. 3(a). A Gaussian

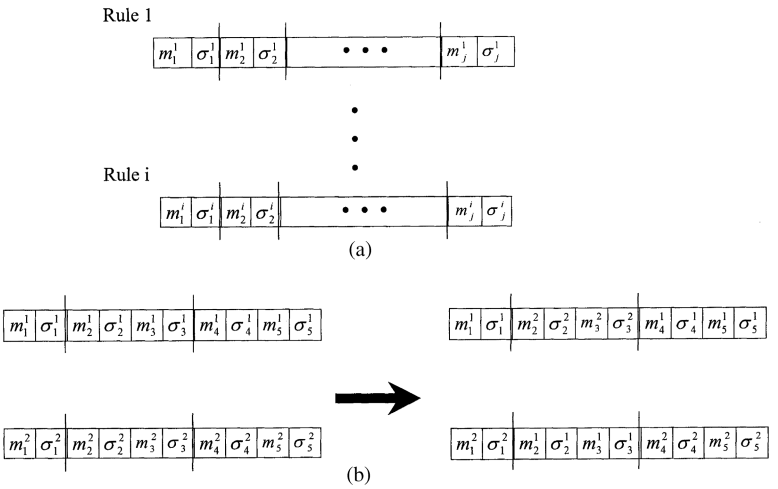

Fig. 3. (a) Coding the precondition parameter of a fuzzy rule into a chromosome. (b) Crossover operation on the rule.

membership function is used with variables m and σ representing the center and width of the membership function, i.e., the rule has the form of

$$\text{IF } x_1 \text{ is } \mu(m_1, \sigma_1) \text{ and } x_2 \text{ is } \mu(m_2, \sigma_2) \text{ and} \ldots \text{and } x_j \text{ is } \mu(m_j, \sigma_j)$$

$$\text{THEN } y = p_0 + p_1 x_1 + \cdots + p_j x_j.$$

A small population is used in our learning scheme. The use of small population reduces the exploration of the multiple (representationally dissimilar) solutions for the same network. The real-valued coding scheme is used in the GA-NFS model. This means that each string is generated by randomly assigning real-valued to each possible parameters of the GA-NFS model. Each string thus represents a set of possible parameters. From the whole parameters space, a population is chosen.

Step 2. *Fitness function*: In this step, each string is decoded by an evaluator into an objective function value. This function value, which should be maximized by the GA, is then converted to a fitness value, $FIT(i)$, where $FIT(\cdot)$ is a fitness function. A fitness value is assigned to each individual in the population, where high values mean good fit. The fitness function can be any nonlinear, nondifferentiable, or discontinuous positive function, because the GA only needs a fitness value assigned to each string. In this paper, the fitness function is defined by

$$FIT(i) = \frac{1}{RMS\_ERROR(i)}, \tag{9}$$

where $RMS\_ERROR(i)$ represents the root-mean-square error between the network outputs and the desired outputs for the ith string. The goal of the GA learning phase is to maximize the above fitness function.

Step 3. *Reproduction*: Reproduction is a process in which individual strings are copied according to their fitness values, i.e., based on the principle of survival of the fittest. This operator is an artificial version of natural selection. Through reproduction, strings with high fitnesses receive multiple copies in the next generation while strings with low fitnesses receive fewer copies or even none at all.

Step 4. *Crossover*: Reproduction directs the search toward the best existing individuals but does not create any new individuals. In nature, an offspring is rarely an exact clone of a parent. It usually has two parents and inherits genes from both. The main operator in GAs to work on the parents is crossover, which occurs with a crossover probability. The crossover operation can be generalized to multipoint crossover in which the number of crossover point N_c is defined. With N_c set to 1, generalized crossover reduces to simple crossover. The multipoint crossover can solve one major problem of the simple crossover; one point crossover cannot combine certain combinations of features encoded on chromosomes. In this paper, we use the two-point crossover operator. At first, two strings from the reproduced population are mated at random, and two crossover points are randomly selected. Then the strings are crossed and separated at these points shown in Fig. 3(b).

Step 5. *Mutation*: Even though reproduction and crossover come up with many new strings, they do not introduce any new information into the population at the bit level. Mutation is the random alteration of bits in the string that assists in keeping diversity in the population. Since we use the real-value coding scheme, we use a higher mutation probability p_m. This is different from the traditional GAs using the binary-value coding scheme, that is largely driven by recombination, not mutation [32]. The mutation operator serves as a means to avoid local minima in the search space.

Step 6. If the stopping condition is not satisfied (i.e., the current smallest root-mean-square error is not small enough), return to Step 2. Otherwise, the GA is terminated and proper parameters are obtained by encoding the best string into a set of parameters.

4. Control of water bath temperature system

The goal of this section is to control the temperature of a water bath system given by

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{RC}, \tag{10}$$

where $y(t)$ is system output temperature in °C, $u(t)$ is heating flowing inward the system, Y_0 is room temperature, C is the equivalent system thermal capacity, and R is the equivalent thermal resistance between the system borders and surroundings.

Assuming that R and C are essentially constant, we rewrite the system in Eq. (10) into discrete-time form with some reasonable approximation. The system

$$y(t+1) = \mathrm{e}^{-\alpha T_s} y(k) + \frac{\beta/\alpha(1 - \mathrm{e}^{-\alpha T_s})}{1 + \mathrm{e}^{0.5y(k)-40}} u(k) + [1 - \mathrm{e}^{-\alpha T_s}] Y_0 \tag{11}$$

is obtained, where α and β are some constant values describing R and C. The system parameters used in this example are $\alpha = 1.0015\mathrm{e}^{-4}, \beta = 8.67973\mathrm{e}^{-3}$ and $Y_0 = 25.0$ (°C), which were obtained from a real water bath plant in [28]. The plant input $u(t)$ is limited between $0v$ and $5v$ where v represents voltage unit. The sampling period is $T_s = 30$. The system configuration is shown in Fig. 4, where y_{ref} is the desired temperature of the controlled plant.

In this paper, we compare the GA-NFS controller to the PID controller and the manually designed fuzzy controller. Each of the three controllers is applied to the water bath temperature control system. The comparison performance measures include set-points regulation, ramp-points tracking, the influence of impulse noise, and a large parameter variation in the system.

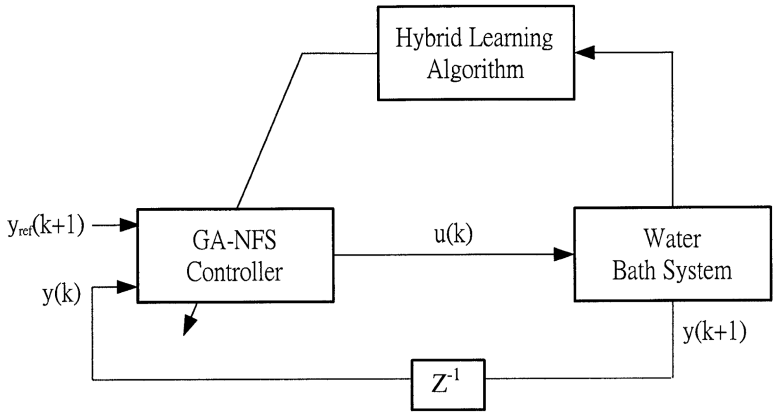

Fig. 4. Flow diagram of using GA-NFS controller for solving temperature control problem.

For the PID control, a velocity-form discrete PID controller [1] is used and is described by

$$\Delta u(k) = K\left\{e(k) - e(k-1) + \frac{T_s}{2T_i}[e(k) + e(k-1)] + \frac{T_d}{T_s}[e(k) - 2e(k-1) + e(k-2)]\right\}$$
$$= K_P[e(k) - e(k-1)] + K_I e(k) + K_D[e(k) - 2e(k-1) + e(k-2)], \quad (12)$$

where

$$K_P = K - \frac{1}{2}K_I, \quad K_I = \frac{KT_s}{T_i}, \quad K_D = \frac{kT_d}{T_s}.$$

The parameter $\Delta u(k)$ is the increment of the control input, $e(k)$ is the performance error at the sampling instant k, and K_P, K_I, and K_D are the proportional, integral, and derivative parameters, respectively. In order not to aggravate noise in the plant, only a two-term PID controller is used, i.e., K_D is set to zero in the water bath system. The other two parameters K_P and K_I are set as 80 and 70, respectively.

For the manually designed fuzzy controller, the input variables are chosen as $e(t)$ and $ce(t)$, where $e(t)$ is the performance error indicating the error between the desired water temperature and the actual measured temperature and $ce(t)$ is the rate of change in the performance error $e(t)$. The output or the controlled linguistic variable is the voltage signal $u(t)$ to the heater. Seven fuzzy terms are defined for each linguistic variable. These fuzzy terms consist of negative large (NL), negative medium (NM), negative small (NS), zero (ZE), positive small (PS), positive medium (PM), and positive large (PL). Each fuzzy term is specified by a Gaussian membership function. According to common sense and engineering judgment, 25 fuzzy rules are specified in Table 1. Like other controllers, a fuzzy controller has some scaling parameters to be specified. They are GE, GCE, and GU, corresponding to the process error, the change in error, and the controller's output, respectively. We choose these parameters as follows: $GE = 1/15, GCE = 1/15, GU = 450$.

For the aforementioned controllers (GA-NFS controller, PID controller and manually designed fuzzy controller), four groups of computer simulations are conducted on the water bath temperature control system. Each simulation is performed over 120 sampling time steps.

Table 1
Fuzzy rules for the water bath temperature control system

Change in error $e(k)$	Error $e(k)$						
	NL	NM	NS	ZE	PS	PM	PL
NL				PL			
NM				PM			
NS			PS	PS	PM		
ZE	NL	NM	NS	ZE	PS	PM	PL
PS			NS	NS	PS	PM	PL
PM				LM	PS	PM	PL
PL				NL	PS	PL	PL

In the first set of simulations, the regulation capability of the three controllers with respect to set-point changes is studied. Three set-points to be followed are

$$y_{\mathrm{ref}}(k) = \begin{cases} 35^{\circ}\mathrm{C} & \text{for } k \leqslant 40, \\ 55^{\circ}\mathrm{C} & \text{for } 40 < k \leqslant 80, \\ 75^{\circ}\mathrm{C} & \text{for } 80 < k \leqslant 120. \end{cases} \tag{13}$$

The 120 training patterns are chosen from the input–output characteristic in order to cover the entire reference output space. According to the selected training patterns, the GA-NFS controller with hybrid learning algorithm is trained. The GA-NFS used here contains nine rules. Fig. 6(a) illustrates the distribution of the training patterns and the initial of fuzzy rules (i.e., distribution of input membership functions) in the $[y(k), y(k+1)]$ plain. The boundary of each ellipse represents a rule with firing strength 0.5. The population size $N_{\mathrm{pop}} = 100$, mutation probability $P_{\mathrm{m}} = 0.1$, and the two-point crossover operator are used. The mean number of generations in this GA learning phase is about 350 generations. The learning result is shown in Fig. 5. Fig. 6(b) illustrates the distribution of the training patterns and the final assignment of fuzzy rules in the $[y(k), y(k+1)]$ plain. In this figure, the membership functions exhibit a high degree of overlapping between each other. Many papers [9,17,19] have used fuzzy similarity measure method to determine the similarity between two fuzzy sets in order to avoid the existing membership functions being too similar.

After training, the regulation performance of the GA-NFS controller is shown in Fig. 7. The real line represents the actual output while the dash line represents the reference output. Figs. 8 and 9 show the regulation performance through the PID controller and fuzzy controller. The regulation errors between the reference output and the actual output of the GA-NFS controller, the PID controller, and the fuzzy controller are shown in Fig. 10. The GA-NFS controller operates much smaller overshoot in achieving the set-points than the PID and fuzzy controllers. With regard to the steady-state error of set-points, the GA-NFS controller also shows the smaller error. The training set comprise a sufficient variety of random step changes, so that after training the controller is able to respond satisfactorily to an arbitrary reference profile. The GA-NFS controller has also been tested on the four set-point regulation problem providing a good performance. To test their regulation performance,

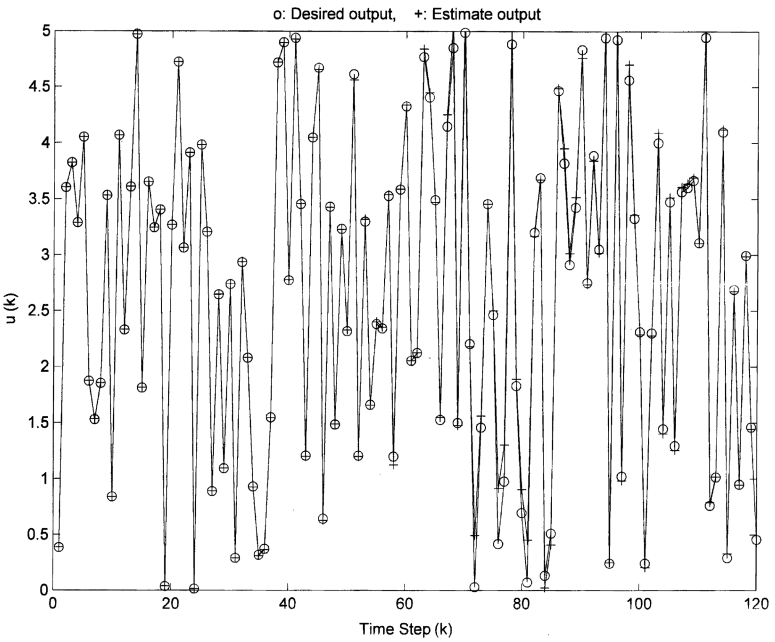

Fig. 5. Simulation results of using the GA-NFS controller in the training data set.

a performance index, sum of absolute error (SAE), is defined by

$$SAE = \sum_{k} |y_{\text{ref}}(k) - y(k)|, \tag{14}$$

where $y_{\text{ref}}(k)$ and $y(k)$ are the reference output and the actual output of the simulated system, respectively. The SAE values of the GA-NFS controller, the PID controller, and the fuzzy controller are 353.5, 418.5, and 401.5, which are shown in the first column of Table 2.

In the second set of simulations, the tracking capability of the three controllers with respect to ramp-reference signals is studied. We define

$$y_{\text{ref}}(k) = \begin{cases} 34^{\circ}\text{C} & \text{for } k \leqslant 30, \\ (34 + 0.5(k - 30))^{\circ}\text{C} & \text{for } 30 < k \leqslant 50, \\ (44 + 0.8(k - 50))^{\circ}\text{C} & \text{for } 50 < k \leqslant 70, \\ (60 + 0.5(k - 70))^{\circ}\text{C} & \text{for } 70 < k \leqslant 90, \\ 70^{\circ}\text{C} & \text{for } 90 < k \leqslant 120. \end{cases} \tag{15}$$

For the GA-NFS controller, the same training scheme, training data and learning parameters are used as those used in the first set of simulations. The tracking performances of the GA-NFS controller, the PID controller, and the fuzzy controller are shown in Figs. 11–13. The tracking errors of the three controllers are shown in Fig. 14. As shown in the error curves, the GA-NFS controller exhibits the smallest error. The PID and fuzzy controllers show poor tracking-control capability. The

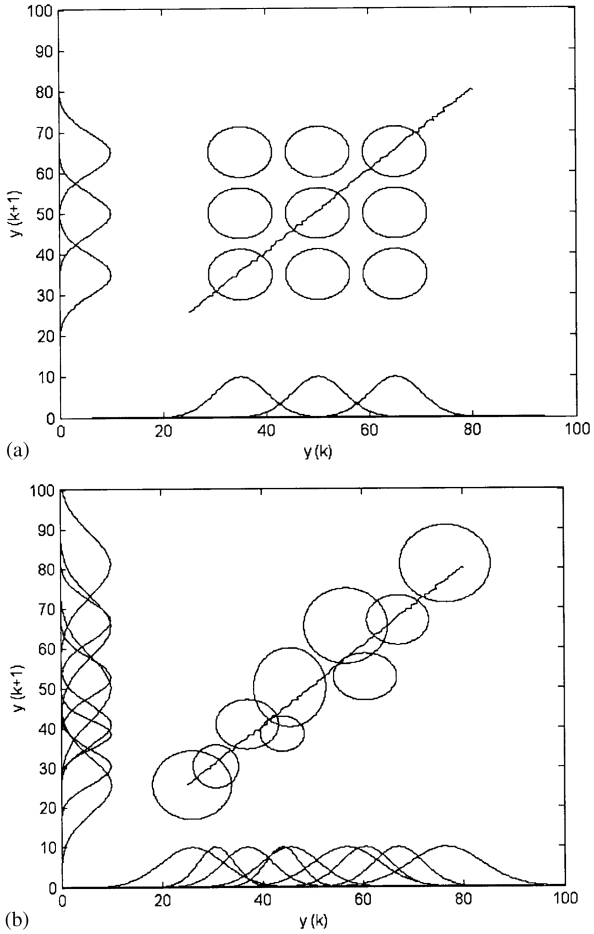

Fig. 6. (a) The input training patterns and the initial assignment of rules. (b) The input training patterns and the final assignment of rules.

SAE values of the GA-NFS controller, the PID controller, and the fuzzy controller are 37.1, 100.6 and 68.1, which are shown in the second column of Table 2.

The third set of simulations is carried out for the purpose of studying the noise-rejection ability of the three controllers when some unknown impulse noise is imposed on the process. One impulse noise value $-5°C$ is added to the plant output at the 60th sampling instant. A set-point of 50°C is performed in this set of simulations. For the GA-NFS controller, the same training scheme, training data and learning parameters are used as those used in the first set of simulations. The behaviors of the GA-NFS controller, the PID controller, and the fuzzy controller under the influence of impulse noise are shown in Figs. 15–17. The corresponding errors of the three controllers are shown in Fig. 18. The SAE values of the GA-NFS controller, the PID controller, and the fuzzy controller are 260.2, 311.5, and 275.8, which are shown in the third column of Table 2. It is observed that the GA-NFS controller performs quite well. It recovers very quickly and steadily after the presentation of the impulse noise. However, the PID controller is affected seriously by the impulse noise.

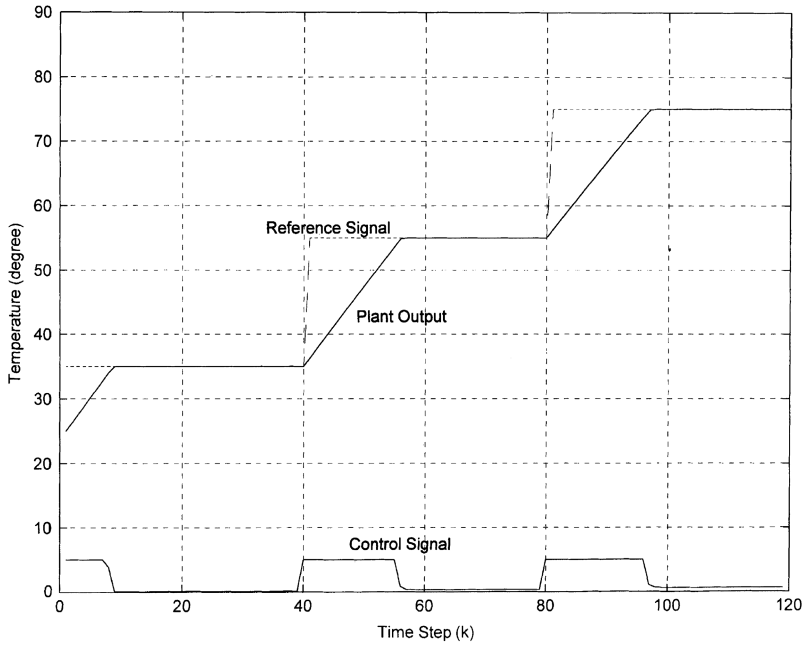

Fig. 7. The regulation performance of the GA-NFS controller for the water bath system.

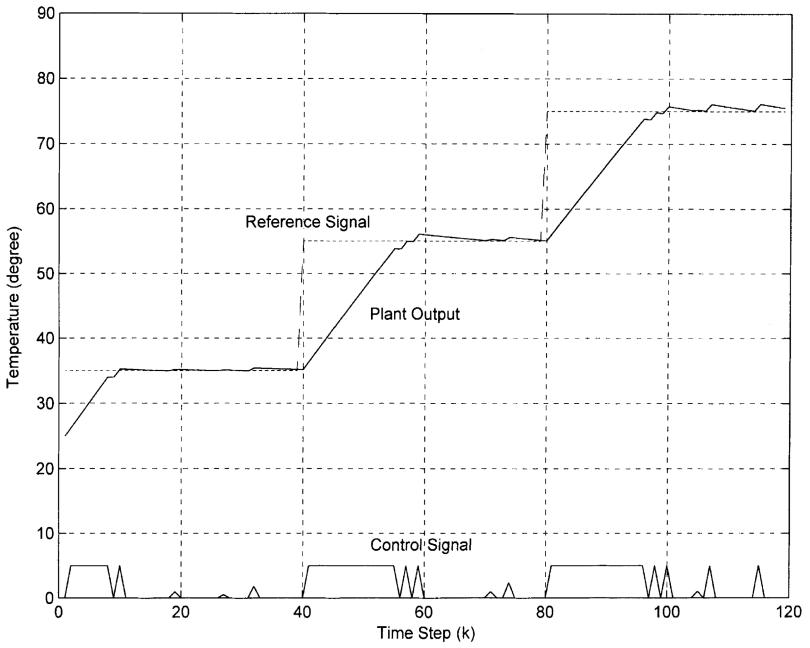

Fig. 8. The regulation performance of the PID controller for the water bath system.

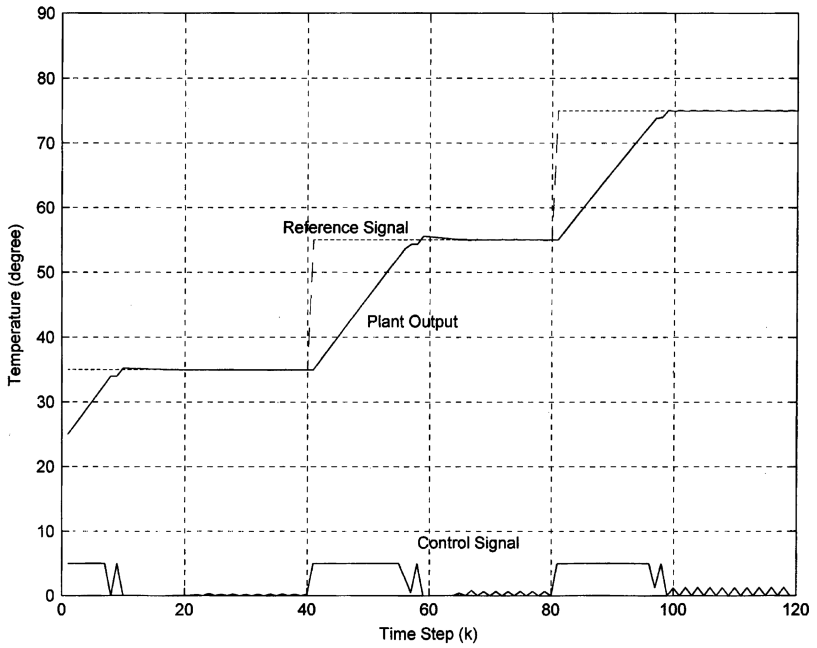

Fig. 9. The regulation performance of the manually designed fuzzy controller for the water bath system.

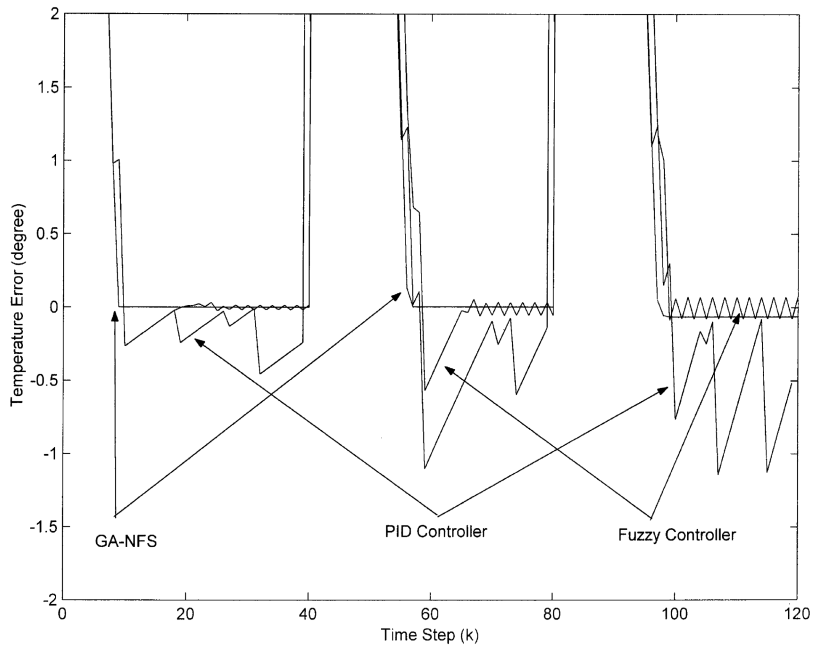

Fig. 10. The corresponding errors between the reference output and the actual output by using GA-NFS, PID, and fuzzy controllers.

Table 2
Performance comparison of various controllers

	GA-NFS controller	PID controller	Fuzzy controller	ANFIS controller
Regulation performance	353.5	418.5	401.5	361.5
Tracking performance	37.1	100.6	68.1	39.2
Influence of impulse noise	260.2	311.5	275.8	262.1
Effect of change in plant dynamics	258.9	322.2	273.5	259.3

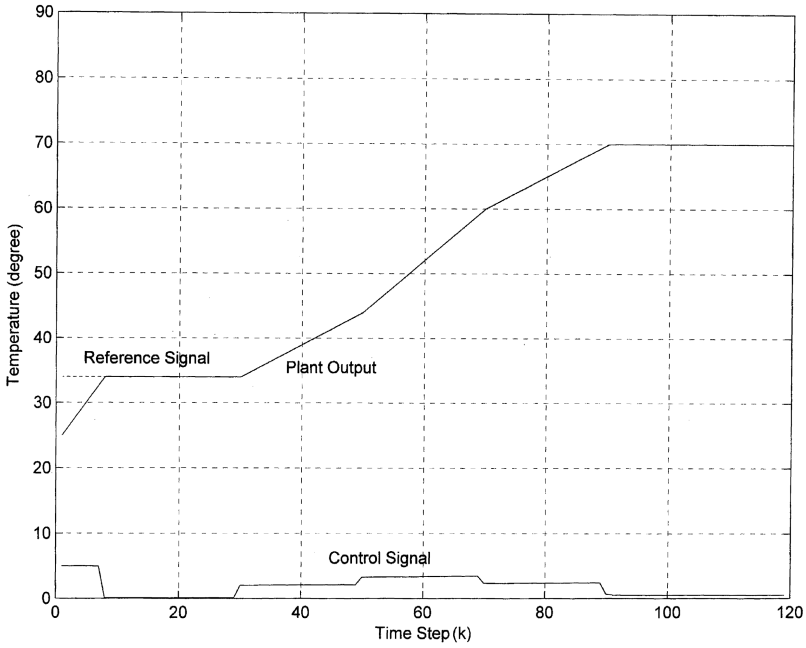

Fig. 11. The tracking performance of the GA-NFS controller for the water bath system.

One common characteristic of many industrial control processes is that their parameters tend to change in an unpredictable way. To test the robustness of the three controllers, a value of $0.7u(k-2)$ is added to the plant input after the 60th sample in the fourth set of simulations. A set-point of 50°C is used in this set of simulations. For the GA-NFS controller, the same training scheme, training data and learning parameters are used as those used in the first set of simulations. The behaviors of the GA-NFS controller, the PID controller, and the fuzzy controller when there is a change in the plant dynamics are shown in Figs. 19–21. The corresponding errors of the three controllers are shown in Fig. 22. The SAE values of the GA-NFS controller, the PID controller, and the fuzzy controller are 258.9, 322.2, and 273.5, which are shown in the fourth column of Table 2. The PID controller is affected more seriously after the 60th sample. The results show the good control and disturbance rejection capabilities of the trained GA-NFS in the water bath system.

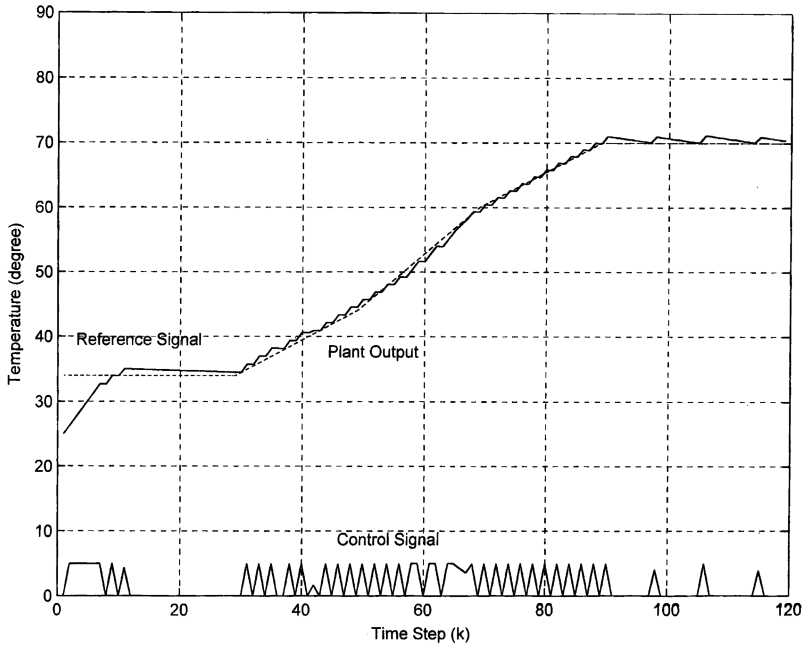

Fig. 12. The tracking performance of the PID controller for the water bath system.

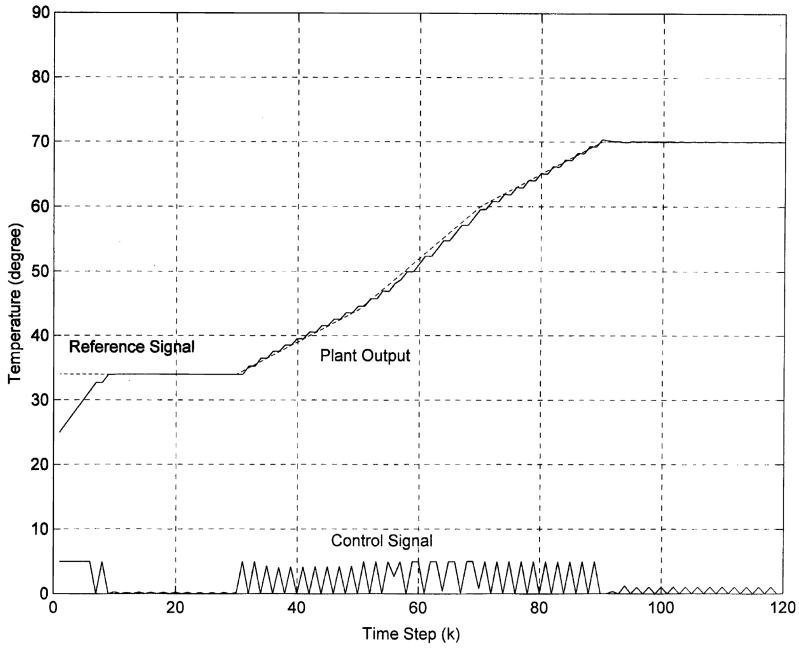

Fig. 13. The tracking performance of the manually designed fuzzy controller for the water bath system.

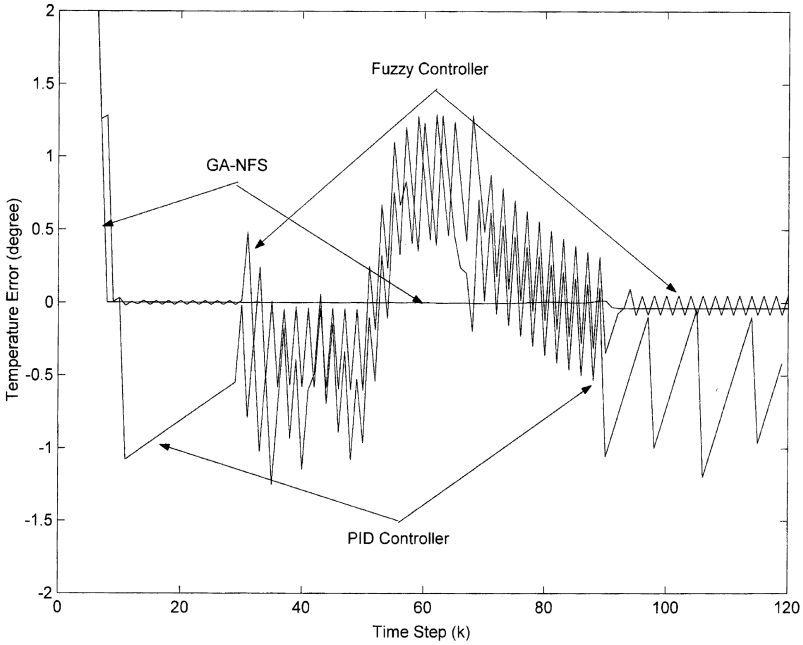

Fig. 14. The corresponding errors between the reference output and the actual output by using GA-NFS, PID, and fuzzy controllers.

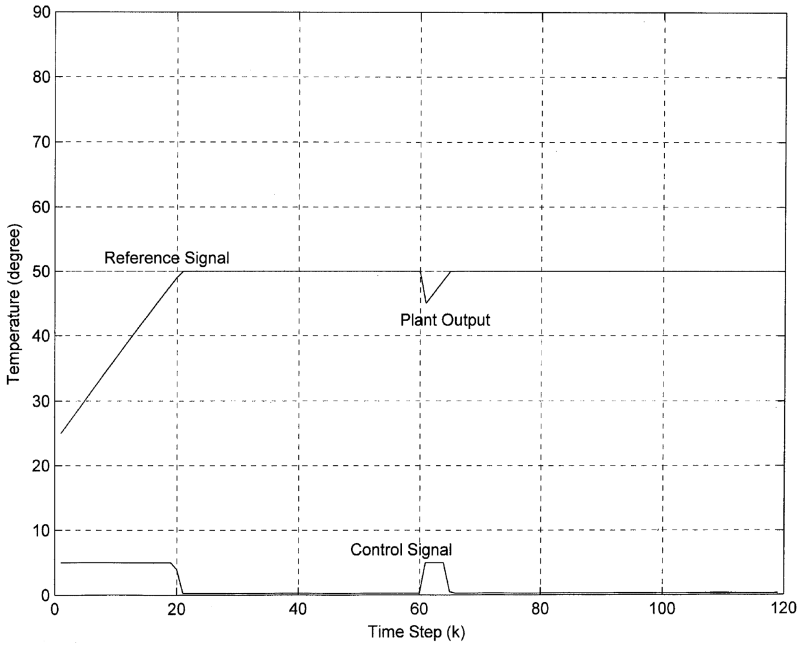

Fig. 15. The behavior of the GA-NFS controller under the impulse noise for the water bath system.

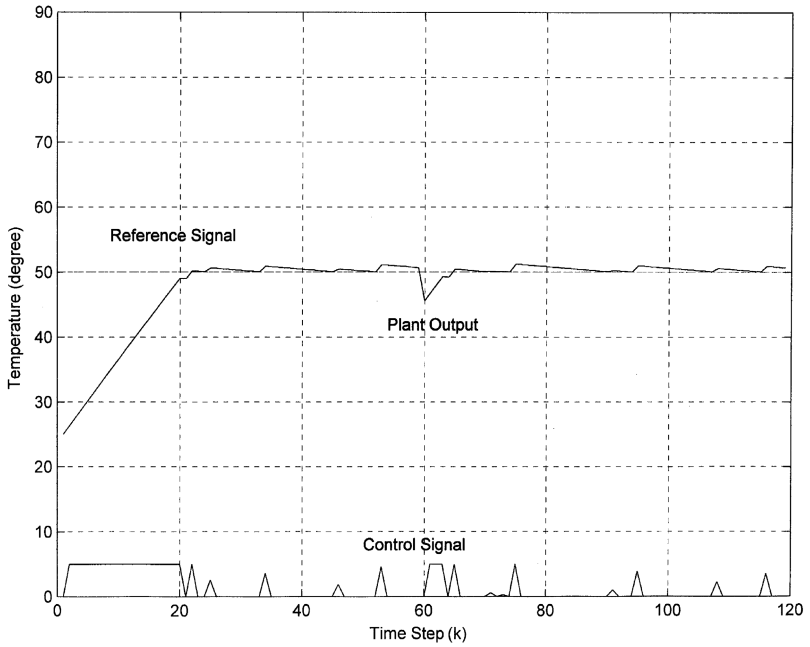

Fig. 16. The behavior of the PID controller under the impulse noise for the water bath system.

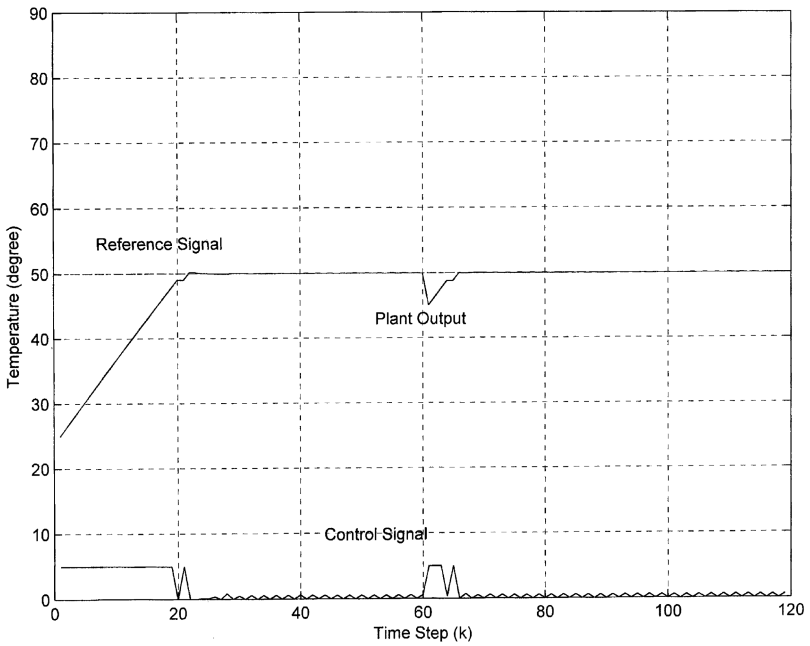

Fig. 17. The behavior of the manually designed fuzzy controller under the impulse noise for the water bath system.

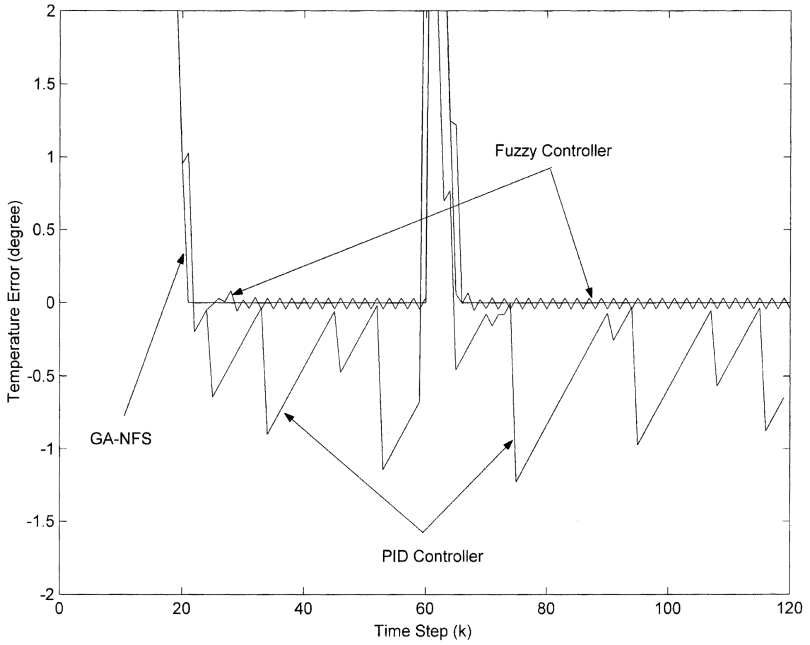

Fig. 18. The corresponding errors between the reference output and the actual output by using GA-NFS, PID, and fuzzy controllers.

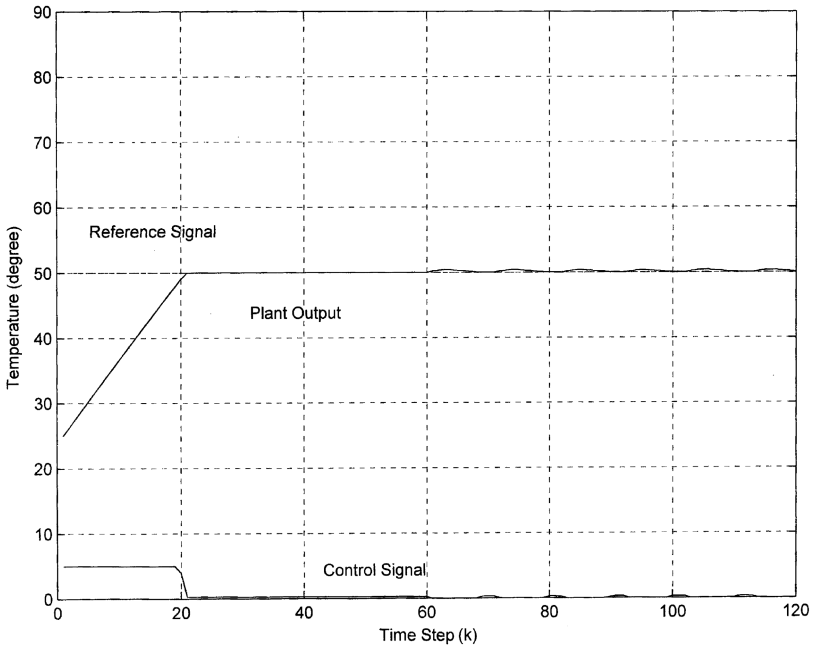

Fig. 19. The behavior of the GA-NFS controller when a change occurs in the water bath system dynamics.

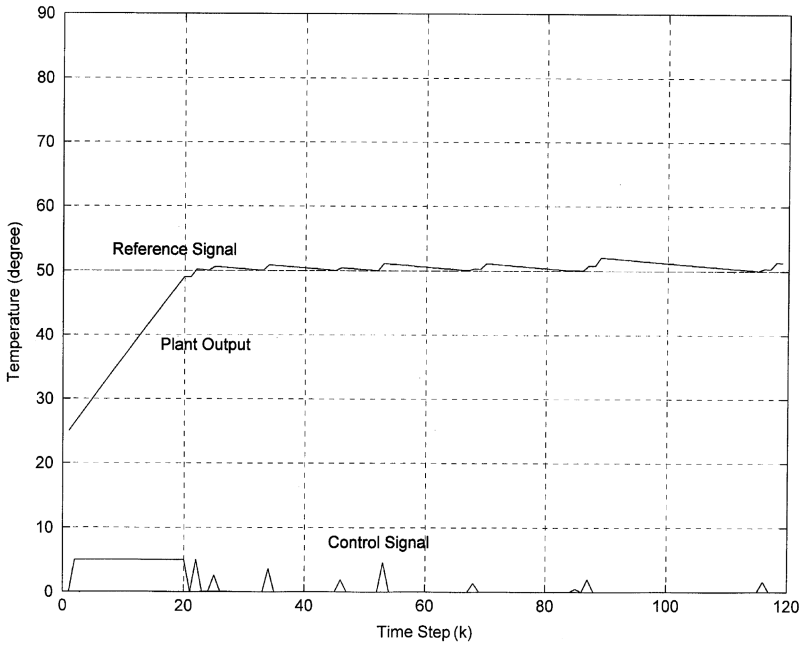

Fig. 20. The behavior of the PID controller when a change occurs in the water bath system dynamics.

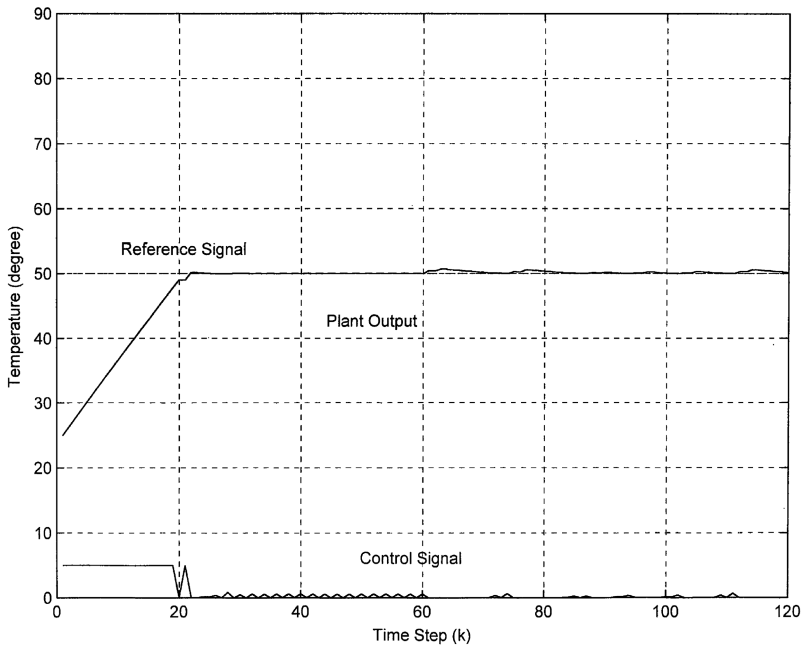

Fig. 21. The behavior of the manually designed fuzzy controller when a change occurs in the water bath system dynamics.

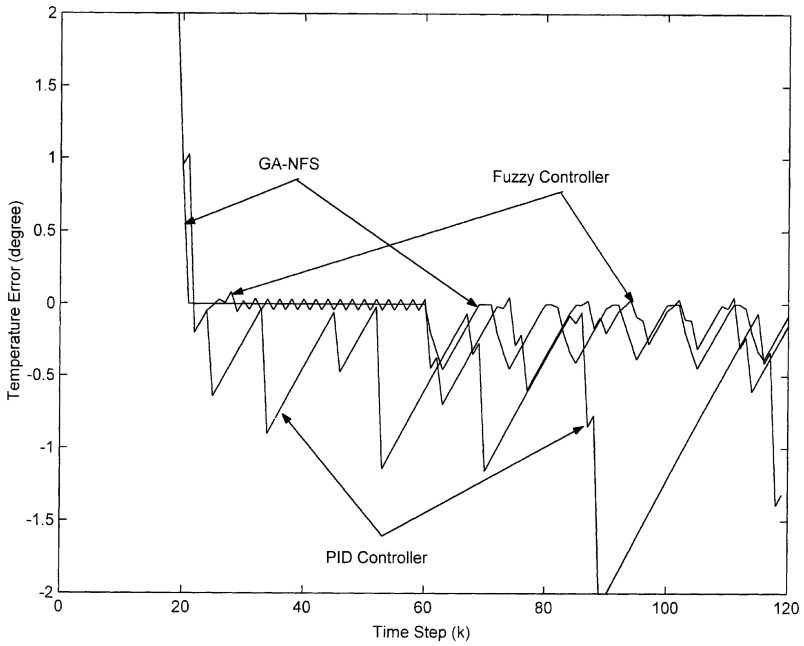

Fig. 22. The corresponding errors between the reference output and the actual output by using GA-NFS, PID, and fuzzy controllers.

Jang [8] presented an adaptive-network-based fuzzy inference system (ANFIS), which is a fuzzy inference system implemented in the framework of adaptive networks. Through the learning procedure, the proposed ANFIS can construct an input–output mapping based on input–output data pairs. The structure of the ANFIS is fixed and the parameter identification is solved through the backpropagation algorithm. We also compare the GA-NFS controller to the ANFIS model. There are nine rules generated in the ANFIS model. These results are shown in the fifth column of Table 2. The results show that the GA-NFS controller is better than the ANFIS controller.

5. Discussions

In this section, we summarize the features of the proposed GA-NFS model. First, a real-value encoding is used. This means that each parameter is represented by a single real value and that recombination can only occur between parameters. The real-value coding is different from the traditional GAs using the binary-value coding scheme [20], whose recombination occurs between the binary bits. Second, a much higher level of mutation is used. This is also different from the traditional GAs using the binary-value coding scheme that is largely driven by recombination, not by mutation. Since a real-value coding and a higher mutation are used in our system, the learning speed may become quicker than the traditional GAs using the binary-value coding scheme [20]. Third, we use a small population. The memory size that is required may become smaller. Finally, the proposed hybrid learning algorithm can be used to find optimal parameters under the structure is fixed in advance. The proposed learning algorithm is different from [8,9]. Jang [8] proposed a learning rule

which combines the gradient method and the LSE to identify parameters. Though the LSE method can be used to find optimal parameters, the gradient method is easy trapped in local minimum. In [9], they used the symbiotic evolution method to tune the precondition and consequent parameters of an TSK-type fuzzy rule. Since the learning speed of GA is slow, we adopted the LSE method to tune the consequent parameters in this paper.

6. Conclusions and future work

In this paper, we introduced a general connectionist model of a fuzzy logic control system called GA-NFS. A hybrid learning algorithm that combines the GA and the LSE method was proposed for constructing the GA-NFS. Simulations demonstrate that the proposed GA-NFS model has good generalization capability and robustness.

Two advanced topics on the proposed GA-NFS controller should be addressed in future researches. First, in many existing fuzzy systems and neural networks, the numbers of the fuzzy rules and hidden nodes are always determined by users in advance. In [17,18], we proposed self-constructing neural fuzzy systems, which can dynamically partition the input–output spaces, tune membership function, and find proper fuzzy rules. Thus, the proposed hybrid learning algorithm with GA and LSE method can be easily extended to these models [17,18]. Second, it would be better if the GA-NFS has the ability to delete unnecessary or redundant rules. The fuzzy similarity measure [19] determines the similarity between two fuzzy sets in order to avoid the existing membership functions being too similar.

References

[1] C.W. Anderson, Strategy learning with multilayer connectionist representations, Proc. 4th Int. Workshop on Machine Learning, Irvine, CA, June 1987, pp. 103–114.
[2] H.R. Berenji, P. Khedkar, Learning and tuning fuzzy logic controllers through reinforcements, IEEE Trans. Neural Networks 3 (5) (1992) 724–740.
[3] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[4] 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.
[5] S. Horikawa, T. Furuhashi, Y. Uchikawa, On fuzzy modeling using fuzzy neural networks with the backpropagation algorithm, IEEE Trans. Neural Networks 3 (5) (1992) 801–806.
[6] H. Ishibuchi, K. Nozaki, N. Yamamoto, H. Tanaka, Selecting fuzzy if-then rules for classification problems using genetic algorithms, IEEE Trans. Fuzzy Systems 3 (3) (1995) 260–270.
[7] J.S. Jang, Self-learning fuzzy controllers based on temporal back propagation, IEEE Trans. Neural Networks 3 (5) (1992) 723–741.
[8] J.S. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Systems Man Cybernet. 23 (3) (1993) 665–685.
[9] C.F. Juang, C.T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, IEEE Trans. Fuzzy System 6 (1) (1998) 12–32.
[10] C. Karr, Genetic algorithms for fuzzy controllers, AI Expert, February 1991, pp. 26–33.
[11] C. Karr, Applying genetics to fuzzy logic, AI Expert, March 1991, pp. 39–43.
[12] M. Khalid, S. Omatu, A neural network controller for a temperature control system, IEEE Control System 12 (1992) 58–64.

[13] M.S. Klassen, Y.H. Pao, Characteristics of the functional-link net: a higher order delta rule net, in: IEEE Proc. Conf. Neural Networks, San Diego, June 1988.
[14] B. Kosko, Neural Networks and Fuzzy Systems, Prentice-Hall, Englewood Cliffs, NJ, 1992.
[15] C.C. Lee, Fuzzy logic in control systems: fuzzy logic controller—Part I and Part II, IEEE Trans. Systems Man Cybernet. 20 (2) (1990) 404–435.
[16] M.A. Lee, H. Takagi, Integrating design stages of fuzzy systems using genetic algorithms, Proc. IEEE Int. Conf. Fuzzy Systems, San Francisco, 1993, pp. 612–617.
[17] C.J. Lin, W.H. Ho, A self-constructing compensatory neural fuzzy system and its applications, The 7th Conf. on Artificial Intelligence and Applications, Taiwan, 2002, pp. 12–16.
[18] C.J. Lin, C.T. Lin, An ART-based fuzzy adaptive learning control network, IEEE Trans. Fuzzy Systems 5 (4) (1997) 477–496.
[19] C.T. Linand, C.S.G. Lee, Neural-network-based fuzzy logic control and decision system, IEEE Trans. Comput. C-40 (12) (1991) 1320–1336.
[20] D. Montana, L. Davis, Training feedforward neural networks using genetic algorithms, Proc. of the Int. Joint Conf. on Artificial Intelligence, Detroit, 1989, pp. 762–767.
[21] D. Nauck, R. Kruse, A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation, Proc. IEEE Int. Conf. on Neural Networks, San Francisco, 1993, pp. 1022–1027.
[22] H. Nomura, I. Hayashi, N. Wakami, A self-tuning method of fuzzy reasoning by genetic algorithm, Proc. IEEE Int. Conf. on Fuzzy Systems, San Diego, 1992, pp. 236–245.
[23] K. Ogata, Discrete-Time Control Systems, Prentice-Hall, Englewood Cliffs, NJ, 1987.
[24] M. Sugeno, An introduction survey of fuzzy control, Inform. Sci. 36 (1985) 59–83.
[25] M. Sugeno, T. Yasukawa, A fuzzy-logic-based approach to qualitative modeling, IEEE Trans. Fuzzy Systems 1 (1) (1993) 7–31.
[26] T. Tagaki, I. Hayashi, NN-driven fuzzy reasoning, Internat. J. Approx. Reason. 5 (1991) 191–212.
[27] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Systems Man Cybernet. 15 (1) (1985) 116–132.
[28] J. Tanomaru, S. Omatu, Process control by on-line trained neural controllers, IEEE Trans. Ind. Electron. 39 (1992) 511–521.
[29] P. Thrift, Fuzzy logic synthesis with genetic algorithms, Proc. Int. Conf. Genetic Algorithms, San Diego, July 1991, pp. 509–513.
[30] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Systems Man Cybernet. 22 (6) (1992) 1414–1427.
[31] P.J. Werbos, Neural control and fuzzy logic: connections and designs, Internat. J. Approx. Reason. 6 (1992) 185–219.
[32] D. Whitley, T. Starkweather, C. Bogart, Genetic algorithm and neural networks: optimizing connections and connectivity, Parallel Comput. 14 (1990) 347–361.
[33] R.R. Yager, Implementing fuzzy logic controller using a neural network, Fuzzy Sets and Systems 48 (1992) 53–64.