Correspondence

Temperature Control with a Neural Fuzzy Inference Network

Chin-Teng Lin, Chia-Feng Juang, and Chung-Ping Li

***Abstract*— Although multilayered backpropagation neural networks (BPNN's) have demonstrated high potential in the nonconventional branch of adaptive control, their long training time usually discourages their applications in industry. Moreover, when they are trained on-line to adapt to plant variations, the over-tuned phenomenon usually occurs. To overcome the weakness of the BPNN, in this paper we propose a neural fuzzy inference network (NFIN) suitable for adaptive control of practical plant systems in general and for adaptive temperature control of a water bath system in particular. The NFIN is inherently a modified Takagi–Sugeno–Kang (TSK)-type fuzzy rule-based model possessing a neural network's learning ability. In contrast to the general adaptive neural fuzzy networks, where the rules should be decided in advance before parameter learning is performed, there are no rules initially in the NFIN. The rules in the NFIN are created and adapted as on-line learning proceeds via simultaneous structure and parameter identification. The NFIN has been applied to a practical water bath temperature-control system. As compared to the BPNN under the same training procedure, the simulated results show that not only can the NFIN greatly reduce the training time and avoid the over-tuned phenomenon, but the NFIN also has perfect regulation ability. The performance of the NFIN is also compared to that of the traditional PID controller and fuzzy logic controller (FLC) on the water bath temperature-control system. The three control schemes are compared through experimental studies with respect to set-points regulation, ramp-points tracking, and the influence of unknown impulse noise and large parameter variation in the temperature-control system. It is found that the proposed NFIN control scheme has the best control performance of the three control schemes.**

***Index Terms*—Kalman filter algorithm, similarity measure, space partitioning, TSK fuzzy rules, water bath temperature control.**

I. Introduction

Classical control theory usually requires a mathematical model for designing the controller [1]–[3]. The inaccuracy of mathematical modeling of the plants usually degrades the performance of the controller, especially for nonlinear and complex control problems. Recently, the advent of the fuzzy logic controllers (FLC's) and the neural controllers based on multilayered backpropagation neural networks (BPNN's) has inspired new resources for the possible realization of better and more efficient control [4], [21]. They offer a key advantage over traditional adaptive control systems. That is, they do not require mathematical models of the plants. The concept of fuzzy logic has been applied successfully to the control of industrial processes [5]–[7]. Conventionally, the selection of fuzzy if-then rules often relies on a substantial amount of heuristic observation to express proper strategy knowledge. Obviously, it is difficult for human experts to examine all of the input–output data from a complex system to find a number of proper rules for the FLC. For a BPNN, its nonlinear mapping and self-learning abilities have been the motivating factors for its use in developing intelligent control systems [8]. However, slow convergence is the major disadvantage of the BPNN. When the BPNN is trained off line, it must spend a long time to achieve satisfactory convergence. Moreover, when it is trained on line in order to adapt to the environment variations, its global tuning property usually leads to the over-tuned phenomenon, which will degrade the performance of the controller. Although several approaches [9], [10] have been proposed as efficient ways for solving these problems, the improvement is limited. In this paper, a neural fuzzy inference network (NFIN) is proposed to overcome the disadvantages of the BPNN and FLC.

The NFIN is a fuzzy rule-based network possessing a neural network's learning ability. Compared to other existing neural fuzzy networks [12], [13], [7], a major characteristic of the network is that no preassignment and design of the rules is required. The rules are constructed automatically during the on-line operation. The structure-learning phase and the parameter-learning phase [20], [22] are adopted on-line for the construction task.

One important task in the structure identification of the NFIN is the partition of the input space, which influences the number of fuzzy rules generated. An on-line input space-partition method and the aligned clustering-based method are proposed in this paper, which will reduce not only the number of rules generated but also the number of fuzzy sets in each dimension. Another feature of the NFIN is that it can optimally determine the consequent part of fuzzy if-then rules during the structure-learning phase. A fuzzy rule of the following form is adopted in our system initially

$$\text{Rule } j\text{: IF } x_1 \text{ is in } A_{i1} \text{ and } \ldots \text{ and } x_n \text{ is in } A_{in} \quad \text{THEN } y_i \text{ is } m_i \tag{1}$$

where x_i and y_i are the input and output variables, respectively, A_{ij} is a fuzzy set, and m_i is the position of a symmetric membership function of the output variable with its width neglected during the defuzzification process. Then, by monitoring the change of the network output error, additional terms (the linear terms used in the consequent part of the Takagi–Sugeno–Kang (TSK) model [16], [17]) will be included when necessary to further reduce the output error. This consequent identification process is employed in conjunction with the precondition-identification process to reduce both the number of rules and the number of consequent terms. For the parameter-identification scheme, the consequent parameters are tuned by the Kalman filter algorithm, and the precondition parameters are tuned by the backpropagation learning algorithm. Both the structure and parameter learning are done simultaneously to achieve fast learning.

Temperature control is an important factor in many process control systems [9], [10], [11]. 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 systems are often nonlinear and tend to change in an unpredictable way, they are not easy to control accurately. To verify that the NFIN has good control performance on the temperature-control system and is able to cope with the disadvantages of the BPNN, we compared it with the BPNN under the same training process via a simulation of a water bath temperature-control system.

Manuscript received June 3, 1998; revised December 20, 1998. This work was supported by the R.O.C. National Science Council under Grant NSC88-2213-E-009-114.

The authors are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C. (e-mail: ctlin@fnn.cn.nctu.edu.tw).

Publisher Item Identifier S 1094-6977(99)05329-8.

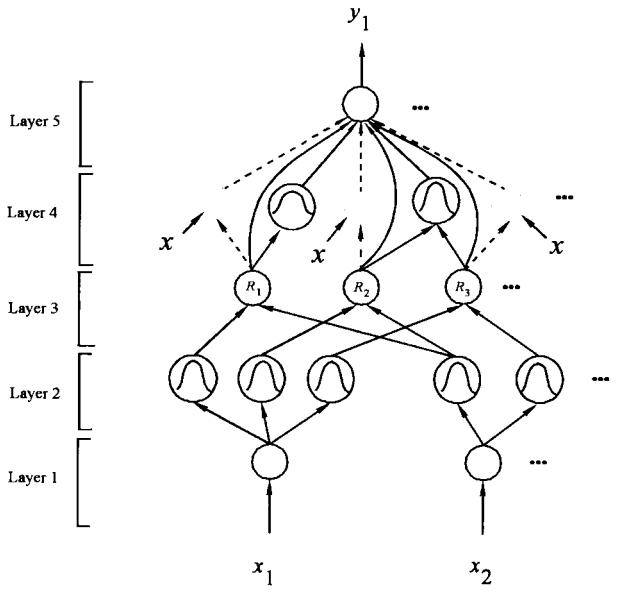


Fig. 1. Structure of the neural fuzzy inference network (NFIN).

To see if the NFIN, based on the inverse modeling scheme, can work well for a practical system, we compare it with the FLC and PID controller on a real water bath temperature-control system. The three schemes are compared through experimental studies with respect to set-points regulation, ramp-points tracking, the influence of unknown impulse noise, and large parameter variation in the system.

This paper is organized as follows. In Section II, the structure and the learning algorithm of the NFIN are proposed. In Section III, the configuration of the NFIN-based control and the training process are introduced. In Sections IV and V, simulation studies and experimental studies are presented, respectively. A conclusion is made in Section VI.

II. Neural Fuzzy Inference Network (NFIN)

In this section, the structure of the NFIN as shown in Fig. 1 is introduced. With this five-layered network structure of the NFIN, we shall define the function of each node of the NFIN in Section II-A and the learning algorithm of the NFIN in Section II-B.

A. Structure of the NFIN

Let $u^{(k)}$ and $a^{(k)}$ denote the input and output of a node in layer k, respectively. The functions of the nodes in each of the five layers of the NFIN are described as follows.

Layer 1: No computation is done in this layer. Each node in this layer which corresponds to only one input variable, transmits input values to the next layer directly. That is

$$a^{(1)} = u_i^{(1)} = x_i. \tag{2}$$

Layer 2: Each node in this layer corresponds to one linguistic label (small, large, etc.) of one of the input variables in Layer 1. In other words, the membership value which 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

$$a^{(2)} = e^{-\frac{(u_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij}^2}} \tag{3}$$

where m_{ij} and σ_{ij} are, respectively, the center (or mean) and the width (or variance) of the Gaussian membership function of the jth term of the ith input variable x_i. Unlike other clustering-based partitioning methods where each input variable has the same number of fuzzy sets, the number of fuzzy sets of each input variable is not necessarily identical in the NFIN.

Layer 3: A node in this layer represents one fuzzy logic rule and performs precondition matching of a rule. Here we use the following AND operation for each Layer 3 node

$$a^{(3)} = \prod_i u_i^{(3)} \tag{4}$$

where the product is over the Layer 2 nodes participating in the IF part of the rule.

Layer 4: This layer is called the consequent layer. Two types of nodes are used in this layer, and they are denoted as blank and shaded circles in Fig. 1, respectively, for functional distinction as described below. The node denoted by a blank circle (blank node) is the essential node representing a fuzzy set (described by a Gaussian membership function) of the output variable. Since the Gaussian membership function is of equal width, only the center of each Gaussian membership function is delivered to the next layer for the local mean of maximum (LMOM) defuzzification operation [18]. The function of the width is used only for output clustering as will be described in Section II-B (Process B). Different nodes in Layer 3 may be connected to the same blank node in Layer 4, meaning that the same consequent fuzzy set is specified for different rules. The function of the blank node is

$$a^{(4)} = \sum_j u_j^{(4)} \cdot a_{0i} \tag{5}$$

where $a_{0i} = m_{0i}$, the center of a Gaussian membership function. As for the shaded node, it is generated only when necessary. Each node in Layer 3 has its own corresponding shaded node in Layer 4. One of the inputs to a shaded node is the output delivered from Layer 3, and the other possible inputs (terms) are the input variables from Layer 1 as depicted in Fig. 1. The shaded node function is

$$a^{(4)} = \sum_j a_{ji} x_j \cdot u_i^{(4)} \tag{6}$$

where the summation is over all the inputs and a_{ji} is the corresponding parameter. Combining these two types of nodes in Layer 5, we obtain the whole function performed by this layer for each rule as

$$a^{(4)} = \left(\sum_j a_{ji} x_j + a_{0i} \right) u_i^{(4)}. \tag{7}$$

From the above equation, we find that with the creation of a shaded node, the consequent part performs the same function as that of the TSK-type fuzzy rule, where the consequent is a linear combination of input variables. That is why we connect the inputs to a shaded node in the NFIN. The motivation for the use of the shaded node, pointed out by Sugeno and Tanaka in [16], is that the TSK model can represent a complex system in terms of fewer rules than the ordinary Mamdani-type fuzzy model. However, the terms used in the consequent part of the TSK model are quite considerable for multiinput-multioutput systems or for the systems with high-dimensional input or output spaces. This problem is solved in our NFIN model. As proposed in Section II-B, the shaded node is created only when necessary for the rule with poor mapping. This way, the number of consequent terms is reduced, and the number of rules required is still small.

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

$$a^{(5)} = \sum_i a_i^{(4)} \Big/ \sum_i a_i^{(3)}. \tag{8}$$

B. Learning Algorithms for the NFIN

Two types of learning, structure, and parameter learning are used concurrently for constructing the NFIN. The structure learning includes both the precondition and consequent structure identification of a fuzzy if-then rule. There are no rules (i.e., no nodes in the network except the input/output nodes) in the NFIN initially. They are created dynamically as learning proceeds upon receiving on-line incoming training data by performing the following learning processes simultaneously:

A) Input/output space partitioning;
B) Construction of fuzzy rules;
C) Consequent structure identification;
D) Parameter identification.

In the above, processes A, B, and C belong to the structure-learning phase, and process D belongs to the parameter-learning phase. Process C is called the consequent part structure identification, because it determines the final consequent function of each rule. The details of these learning processes are described in the rest of this section.

Process A: Input/Output Space Partitioning: The way the input space is partitioned determines the number of rules extracted from training data as well as the number of fuzzy sets on the universal of discourse of each input variable. For each incoming pattern $\mathbf{x}$, the strength a rule is fired can be interpreted as the degree to which the incoming pattern belongs to the corresponding cluster. For computational efficiency, we can use the firing strength given in (4) directly as this degree measure

$$F^i(\mathbf{x}) = \prod_i u_i^{(3)} = e^{-[D_i(\mathbf{x}-\mathbf{m}_i)]^T[D_i(\mathbf{x}-\mathbf{m}_i)]} \tag{9}$$

where $F^i \in [0,1]$, $D_i = \mathrm{diag}(1/\sigma_{i1}, 1/\sigma_{i2}, \ldots, 1/\sigma_{\mathrm{in}})$, and $\mathbf{m}_i = (m_{i1}, m_{i2}, \ldots, m_{in})^T$. Using this measure, we can obtain the following criterion for the generation of a new fuzzy rule. Let $\mathbf{x}(t)$ be the newly incoming pattern. Find

$$J = \arg \max_{1 \le j \le c(t)} F^j(\mathbf{x}) \tag{10}$$

where $c(t)$ is the number of existing rules at time t. If $F^J \le \bar{F}(t)$, then a new rule is generated where $\bar{F}(t) \in (0,1)$ is a prespecified threshold that decays during the learning process. The threshold decides the number of input and output clusters generated in the NFIN. In general, a small variation of $\bar{F}(t)$ changes only the initial location of generated clusters, which does not influence the performance of NFIN owing to the succeeding parameter learning. Also, for the proposed NFIN, if some $\bar{F}(t)$ value can achieve good performance, the increment of this $\bar{F}(t)$ value will increase the number of generated clusters, but the performance of NFIN does not change much. As for the initialization of $\bar{F}(t)$, it is heuristic. The initial $\bar{F}(t)$ for input space partitioning (denoted by $\bar{F}_{\mathrm{in}}$) is set to a larger value for a complex modeling problem than for a simple one. Usually, the initial value of $\bar{F}_{\mathrm{in}}$ is set in the range of 0 and $1/e$. The initial $\bar{F}(t)$ for output space partitioning (denoted by $\bar{F}_{\mathrm{out}}$) is set to a value a little bit smaller than 1. If $\bar{F}_{\mathrm{out}}$ is set to one, then the number of input clusters is the same as that of the output ones. For the decaying of $\bar{F}(t)$, the decay can reduce the rule-generation opportunity gradually in the succeeding learning process and thus avoid unlimited generation of rules. Since most rules are generated in the initial learning stage, the performance of NFIN is insensitive to the decay rate of $\bar{F}(t)$.

Once a new rule is generated, the initial centers and widths are set as

$$\mathbf{m}_{(c(t)+1)} = \mathbf{x} \tag{11}$$

$$D_{(c(t)+1)} = -\frac{1}{\beta} \cdot \mathrm{diag}\left(1/\ln\left(F^J\right), \ldots, 1/\ln\left(F^J\right)\right) \tag{12}$$

according to the first-nearest-neighbor heuristic [19], where $\beta \ge 0$ decides the overlap degree between two clusters.

After a rule is generated, the next step is to decompose the multidimensional membership function formed in (11) and (12) to the corresponding one-dimensional membership function for each input variable. For the Gaussian membership function used in the NFIN, the task can be done easily as

$$e^{-[D_i(\mathbf{x}-\mathbf{m}_i)]^T[D_i(\mathbf{x}-\mathbf{m}_i)]} = \prod_j e^{-\frac{(x_j-m_{ij})^2}{\sigma_{ij}^2}} \tag{13}$$

where m_{ij} and σ_{ij} are, respectively, the projected center and width of the membership function in each input dimension. To reduce the number of fuzzy sets of each input variable and to avoid the existence of redundant ones, we should check the similarities between the newly projected membership function and the existing ones in each input dimension. Since bell-shaped membership functions are used in the NFIN, we use the formula of the similarity measure $E(A,B)$ of two fuzzy sets A and B derived previously (see [20] and [21] for details), where $0 \le E(A,B) \le 1$, and the larger $E(A,B)$ is, the more A is similar to B. Let $\mu(m_i, \sigma_i)$ represent the Gaussian membership function with center m_i and width σ_i. Suppose no rules are existent initially. The whole algorithm for the generation of new fuzzy rules, as well as fuzzy sets in each input dimension, is as follows (see the first algorithm at the bottom of the next page). $\alpha(t)$ is a scalar similarity criterion which is monotonically decreasing such that higher similarity between two fuzzy sets is allowed in the initial stage of learning. For the output space partitioning, the same measure in (10) is used. Since the criterion for the generation of a new output cluster is related to the construction of a rule, we shall describe it together with the rule-construction process in Process B.

Process B: Construction of Fuzzy Rules: As mentioned in learning process A, the generation of a new input cluster corresponds to the generation of a new fuzzy rule, with its precondition part constructed by the learning algorithm in process A. At the same time, we have to decide the consequent part of the generated rule. If a new input cluster is formed after the presentation of the current input–output training pair $(\mathbf{x},\mathbf{d})$, then the consequent part is constructed by the second algorithm shown at the bottom of the next page. The algorithm is based on the fact that different preconditions of different rules may be mapped to the same consequent fuzzy set.

Process C: Consequent Structure Identification: Until now, the NFIN has contained fuzzy rules in the form of (1). Even though such a basic NFIN can be used directly for system modeling, a large number of rules is necessary for modeling sophisticated systems under a tolerable modeling accuracy. To cope with this problem, we adopt the spirit of the TSK model [17] in the NFIN. In the TSK model, each consequent part is represented by a linear equation of the input variables. It is reported in [16] that the TSK model can model a sophisticated system using a few rules. Even so, if the number of input and output variables is large, the consequent parts used in the output are quite considerable, and some may be superfluous. Instead of using the linear equation of all the input variables (terms) in each rule, we add only some additional terms to some rules when necessary, to cope with the dilemma between the number of rules and the number of consequent terms. Monitor the error curve. If the error does not diminish over a period of time $E(t+T) - E(t-T) = \sum_{k=t}^{T}(y(k) - y^d(k))^2 - \sum_{k=t-2T}^{t-T}(y(k) - y^d(k))^2 < 0.5$ and the error is still too large, we will add the additional terms to the rule with the largest error value evaluated during this period of time. The error of rule i during

each period of time is calculated by

$$\mathrm{RE}(i) = \sum_t \frac{a_i^{(3)}}{\sum_{k=1}^{c} a_k^{(3)}} \Big(y(t) - y^d(t)\Big)^2$$

where $a_i^{(3)}$ is the firing strength of rule i, c is the number of rules, $y^d(t)$ is the desired output, $y(t)$ is the current output, and $RE(i)$ is the accumulated error of rule i. The process may be done repeatedly over a period of time until a satisfactory result occurs.

Process D: Parameter Identifi-cation: The parameter-identification process is done concurrently with the structure-identification process. The idea of backpropagation is used for this supervised learning. Considering the single-output case for clarity, our goal is to minimize the error function

$$E = \frac{1}{2}\Big(y(t) - y^d(t)\Big)^2 \tag{14}$$

where $y^d(t)$ is the desired output, and $y(t)$ is the current output. The parameters a_{ji} in Layer 4 are tuned by Kalman filter algorithm [3] as

$$\mathbf{a}(t+1) = \mathbf{a}(t) + P(t+1)\mathbf{u}(t+1)(y^d(t) - y(t)) \tag{15}$$

$$P(t+1) = \frac{1}{\lambda}\left[P(t) - \frac{P(t)\mathbf{u}^T(t+1)\mathbf{u}(t+1)P(t)}{\lambda + \mathbf{u}^T(t+1)P(t)\mathbf{u}(t+1)}\right] \tag{16}$$

where $0 < \lambda \leq 1$ is the forgetting factor, $\mathbf{u}$ is the current input vector, $\mathbf{a}$ is the corresponding parameter vector, and P is the covariance matrix. The initial parameter vector $\mathbf{a}(0)$ is determined in the structure-learning phase and $P(0) = \sigma I$, where σ is a large positive constant. As for the free parameters m_{ij} and σ_{ij} of the input membership functions in Layer 2, they are updated by the backpropagation algorithm. Using the chain rule, we have

$$\begin{aligned} m_{ij}^{(2)}(t+1) &= m_{ij}^{(2)}(t) - \eta \frac{\partial E}{\partial m_{ij}^{(2)}} \\ &= m_{ij}^{(2)}(t) - \eta \frac{\partial E}{\partial y} \sum_k \frac{\partial y}{\partial a_k^{(3)}} \frac{\partial a_k^{(3)}}{\partial m_{ij}^{(2)}} \end{aligned} \tag{17}$$

where

$$\frac{\partial E}{\partial y} = y(t) - y^d(t) \qquad \frac{\partial y}{\partial a_k^{(3)}} = \frac{a_k^{(4)} - y}{\left(\sum_i a_i^{(3)}\right)} \tag{18}$$

$$\frac{\partial a_k^{(3)}}{\partial m_{ij}^{(2)}} = \begin{cases} a_k^{(3)} \frac{2(x_i - m_{ij})}{\sigma_{ij}^2} & \text{if term node } j \text{ is connected} \\ & \quad \text{to rule node } k \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

IF $\mathbf{x}$ is the first incoming pattern THEN do
PART 1. { Generate a new rule,
with center $\mathbf{m}_1 = \mathbf{x}$, width $D_1 = diag(1/\sigma_{init}, \cdots, 1/\sigma_{init})$,
where σ_{init} is a prespecified constant.
After decomposition, we have n one-dimensional membership functions,
with $m_{1i} = x_i$ and $\sigma_{1i} = \sigma_{init}, i = 1 \cdots n$.
}
ELSE for each newly incoming $\mathbf{x}$, do

PART 2. { find $J = \arg\max_{1 \leq j \leq c(t)} F^j(\mathbf{x})$,
IF $F^J \geq \bar{F}_{in}(t)$
do nothing
ELSE
{ $c(t+1) = c(t) + 1$,
generate a new fuzzy rule, with
$\mathbf{m}_{c(t+1)} = \mathbf{x}$, $D_{c(t+1)} = -\frac{1}{\beta} \cdot diag(1/\ln(F^J), \cdots, 1/\ln(F^J))$.
After decomposition, we have
$m_{new-i} = x_i$, $\sigma_{new-i} = -\beta \cdot \ln(F^J), i = 1 \cdots n$.
Do the following fuzzy measure for each input variable i :
$\{degree(i,t) \equiv \max_{1 \leq j \leq k_i} E[\mu(m_{new-i}, \sigma_{new-i}), \mu(m_{ji}, \sigma_{ji})]$,
where k_i is the number of partitions of the ith input variable.
IF $degree(i,t) \leq \alpha(t)$,
THEN adopt this new membership function, and set $k_i = k_i + 1$,
ELSE set the projected membership function as the closest one. }
}
}.

IF there are no output clusters,
do { *PART 1* in Process A, with $\mathbf{x}$ replaced by $\mathbf{d}$ }
ELSE
do {
find $J = \arg\max_j F^j(\mathbf{x})$.
IF $F^J \geq \bar{F}_{out}(t)$
connect input cluster $c(t+1)$ to the existing output cluster J,
ELSE
generate a new output cluster,
do the decomposition process in *PART 2* of Process A,
connect input cluster $c(t+1)$ to the newly generated output cluster.
}.

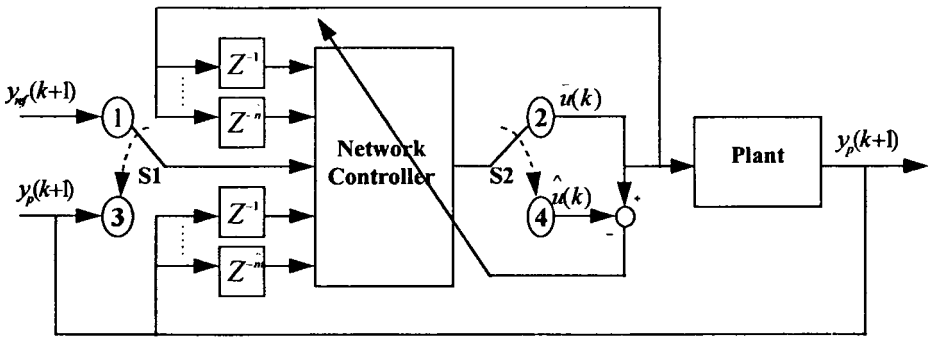


Fig. 2. Conventional on-line training scheme.

Similarly, we have

$$\sigma_{ij}^{(2)}(t+1) = \sigma_{ij}^{(2)}(t) - \eta \frac{\partial E}{\partial \sigma_{ij}^{(2)}} = \sigma_{ij}^{(2)}(t) - \eta \frac{\partial E}{\partial y} \sum_k \frac{\partial y}{\partial a_k^{(3)}} \frac{\partial a_k^{(3)}}{\partial \sigma_{ij}^{(2)}} \tag{20}$$

where

$$\frac{\partial a_k^{(3)}}{\partial \sigma_{ij}^{(2)}} = \begin{cases} a_k^{(2)} \frac{2(x_i - m_{ij})^2}{\sigma_{ij}^3} & \text{if term node } j \text{ is connected to node } k \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

III. NFIN-Based Adaptive Control

The inverse control configuration shown in Fig. 2 is adopted. Two training phases, off-line and on-line training, are used for the design of the controller. For the off-line training, the general inverse-modeling learning scheme [15] is used. A sequence of random input signals $u_{\rm rd}(k)$ under the magnitude limits of the plant input is injected directly to the plant, and then an open-loop input–output characteristic of the plant is obtained. According to the input–output characteristic of the plant, proper training patterns are selected to cover the entire reference output space. Another task for inverse modeling is to decide the proper input variables of the network controller when there is no *a priori* knowledge of the plant, i.e., $\hat{m}$ and $\hat{n}$ in Fig. 2 have to be decided. In Section V-B, we shall demonstrate an experimental method [11] of doing this. Using the collected training patterns with the values of the selected input variables as the input pattern and the corresponding control signal $u_{\rm rd}(k)$ as the target pattern, the network can be supervised and updated to minimize an error function E defined by $E = \sum_{k=1}^{k_n} \frac{1}{2}[u_{\rm rd}(k) - \hat{u}(k)]^2$ where k_n is the number of training patterns.

For the on-line training, a conventional on-line training scheme is used. Fig. 2 is a block diagram for the conventional on-line training scheme. In executing this scheme, we follow two phases: the control phase and the training phase. In the control phase, the switches S1 and S2 are connected to nodes 1 and 2, respectively, to form a control loop. In this loop, the control signal $\bar{u}(k)$ is generated according to the input vector $I'(k) = [y_{\rm ref}(k+1), y_p(k), \ldots, y_p(k-\hat{m}+1), u(k-1), \ldots, u(k-\hat{n})]^T$ where u denotes the input, y_p is the output, and $y_{\rm ref}$ is the reference output. In the training phase, the switch S1 and S2 are connected to nodes 3 and 4, respectively, to form a training loop. In this loop, we can define a training pattern with input vector $I(k) = [y_p(k+1), y_p(k), \ldots, y_p(k-\hat{m}+1), u(k-1), \ldots, u(k-\hat{n})]^T$ and desired output $\bar{u}(k)$, where the input vector of the network controller is the same as that used in the off-line training scheme (see Section IV-B). With this training pattern, the network controller can be supervised and trained at each time step k to minimize the error function $E(k+1)$ defined by $E(k+1) = \frac{1}{2}[\bar{u}(k) - \hat{u}(k)]^2$ where $\hat{u}(k)$ is the actual output of the network controller when it receives the input vector $I(k)$ in the training phase.

IV. Simulation Studies

A. Problem Statement

To see whether the proposed NFIN can achieve good performance and overcome the disadvantages of the BPNN, we compare it with the BPNN under the same aforementioned training procedure on a simulated water bath temperature-control system. Consider a discrete-time SISO temperature-control system

$$y_p(k+1) = A(T_s)y_p(k) + \frac{B(T_s)}{1 + e^{\frac{1}{2}y_p(k) - \gamma}} u(k) + [1 - A(T_s)]y_o \tag{22}$$

where $A(T_s) = e^{-aT_s}$ and $B(T_s) = (b/a)(1 - e^{-aT_s})$. The above equation models a real water bath temperature-control system given in [10]. The parameters in this simulation are $a = 1.00151e^{-4}$, $b = 8.67973e^{-3}$, $\gamma = 40.0$, and $y_o = 25.0$°C. The plant input $u(k)$ is limited to 0 and 5 V, and the sampling period is $T_s = 30$ s. The task is to control the simulated system to follow three set-points.

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

B. Simulation Results

By implementing the off-line training scheme, a sequence of random input signals $u_{\rm rd}(k)$ limited to 0 and 5 V is injected directly into the simulated system described in (22). The initial temperature of the water is 25 °C, and the temperature rises progressively when random input signals are injected. The water temperature finally reaches about 100 °C when 90 input signals are injected. The temperature remains at about this highest value when more input signals are injected. From this input–output characteristic of the simulated system, 90 training patterns are selected to cover the entire reference output space. In this way, another set of about 90 training patterns can be collected. Observing this input/output characteristic, we find that the input/output mapping of the two sets is about the same. So only one set, which contains 90 training patterns, is needed for the off-line training of NFIN. From (22), the input vector of the network controller clearly can be decided as $\hat{m} = 1$ and $\hat{n} = 0$, i.e., $I(k) = [y_p(k+1), y_p(k)]^T$.

For the BPNN, a four-layer feedforward network with two hidden layers is used. To find a suitable number of hidden nodes, three networks $\mathrm{BPNN}(5,5)$, $\mathrm{BPNN}(10,10)$, and $\mathrm{BPNN}(15,15)$ are chosen, where the notation $\mathrm{BPNN}(a,b)$ denotes that the number of nodes in

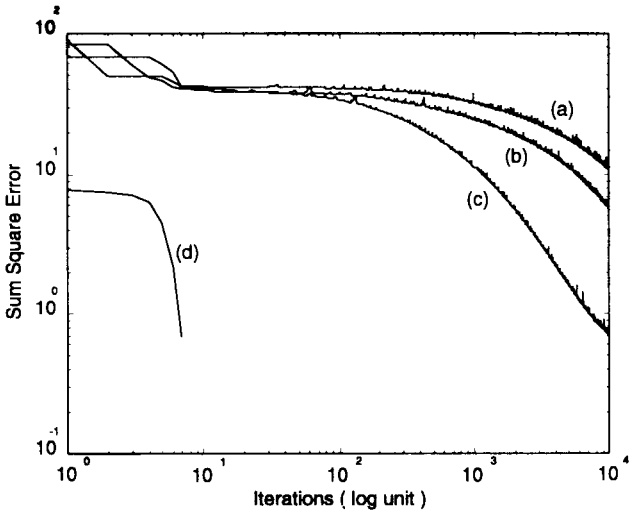


Fig. 3. Convergence curves of the simulated system through off-line training using (a) $\mathrm{BPNN}(5,5)$, (b) $\mathrm{BPNN}(10,10)$, (c) $\mathrm{BPNN}(15,15)$, and (d) NFIN.

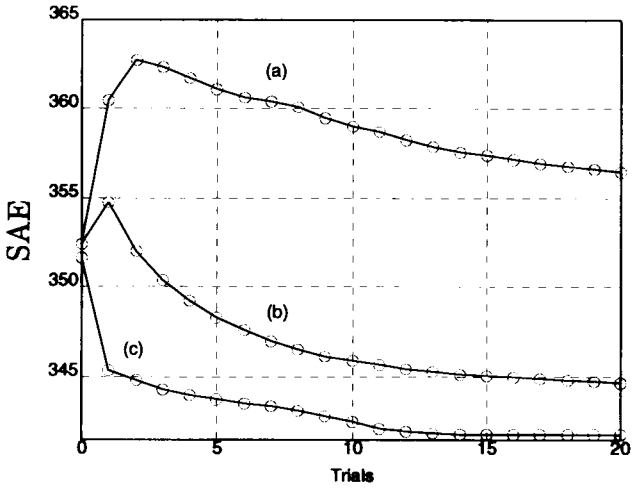


Fig. 4. Regulation performance of each trial of the simulated system through on-line training using (a) $\mathrm{BPNN}(15,15)$ based on the conventional on-line training scheme, (b) $\mathrm{BPNN}(15,15)$ based on the new on-line training scheme, and (c) NFIN based on the conventional on-line training scheme.

the first and second hidden layer are a and b, respectively. To increase the convergence speed, a modified form of the generalized delta rule [14] is used.

For the NFIN, the learning parameters are set to $\sigma_{\mathrm{init}} = 0.1$, $\eta = 0.005$, $\beta = 0.7$, $\bar{F}_{\mathrm{in}} = 0.1$, $\bar{F}_{\mathrm{out}} = 0.7$, and $\alpha = 0.5$. Values σ_{init}, α, and $\bar{F}_{\mathrm{out}}$ are usually kept the same for different modeling problems, and the value η influences mainly the learning speed. In general, the parameter that needs to be reassigned for different modeling problems is $\bar{F}_{\mathrm{in}}$. In the off-line training, the convergence curves of the NFIN and the three BPNN's are shown in Fig. 3. The curves show the sum square error per iteration as a function of the number of iterations. For the three BPNN's, we find that the $\mathrm{BPNN}(15,15)$ shows the highest convergence speed of all. As expected, the convergence speed of the NFIN is much higher than that of the $\mathrm{BPNN}(15,15)$. It is noted that the error of the NFIN [curve (d)] before the first iteration is unknown, since it is an empty (no rule) network before learning.

In the on-line training, the NFIN and $\mathrm{BPNN}(15,15)$ controllers are trained by the conventional on-line training scheme shown in Fig. 2. Moreover, the new on-line training method that performs multiple updating operations during each sampling period [10] is also applied to the $\mathrm{BPNN}(15,15)$ for comparison. In the new on-line training method, we choose 20 additional adjacent training patterns per sampling period. To test their regulation performance in each trial, a performance index, sum of absolute error (SAE), is defined by

$$\mathrm{SAE} = \sum_{k} |y_{\mathrm{ref}}(k) - y_p(k)| \tag{23}$$

where $y_{\mathrm{ref}}(k)$ and $y_p(k)$ are the reference output and the actual output of the simulated system, respectively. The performance index SAE is calculated for k ranging from 1–120, which is called a trial.

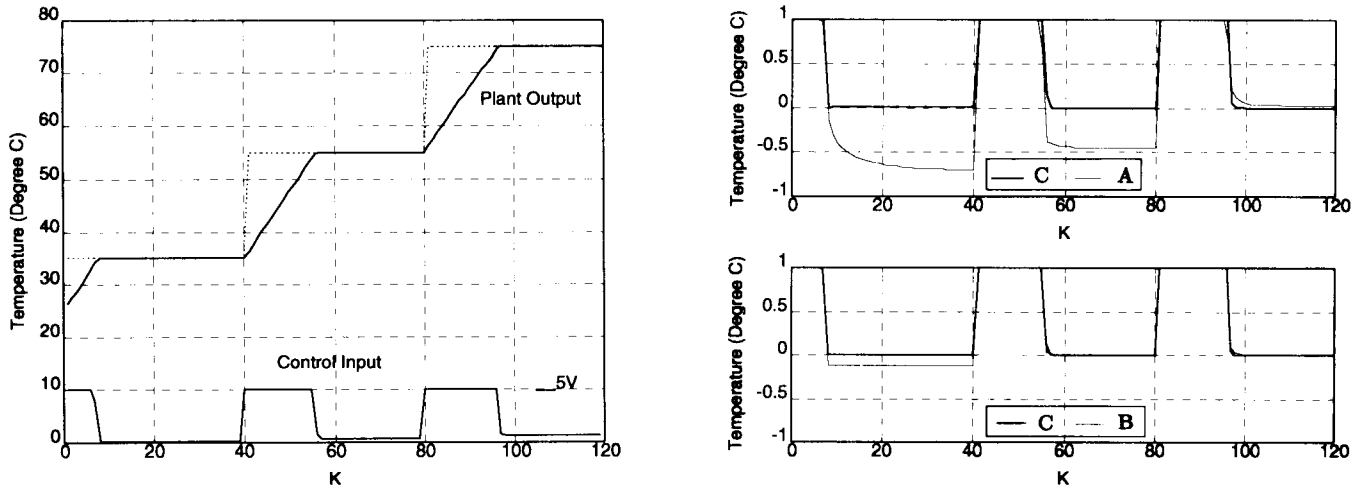
After 20 trials of on-line training, the regulation performance of each trial of the aforementioned three cases is shown in Fig. 4, and the final regulation performance of the NFIN and the regulation errors of the three cases in the twentieth trial of the on-line training are shown in Fig. 5. The curves in Fig. 4 show the sum of absolute error per trial as a function of the number of trials. As expected, the global learning ability for the $\mathrm{BPNN}(15,15)$, based on the conventional on-line training scheme, leads to seriously over-tuned phenomenon. From the corresponding errors in Fig. 5(b), we find that it only performs good regulation at the upper set-point but deteriorates at others. Furthermore, we find its performance index value after 20 trials of on-line training is larger than its initial. The over-tuned phenomenon still exists for the $\mathrm{BPNN}(15,15)$ based on the new on-line training scheme, although such a method increases the on-line convergence speed and improves control performance as expected. From the corresponding errors in Fig. 5(b), we also observe that it only improves the performance at the upper and middle set-points, not at the lower set-points. Furthermore, the additional training of the adjacent patterns increases the computational load per sampling period. The NFIN based on the conventional on-line training scheme shows the highest on-line convergence speed and, owing to the local tuning property of fuzzy rule-based systems, there is little over-tuned phenomenon after 20 trials of on-line training. According to the corresponding errors in Fig. 5(b), the NFIN shows good regulation ability at all set-points, and few errors exist. According to the performance index, among the three cases the NFIN shows the best regulation-control performance for the overall process. Fig. 6(a) shows the final assignment of fuzzy rules of the NFIN after 20 trials of on-line training in the $[y(k), y(k+1)]$ plain. The number of generated rules is seven, and the numbers of fuzzy sets on the $y(k)$ and $y(k+1)$ dimensions are four and four, respectively, as shown in Fig. 6(b). In total, the number of network-structure parameters is 29, but that of the $\mathrm{BPNN}(15,15)$ is 270.

For the simulation studies, a summary of comparisons between the two networks, NFIN and $\mathrm{BPNN}(15,15)$, is shown in Table I. The NFIN has fewer structure parameters and higher off-line convergence speed than the BPNN. Also, the NFIN based on the conventional on-line training scheme has a higher on-line convergence speed than the $\mathrm{BPNN}(15,15)$ based on the conventional or the new on-line training scheme proposed in [10]. After training, the regulation performance of the NFIN outperforms the $\mathrm{BPNN}(15,15)$ controller.

V. Experimental Studies

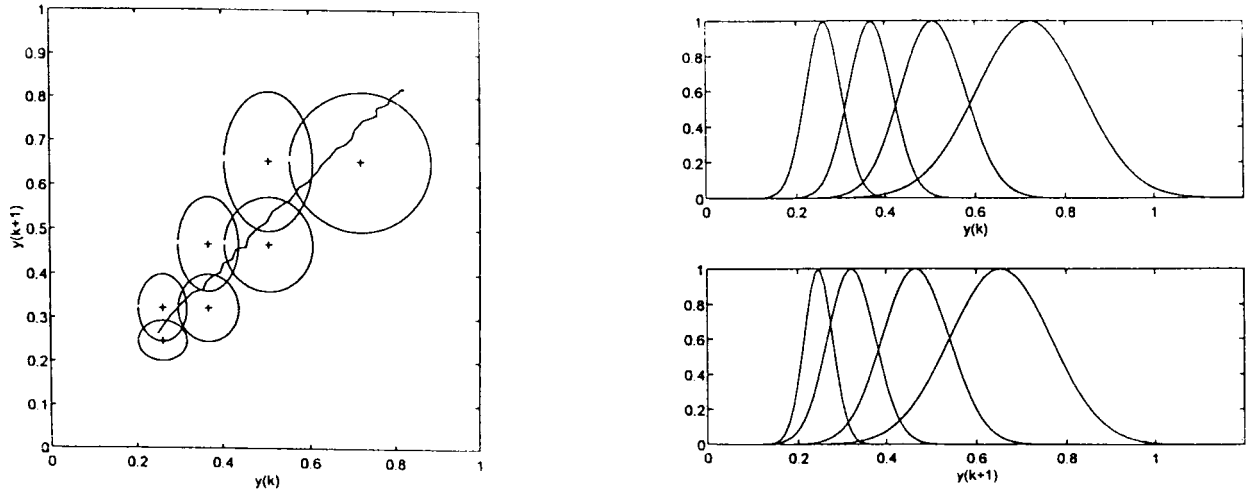
A. Water Bath Temperature-Control System

The experiment is performed on a real water bath temperature-control system. It imitates the water bath temperature-control system (BT-5 model) from a Yamato Science Inc. laboratory [11]. The water bath is an example of an important component in a batch-



(a) (b)

Fig. 5. Final regulation performance of the simulated system in the twentieth trial of on-line training using (a) NFIN based on the conventional on-line training scheme. (b) The corresponding errors: A: $\mathrm{BPNN}(15,15)$ based on the conventional on-line training scheme, B: $\mathrm{BPNN}(15,15)$ based on the new on-line training scheme, and C: NFIN based on the conventional on-line training scheme.



(a) (b)

Fig. 6. Final structure of the NFIN after 20 trials of on-line training. (a) The final assignment of fuzzy rules in the $[y(k), y(k+1)]$ plain. (b) The corresponding membership functions on the $y(k)$ and $y(k+1)$ dimensions.

TABLE I
SUMMARY OF COMPARISONS BETWEEN THE TWO NETWORK CONTROLLERS ON THE SIMULATED WATER BATH SYSTEM

Change in Error $\Delta e(k)$ \ 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

reactor process. A schematic diagram of the experimental setup is shown in Fig. 7. The system consists of five main components: a 6-liter stirrer tank, an industrial personal computer, a temperature control board, a k-type thermocouple sensor module, and a heater based on a solid state relay (SSR). A brief description of the five components follows. The capacity of the water bath is 8 liters, with dimensions $280 * 200 * 150$ (mm^3). It is heated by a 1000 W heater. To ensure even temperature distribution, a stirrer which can rotate at 180 r/min is used. An industrial personal computer with Intel 486-DX2-66 CPU is used in this experiment. To measure the water bath temperature and send out the control signals, a temperature-control board (ETC) is used. The temperature-control board is developed by the Industrial Technology Research Institute, Taiwan. It provides the ability to measure temperature and control signal output. For the temperature measurement, a k-type thermocouple (WSS.02) sensor is connected to the temperature-control board in order to measure the water bath temperature. The output signals of the temperature-control board are pulse width modulated (PWM) and are connected to the SSR to switch the heater on or off. The main control program is written in Microsoft-C, which decides on the control signals sent

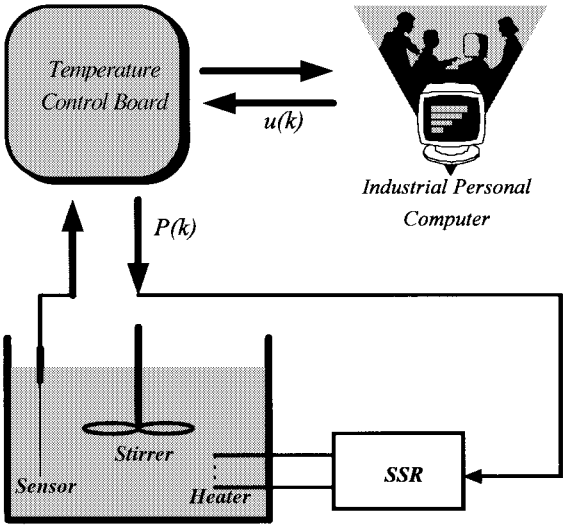


Fig. 7. Schematic diagram of the water bath temperature-control system.

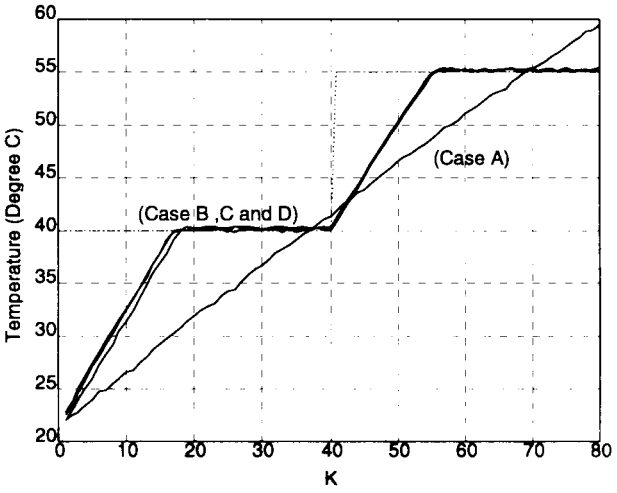


Fig. 8. Performance of the NFIN using four different input vectors after off-line training.

to the water bath system through the temperature-control board. The control action is according to the following constraints:

IF $u(k) \leq 0$, THEN $P(k) = 0.0\%$ heater off
IF $u(k) \geq 100.0$, THEN $P(k) = 100.0\%$ heater on
IF $0.0 < u(k) < 100.0$, THEN $P(k) = u(k)\%$, heater on

where k is the sampling number $(k = 1, 2, 3, \ldots)$, $u(k)$ is the output of the controller, and $P(k)$ is the output of the temperature-control board.

For a discrete-time control system, it is necessary to decide on an appropriate sampling period. In many practical applications, the sampling frequency cannot or should not exceed some limit. Therefore, for the water bath system, by considering its transient response behavior and the resolution of sensor, we chose $T_s = 30$ seconds as the sampling period.

B. Determination of the Input Vector of the NFIN Controller

As we have no *a priori* knowledge of the plant (especially the plant's order), an experimental method is used to determine the proper input variables of the network controller. First, with this method some different sets of input variables (i.e., input vectors) are selected in advance. These input vectors are all tried by the NFIN, and the one with the smallest input dimension and satisfactory control result is adopted. More clearly, each NFIN with a different input vector is trained by the off-line training scheme. To perform off-line training, we need to collect training data in advance. The input/output training data are collected the same way as in Section IV. A sequence of random input signals $u(k)$ in 0 and 100 is injected into the water bath system with an initial temperature near 25 °C. The temperature rises when the input signals are injected into the system continuously and reaches the boiling point of 100 °C after 110 input signals are injected. The temperature remains near 100 °C when succeeding random input signals are injected. From this input/output characteristic, the first 110 patterns covering the entire output range are selected as training patterns for off-line training. After off-line training, each NFIN is configured as a controller to the plant, and then the plants are controlled to follow some reference outputs. From the experimental results, we can find a proper input vector for the NFIN controller.

Based on the above idea, four input vectors $I(k)$ defined in Section III are selected for the water bath system as follows:

Case A: $I(k) = [y_p(k+1)]^T$;
Case B: $I(k) = [y_p(k+1), y_p(k)]^T$;
Case C: $I(k) = [y_p(k+1), y_p(k), y_p(k-1)]^T$;
Case D: $I(k) = [y_p(k+1), y_p(k), u(k-1)]^T$.

In the first case (Case A), only the plant output is used as the input vector. In the second and third cases (Cases B and C), in addition to the plant output, the delayed plant outputs are used in the input vector. For Case D, a delayed plant input is included. As described in Section III, for all of the above four cases, the desired output is $u(k)$, the current input. The input/output training data of the four cases are collected from the 110 selected training patterns. If the above four cases cannot meet the control requirement, then another type of input vector should be tried. An experiment to control the plant output to follow two set-points

$$y_{\mathrm{ref}}(k) = \begin{cases} 40\ ^\circ\mathrm{C}, & \text{for } k \leq 40 \\ 55\ ^\circ\mathrm{C}, & \text{for } 40 < k \leq 80 \end{cases}$$

is realized.

After the off-line training, the performance of the four NFIN's are shown in Fig. 8. Case A shows the largest regulation error and the poorest convergence status during training. Therefore, we know that only one input variable for the NFIN controller cannot make it work well. For Cases B, C and D, each shows good control performance and relatively fast convergence during training. This means that all of Cases B, C and D catch the basis order of the water bath system. Hence, for simplicity, we only need to choose Case B as the input vector of the NFIN controller for the water bath system. In other words, the NFIN controller based on input vector $[y_p(k+1), y_p(k)]^T$ has a good control capability for the water bath temperature-control system.

C. FLC and PID Control

One way to test whether the NFIN control works well is to compare it directly with other types of existing controllers. In this paper, we compare the NFIN controller to two traditional controllers: the FLC and the PID controller. Each of these three controllers is applied to the water bath temperature-control system. The formed three control systems are tested experimentally. The comparison indexes include

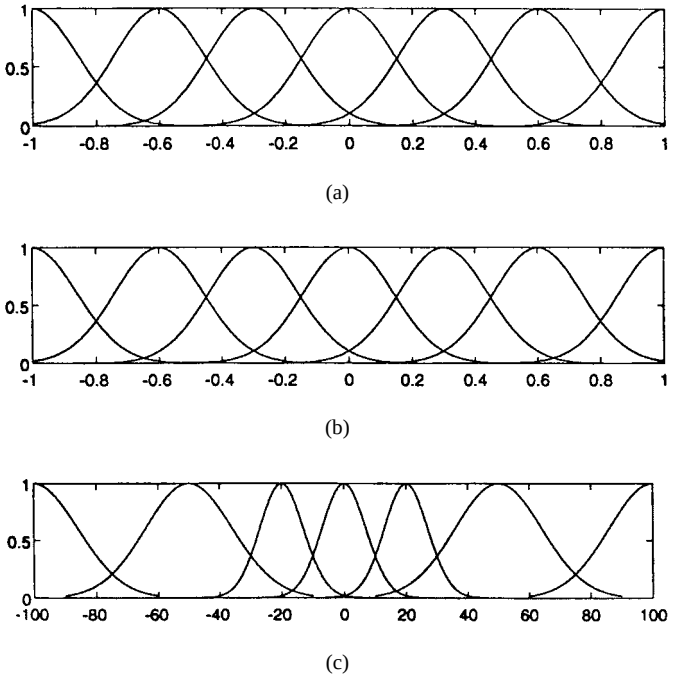


Fig. 9. Membership functions of the three fuzzy variables of the water bath system. (a) Input variable $e(k)$, (b) input variable $\Delta e(k)$, and (c) output variable $u(k)$.

set-points regulation, ramp-points tracking, the influence of impulse noise, and a large parameter variation in the system.

For the FLC, we specify the input variables as the performance error $e(k)$, which is the error between the reference output and actual temperature of the water bath system, and the rate of change of the performance error $\Delta e(k)$. The output variable $u(k)$ is the PWM signal between 0% and 100%. We quantified the three variables into seven fuzzy subsets from negative large (NL) to positive large (PL). The Gaussian membership function is chosen for the fuzzy subsets of the three fuzzy variables as shown in Fig. 9. According to common sense and engineering judgment, 25 fuzzy rules are specified in matrix form as shown in Table II.

For the PID control, a velocity-form discrete PID controller [3] is used and is described by

$$\begin{aligned}\Delta u(k) &= K\Big\{e(k)-e(k-1)+\frac{T_s}{2T_i}[e(k)+e(k-1)] \\ &\qquad +\frac{T_d}{T_s}[e(k)-2e(k-1)+e(k-2)]\Big\} \\ &= K_P[e(k)-e(k-1)]+K_I e(k) \\ &\quad +K_D[e(k)-2e(k-1)+e(k-2)] \end{aligned} \tag{24}$$

where

$$K_P = K-\frac{1}{2}K_I, \qquad K_I=\frac{KT_s}{T_i}, \qquad K_D=\frac{KT_D}{T_s}. \tag{25}$$

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 90 and 60, respectively. For the above designed FLC and PID controller, we have tried our best to achieve their respective best performance through several trial-and-error experiments.

TABLE II
FUZZY RULES FOR THE WATER BATH TEMPERATURE CONTROL SYSTEM

Criteria	NFIN	BPNN(15,15)	BPNN(15,15) (Using New On-line Training Scheme, [10])
Network Structure Parameters	29	270	270
Convergence Speed (Off-line)	8 iterations	10000 iterations	10000 iterations
Convergence Speed (On-line)	13 trials	30 trials	30 trials
The Final Regulation Performance SAE=$\sum_{k=1}^{120}\lvert y_{ref}(k)-y_p(k)\rvert$	SAE=341.7778	SAE=356.7284	SAE=344.1059
Computational Load (On-line)	29 parameters are tuned	270 parameters are tuned	21 times of conventional method

D. Experimental Results

For the aforementioned controllers (NFIN, FLC, and PID), four groups of experiments are conducted on the water bath temperature control system. Each experiment is performed over 120 sampling time steps, which result in a 60 min duration.

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

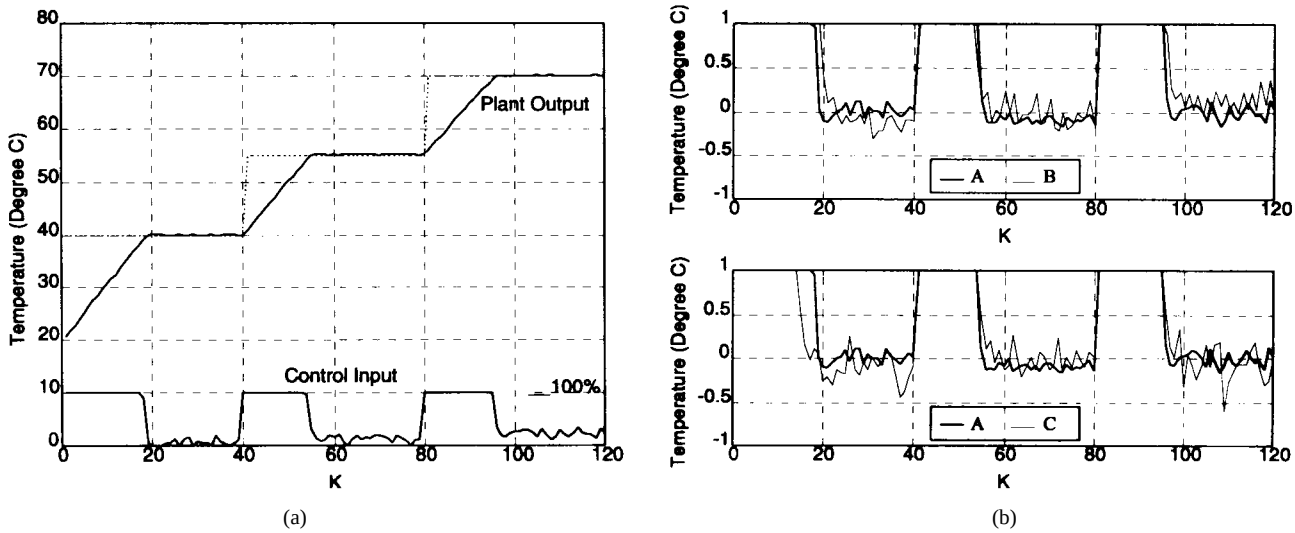
$$y_{\rm ref}(k)=\begin{cases}40\ ^\circ{\rm C}, & \text{for } k\le 40,\\ 55\ ^\circ{\rm C}, & \text{for } 40<k\le 80,\\ 70\ ^\circ{\rm C}, & \text{for } 80<k\le 120.\end{cases}$$

As mentioned in Section V-B, in obtaining the NFIN controller, 110 training patterns are chosen from the input–output characteristic in order to cover the entire reference output space. According to the selected training patterns based on the determined input vector (Case B in Section V-B), the NFIN is trained off-line. The learning parameters are selected as $\sigma_{\rm init}=0.1$, $\alpha=0.5$, $\eta=0.005$, $\beta=0.7$, $\bar{F}_{\rm in}=0.1$, and $\bar{F}_{\rm out}=0.7$. After off-line training, where eight rules are generated, the NFIN is configured as a direct controller to the water bath system, and then the conventional on-line training scheme is performed on the trained NFIN controller. The same learning parameters are used as those used in the off-line training scheme. The regulation performance after one trial of on-line training is compared to that of the other two controllers.

The regulation performance of the NFIN controller is shown in Fig. 10(a), and the regulation errors of the three controllers are shown in Fig. 10(b). The PID controller only performs good regulation at one operating point (at the middle set-point) but deteriorates at others. The NFIN seems to operate much faster in achieving the set-points. It has a shorter rise-time than the other controllers. For the steady-state error of set-points, the NFIN also shows the smallest error.

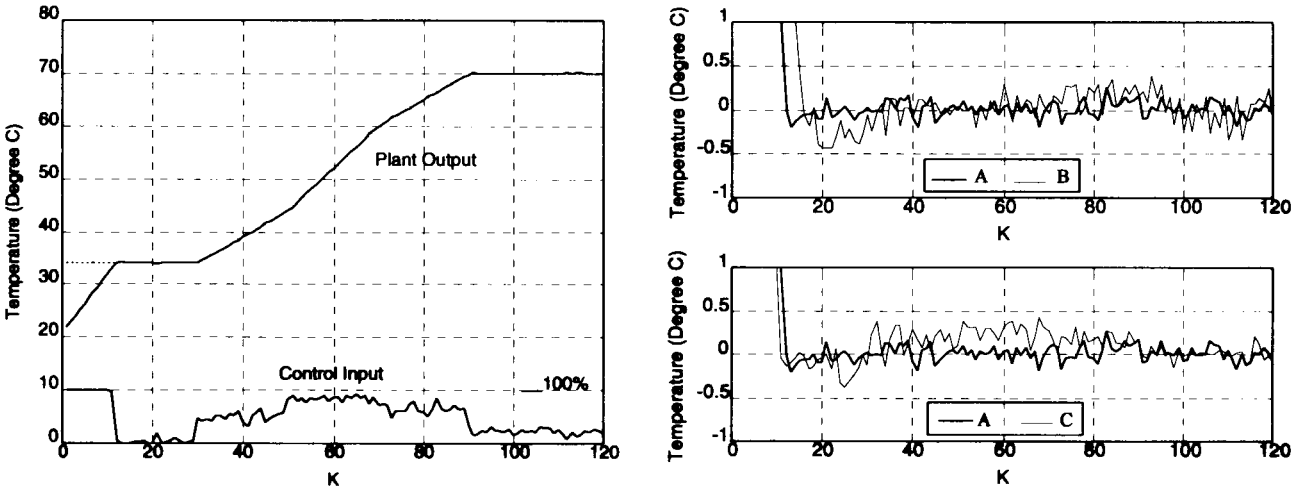
In the second set of experiments, the tracking capability of the three controllers with respect to ramp-points is studied. We define three ramp-point regions as

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



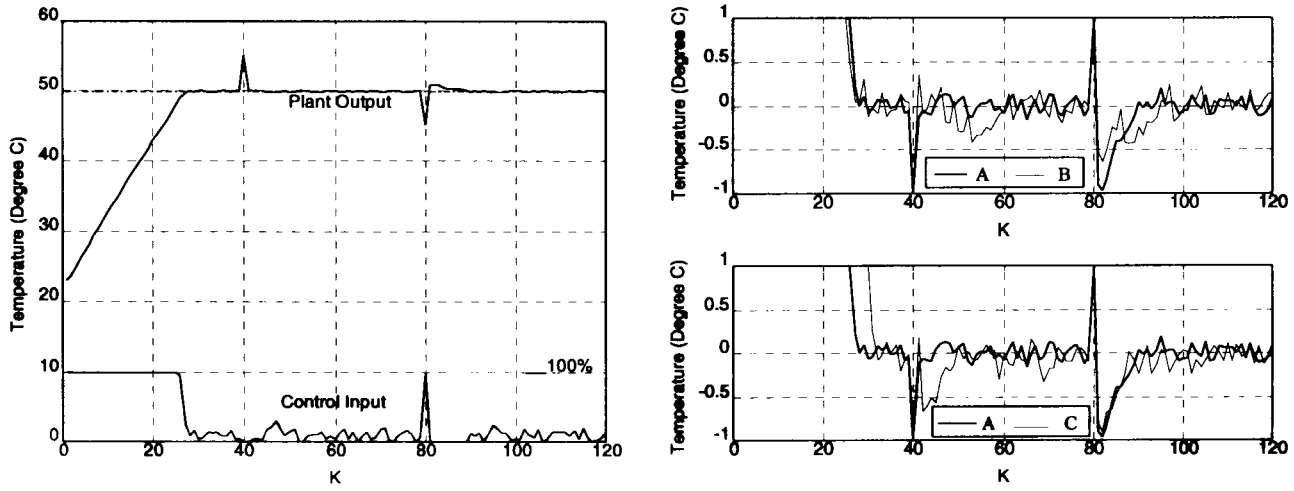
(a) (b)

Fig. 10. Regulation performance of the three controllers for the water bath system. (a) NFIN. (b) Corresponding errors: A: NFIN; B: FLC; C: PID.



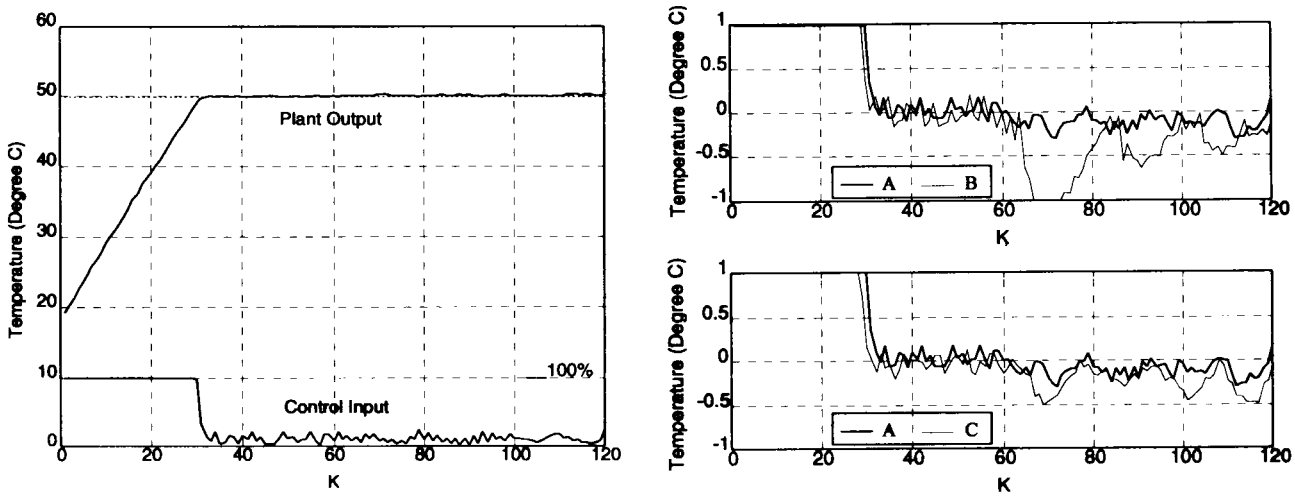
(a) (b)

Fig. 11. Tracking performance of the three controllers for the water bath system. (a) NFIN. (b) Corresponding errors: A: NFIN; B: FLC; C: PID.



(a) (b)

Fig. 12. Behavior of the three controllers under the impulse noise for the water bath system. (a) NFIN. (b) Corresponding errors: A: NFIN; B: FLC; C: PID.



(a) (b)

Fig. 13. Behavior of the three controllers when a change occurs in the water bath system dynamics. (a) NFIN. (b) Corresponding errors: A: NFIN; B: FLC; C: PID.

For the NFIN control, the same on-line training scheme and learning parameters are used as those used in the first set of experiments. In implementing the FLC and the PID controller, the same controller parameters are also used as in the first set of experiments. The tracking performance of the NFIN controller is shown in Fig. 11(a), and the tracking errors of the three controllers are shown in Fig. 11(b). As shown in the error curves, the NFIN has the smallest error. The FLC and PID show poor tracking-control capability.

The third set of experiments 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. To make fair comparisons among the controllers, two impulse noise values 5.0 °C and –5.0 °C are added to the plant output at the fortieth and eightieth sampling instants, respectively. A set-point of 50.0 °C is performed in this set of experiments. The behavior of the NFIN control system under the influence of impulse noise is shown in Fig. 12(a), and the corresponding errors of the three controllers are shown in Fig. 12(b). It is observed that the NFIN performs quite well. It recovers very quickly and steadily after the presentation of the two impulse noises. However, the FLC and PID are affected seriously by the impulse noise and unpredictable behavior occurs.

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 $0.7 * u(k-2)$ is added to the plant input after the sixtieth sample in the fourth set of experiments. Although it is difficult to imagine a real plant behavior in such a drastic way, it can help to understand their robustness. A set-point of 50.0 °C is used in this set of experiments. The behavior of the NFIN controller when there is a change in the plant dynamics is shown in Fig. 13(a), and the corresponding errors of the three controllers are shown in Fig. 13(b). For the NFIN, the on-line training scheme helps to improve its performance. We find that a little fluctuation resulted after the sixtieth sample. However, the FLC and PID are affected more seriously after the sixtieth sample.

For the experimental studies, a summary of comparisons among the three controllers on the water bath temperature-control system is shown in Table III. In implementing the FLC, the number of rules and membership functions have to be decided and tuned by hand. As for the PID controller, the parameters K_P, K_I, and K_D also have to be decided properly. For the FLC and PID controllers, therefore, they usually require a long time in design for achieving good performance. In implementing the NFIN controller, however, no controller parameters have to be decided in advance. We only need to choose proper training patterns and the input vector of the NFIN controller. After off-line and on-line training, it can achieve good control performance. In general, the NFIN usually spends a

TABLE III
SUMMARY OF COMPARISONS AMONG THE THREE CONTROLLERS ON THE EXPERIMENTAL WATER BATH SYSTEM

Criteria	NFIN	FLC	PID
Design Effort	Self-tuning, no controller parameters have to be tuned *a priori.*	Rules and membership functions have to be decided and tuned *a priori.*	Controller parameters K_P, K_I and K_D have to be tuned *a priori.*
Regulation Performance SAE = $\sum_{k=20}^{120} \lvert y_{ref}(k) - y_p(k) \rvert$	SAE = 221.7828	SAE = 227.8678	SAE = 228.1601
Tracking Performance SAE = $\sum_{k=30}^{90} \lvert y_{ref}(k) - y_p(k) \rvert$	SAE = 4.6111	SAE = 7.3998	SAE = 12.7488
Influence of Impulse Noises SAE = $\sum_{k=40}^{120} \lvert y_{ref}(k) - y_p(k) \rvert$	SAE = 19.7440	SAE = 22.3247	SAE = 23.0082
Effect of Change in Plant Dynamics SAE = $\sum_{k=60}^{120} \lvert y_{ref}(k) - y_p(k) \rvert$	SAE = 7.0567	SAE = 27.4009	SAE = 14.4143

relatively short time training to reach good performance. From the regulation performance of the three controllers, the NFIN controller has the shortest rise-time and the best regulation-control performance of all. From the tracking performance of the three controllers, the NFIN controller has the smallest errors of all in the tracking path. Finally, when testing the robustness, since the NFIN controller adopts the on-line training scheme, it also shows better performance under impulse noise and changes in the plant dynamics.

VI. Conclusion

In this paper, an NFIN is proposed and applied to the water bath temperature-control problem. The on-line self-constructing property of the NFIN reduces the design efforts when it is applied as a controller, as compared to other existing intelligent controllers such as neural networks or fuzzy controllers. Also, this property makes it able to deal with the problem of a changing environment or plant, which cannot be handled perfectly by conventional controllers like the PID controller. As for the network size, the input clustering method in the precondition part and the flexible linear-combination structure in the consequent part of fuzzy rules, reduce not only the number of rules but also the number of parameters in each rule. These advantages of the NFIN have been verified by the water bath temperature-control experiments performed in this paper. Future work is to investigate a recurrent NFIN controller such that we need no additional efforts to find the input order of the plant, as the recurrency of the network can deal with this problem implicitly.

References

[1] K. J. Åstrom and B. Wittenmark, *Adaptive Control*. Reading, MA: Addison-Wesley, 1989.

[2] K. S. Narendra, R. Ortega, and P. Dorato, Eds., *Advances in Adaptive Control*. New York: IEEE Press, 1991.

[3] K. Ogata, *Discrete-Time Control Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[4] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[5] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controllers—Parts I, II," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 404–435, Mar./Apr. 1990.

[6] C. Ling and T. Edgar, "A new fuzzy gain scheduling algorithm for process control," in *Proc. ACC'92*, Chicago, IL, 1992, pp. 2284–2290.

[7] C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 46–53, Feb. 1993.

[8] W. T. Miller III, R. S. Sutton, and P. J. Werbos, Eds., *Neural Networks for Control*. Cambridge, MA: M.I.T. Press, 1990.

[9] M. Khalid, S. Omatu, and R. Yusof, "MIMO furnace control with neural networks," *IEEE Trans. Contr. Syst. Technol.*, vol. 1, pp. 238–245, July 1993.

[10] J. Tanomaru and S. Omatu, "Process control by on-line trained neural controllers," *IEEE Trans. Ind. Electron.*, vol. 39, pp. 511–521, Dec. 1992.

[11] M. Khalid and S. Omatu, "A neural network controller for a temperature control system," *IEEE Contr. Syst.*, vol. 12, pp. 58–64, June 1992.

[12] J. S. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, May 1993.

[13] L. X. Wang, *Adaptive Fuzzy Systems and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[14] S. Nagata, M. Sekiguchi, and K. Asakawa, "Mobile robot control by a structured hierarchical neural network," *IEEE Contr. Syst.*, vol. 10, pp. 69–76, June 1990.

[15] D. Psaltis, A. Sideris, and A. Yamamura, "A multilayered neural network controller," *IEEE Contr. Syst.*, vol. 8, pp. 17–21, Apr. 1988.

[16] M. Sugeno and K. Tanaka, "Successive identification of a fuzzy model and its applications to prediction of a complex system," *Fuzzy Sets Syst.*, vol. 42, no. 3, pp. 315–334, 1991.

[17] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, Jan. 1985.

[18] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Networks*, vol. 3, pp. 724–740, Sept. 1992.

[19] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and and decision system," *IEEE Trans. Comput.*, vol. 40, pp. 1320–1336, Dec. 1991.

[20] C. T. Lin, *Neural Fuzzy Control Systems with Structure and Parameter Learning*. Singapore: World Scientific, 1994.

[21] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hall, May 1996.

[22] C. J. Lin and C. T. Lin, "Reinforcement learning for ART-based fuzzy adaptive learning control networks," *IEEE Trans. Neural Networks*, vol. 7, pp. 709–731, May 1996.

Sensor-Based Fuzzy Reactive Navigation of a Mobile Robot Through Local Target Switching

W. L. Xu and S. K. Tso

***Abstract*—Fuzzy reactive control, incorporating a local target-switching scheme, is applied to the automatic navigation of an intelligent mobile robot in an unknown and changing environment. Sensed-ranging signals and relative target position signals are input to the fuzzy controller. The steering angle and the velocity change are inferred to drive the mobile robot. A reactive rulebase governing the robot behavior is synthesized from the human heuristics with respect to various situations of environment. A local target-switching scheme is proposed to serve as a front-end processor of the fuzzy active controller and to deal with the local trapping and wandering cycle problem in the navigation of a behavior-based mobile robot. The algorithm is described together with some particular considerations about implementation. Efficiency and effectiveness of the proposed approach are verified through simulation and experiments conducted on a Nomad 200 mobile robot.**

I. Introduction

FOR MOBILE robots operating in an unknown and changing environment, coping with uncertainty is one of the most challenging problems. There are several approaches proposed to solve the robot navigation problem, including the model-based methods [1]–[5] and the fuzzy logic (FL)-based [6]–[18] and neural network (NN)-based [19]–[22] reactive methods. FL-based or NN-based navigation is fast in reaction and tolerant to sensing noises. Robot reactions are decided through the reasoning of various types of qualitative behaviors using FL or through prediction using an NN, which is trained by a database quantitatively representing the behaviors. In the FL control of a mobile robot using a fuzzy expert system [6], recognition rules are proposed to classify the road shapes derived from the camera images, and control rules are designed to produce the

Manuscript received February 8, 1997; revised November 9, 1997 and December 5, 1998. This work was supported by the City University of Hong Kong under Strategic Research Grant 7000731.

W. L. Xu is with the Institute of Technology and Engineering, Massey University, Palmerston North, New Zealand (e-mail: w.l.xu@massey.ac.nz)

S. K. Tso is with the Centre for Intelligent Design, Automation, and Manufacturing, Faculty of Science and Engineering, City University of Hong Kong, Kowloon, Hong Kong (e-mail: mewxu@citu.edu.hk).

Publisher Item Identifier S 1094-6977(99)05331-6.