Journal of Intelligent & Fuzzy Systems 17 (2006) 145–157
IOS Press

Temperature control using neuro-fuzzy controllers with compensatory operations and wavelet neural networks

Cheng-Jian Lin[a,*], Chi-Yung Lee[b] and Cheng-Chung Chin[c]
[a]*Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung County, 413 Taiwan*
[b]*Department of Computer Science and Information Engineering, Nankai Institute of Technology, Nantou County 542, Taiwan*

Abstract. This paper addresses a Compensatory Wavelet Neuro-Fuzzy System (CWNFS) for temperature control. The proposed CWNFS model is five-layer structure, which combines the traditional Takagi-Sugeno-Kang (TSK) fuzzy model and the wavelet neural networks (WNN). We adopt the non-orthogonal and compactly supported functions as wavelet neural network bases. Besides, the compensatory fuzzy reasoning method is used in adaptive fuzzy operations that can make the fuzzy logic system more adaptive and effective. An on-line learning algorithm, which consists of structure learning and parameter learning, is presented. The structure learning is based on the degree measure to determine the number of fuzzy rules and wavelet functions. The parameter learning is based on the gradient descent method to adjust the shape of membership function, compensatory operations and the connection weights of WNN. Simulation results have been given to illustrate the performance and effectiveness of the proposed model.

Keywords: Temperature control, TSK-type fuzzy model, wavelet neural networks, on-line learning, gradient descent, compensatory operation

1. Introduction

In recent years, the concept of fuzzy logic and artificial neural network for control problem has been grown into a popular research topic [9,10,17]. The reason is that the classical control theory usually requires a mathematical model for designing the controller. The inaccuracy of mathematical modeling of the plants usually degrades the performance of the controller, especially for nonlinear and complex control problems [1]. On the contrary, the fuzzy logic controller (FLC's) and the artificial neural network controller, they offer a key advantage over traditional adaptive control systems. That is, they do not require mathematical models of the plants. The traditional neural networks can learn from data and feedback, but the meaning associated with each neuron and each weight in the network is not easily understood. Alternatively, the fuzzy logical models are easy to appreciate, because it uses linguistic terms and the structure of if-then rules. However, as compared with the neural networks, learning ability is lack of fuzzy logic. In contrast to the pure neural network or fuzzy system, the fuzzy neural network representations have emerged as a powerful approach to the solution of many problems [5,11,13,15].

In this paper, the compensatory wavelet neuro-fuzzy system (CWNFS) is proposed to overcome the disadvantages of the FLC and the artificial neural network. Each fuzzy rule corresponding to a WNN consists of single-scaling wavelets. The non-orthogonal and compactly supported functions are adopted as wavelet neural network bases. The

*Corresponding author. Fax: +886 4 23742375; E-mail: cjlin@mail.cyut.edu.tw.

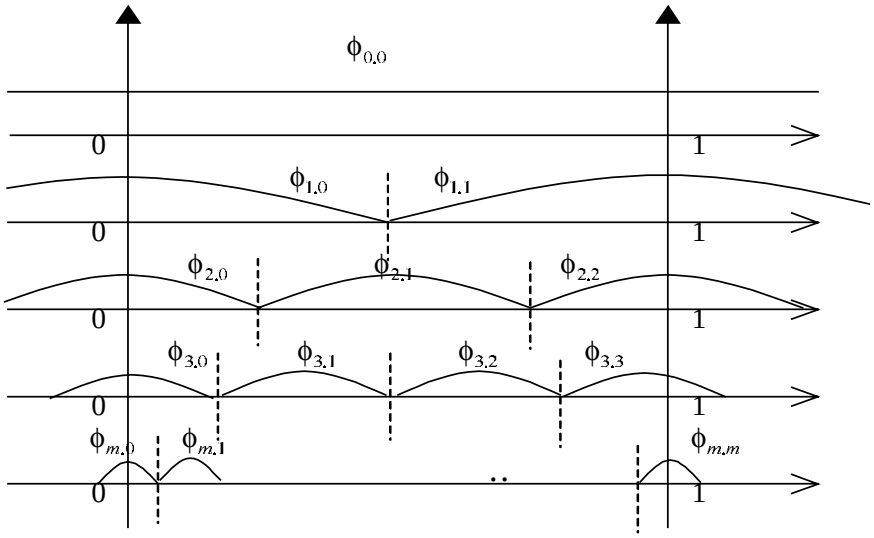


Fig. 1. Wavelet bases are over-complete and compactly supported.

compensatory fuzzy reasoning method is used in adaptive fuzzy operations that can make the fuzzy logic system more adaptive and effective. Therefore, an effective neural fuzzy system should be able not only to adaptively adjust fuzzy membership functions but also to dynamically optimize adaptive fuzzy operators. An on-line structure/parameter learning algorithm is performed concurrently in the CWNFS. The CWNFS model cannot only find itself optimal or almost optimal network size, but the parameters of the CWNFS are adjusted via a proposed dynamic training algorithm. Finally, the encouraging results are obtained via series simulations of a water bath temperature control system. We also compare our approach with other methods in the literature early.

2. The structure of a compensatory wavelet neuro-fuzzy system

2.1. Description of wavelet neural network

To generate the novel form of the TSK model, the CWNFS is integrates the traditional TSK-type fuzzy model and the WNN [4]. Each fuzzy rule corresponding to a WNN consists of single-scaling wavelets. The non-orthogonal and compactly supported functions are adopted in finitely range as wavelet bases [3]. The shape and position of wavelet bases are shown in Fig. 1.

Neural networks employing wavelet neurons are refereed to *wavelet neural networks*. According to Yamakawa et al. [18], we propose a new type of wavelet neural network model that is shown in Fig. 2. Consider n inputs vectors $\{x_1, x_2, \ldots, x_n\} \in R^n$ and single-output $Y \in R$, respectively. This model is obtained by replacing a sigmoidal activation function with single-scaling wavelets. The wavelet neural networks are characterized by weighted and wavelet base. Each linear synaptic weight of wavelet basis is adjustable by learning. Noted that, the ordinary wavelet neural network model applications, it is often useful to normalize the input vectors into the interval $[0, 1]$. When the input signal fire up the interval of wavelet neurons, and the $\phi_{a.b}(x_i)$ function be calculated by

$$\begin{cases} \phi(x_i) = \cos(x_i) & -0.5 \leqslant x_i \leqslant 0.5 \\ 0 & (\text{otherwise}) \end{cases}, \quad \phi_{a.b}(x_i) = \cos(ax_i - b) \tag{1}$$

Above equation is formulating the non-orthogonal wavelet neurons in finitely range, the symbol b is a shifting parameter, the maximum value of witch equals the corresponding scaling parameter a. Obviously, a crisp value $\Psi_{a.b}$ obtained as follows:

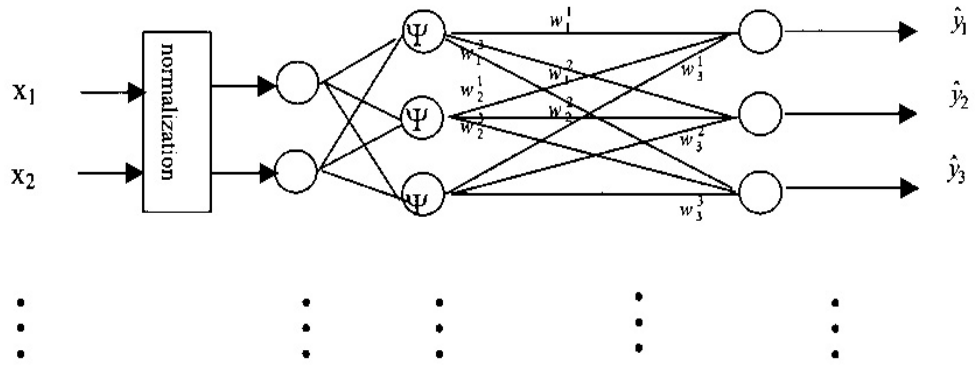


Fig. 2. Schematic diagram of the WNN.

$$\psi_{a.b} = \frac{\sum_{i=1}^{n} \phi_{a.b}(x_i)}{|X|} \tag{2}$$

where |X| means the number of input dimension. The final output of the wavelet neural networks is:

$$\hat{y}_j = \sum_{k=1}^{M} w_{jk} \psi_{a.b} \tag{3}$$

where $\hat{y}_j$ is the local output of the WNN for output Y and jth rule, and the link weight w_{jk} is the output action strength associated with in the jth rule and kth $\Psi_{a.b}$. The symbol M denotes the number of wavelets, which are equal the number of existing fuzzy rules in CWNFS.

2.2. Description of the CWNFS model

According to [12], a novel CWNFS model, which combines fuzzy model, compensatory operations, and wavelet neural networks, can be written in the following general form:

$$R_j : [\text{IF } x_1 \text{ is } A_{1j} \text{ and } \ldots \text{and } x_n \text{ is } A_{nj}^{1-\gamma_j + r_j/n} \text{ THEN } \quad \hat{y}_j = \sum_{k=1}^{M} w_{jk} \psi_{a.b} \tag{4}$$

where x_i is input variable, $\hat{y}_j$ is output variable, A_{nj} is linguistic term of the precondition part, and n is number of input variables.

The structure of the CWNFS is shown in Fig. 3. It is a five-layer structure. Each node in layer 1 is an input node; these nodes only pass the input signal to next layer. Each node in layer 2 acts as membership function representing the term of the respective input-linguistic variables. The Gaussian function is adopted as the membership function. Layer 3 is a rule node representing the precondition part of one fuzzy logic rule. We use a compensatory fuzzy operator mentioned in [12] to perform IF-condition matching of fuzzy rules. As a result, the output function of each inference nodes is

$$O_j^{(3)} = \left(\prod_i I_{ij}^{(2)} \right)^{1-\gamma_j + \frac{\gamma_j}{n}} \tag{5}$$

where $\gamma_j \in [0, 1]$ is called the compensatory degree. Nodes in layer 4 only receive the signal, which is $\hat{y}_j$ from output of wavelet neural network model. The node in layer 5 computes the output signal Y. The output node together with links connected to it act as a defuzzifier. The mathematical function is

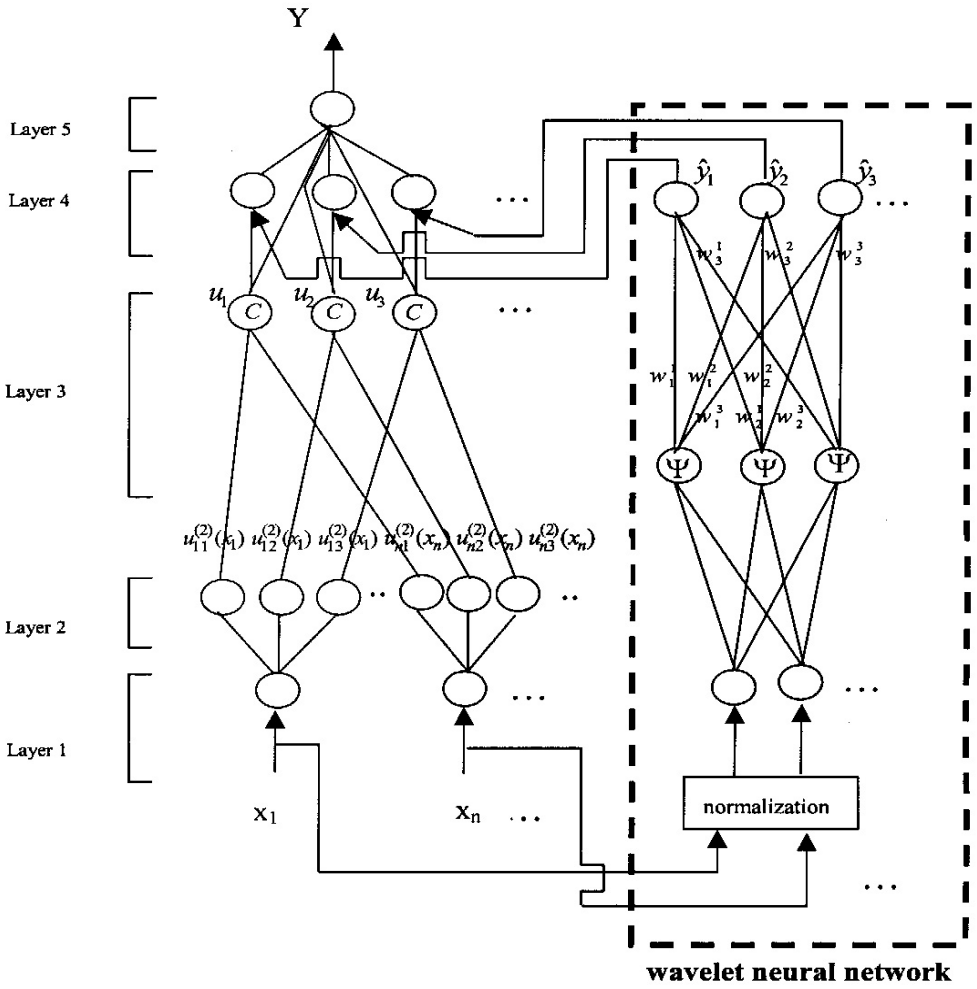


Fig. 3. Schematic diagram of CWNFS model.

$$O^{(5)} = Y = \frac{\sum_{j=1}^{M} O_j^{(4)} O_j^{(3)}}{\sum_{j=1}^{M} O_j^{(3)}} = \frac{\sum_{j=1}^{M} (w_{j1}\psi_{0.0} + w_{j2}\psi_{1.0} + w_{j3}\psi_{1.1}\cdots) O_j^{(3)}}{\sum_{j=1}^{M} O_j^{(3)}} \tag{6}$$

where $O_j^{(4)}$ is the output of the local model of the WNN model for an output Y and the jth rule, $O_j^{(3)}$ is the output of layer 3, and $O^{(5)}$is the output of the CWNFS. The structure of the proposed CWNFS model is different from the CNFN model [12]. In [12], it is four-layer structure and the consequent part of the rules is singleton. In the proposed CWNFS model, the consequent part of the rules is a nonlinear function of input linguistic variables. This study adopts the wavelet neural network to the consequent part of the rules. The local properties of wavelets in the CWNFS model enable arbitrary functions to be approximated more effectively.

3. An on-line learning algorithm for CWNFS controller

In this paper, an on-line learning algorithm, which consists of structure learning and parameter learning, is used concurrently for constructing the CWNFS. The structure learning scheme is used to decide proper fuzzy partition,

the number of the rule nodes and the wavelet neural networks while the parameter learning scheme is used to tune the adjustable parameters. The detail of the learning algorithm is presented below:

3.1. The structure learning scheme

Initially, there are no rules in CWNFS model; the first task in structure learning is to decide when to generate a new rule. Geometrically, a rule is corresponding to a cluster in the input space with m_{ij} and σ_{ij} representing the mean and variance of that cluster. For each incoming pattern x_i the strength a rule is fired can be regard as the degree of the incoming pattern belongs to the corresponding cluster. An input data x_i with higher firing strength means its spatial location is nearer the center of cluster than those with smaller strength. Based on this concept, the firing strength obtained from Eq. (5) is used as the degree measure. The criterion of generating a new fuzzy rule for new incoming data is

$$F_{\max} = \max_{1 \leqslant j \leqslant M} F_j = \max_{1 \leqslant j \leqslant M} O_j^{(3)} \tag{7}$$

If $F_{\max} \leqslant \overline{F}$, then a new rule is generated where $\overline{F} \in (0,1)$ is a pre-specified threshold that should be decayed during the learning process limiting the size of CWNFS. Once the new rule is generated, the next step is to assign initial value of the free parameters. The structure learning method is similar to [8,12].

3.2. The parameter learning scheme

After the network structure is adjusted according to the current training pattern, the network then enters the parameter learning scheme to turn the adjustable parameters of the network optimally based on the same training pattern. Notice that the following parameter learning is performed on the whole networks after structure learning, no matter whether the nodes (links) are newly added or are existent originally. Since the learning process involves the determination of the vector which minimize a given energy function. The gradient of the energy function with respect to the vector is computed and the vector is adjusted along the negative gradient. The energy function E is defined as

$$E = \frac{1}{2}(Y - Y^{des})^2 \tag{8}$$

where Y is the model output and Y^{des} is the desired output. Assuming that w is the adjustable parameter in a nodded, the general learning rule used is

$$w(t+1) = w(t) + \eta \left(-\frac{\partial E}{\partial w} \right) \tag{9}$$

where η is the learning rate. To show the learning rules, we derive the rules layer by layer. For clarity, we consider the single output case.

Layer 5: The error to be propagated to the preceding layer is

$$\delta^{(5)} = -\frac{\partial E}{\partial O^{(5)}} = \frac{-\partial \frac{1}{2}(Y - Y^{des})^2}{\partial O^{(5)}} = Y^{des} - Y \tag{10}$$

Layer 4: The link weight of wavelet neural network is update by

$$\Delta w_{jk} = -\eta_w \frac{\partial E}{\partial w_{jk}} = \left[-\eta_w \frac{\partial E}{\partial O^{(5)}} \right] \cdot \left[\frac{\partial O^{(5)}}{\partial O_{lj}^{(4)}} \right] \cdot \left[\frac{\partial O_{lj}^{(4)}}{\partial w_{jk}} \right] = \eta_w \cdot \delta^{(5)} \cdot \frac{O_j^{(3)} \psi_{a.b}(x)}{\sum_j O_j^{(3)}} \tag{11}$$

where η_w is the learning rate.

Layer 3: In this layer only the error term needs to be calculated and propagated

$$\delta^{(3)} = -\frac{\partial E}{\partial O_j^{(3)}} = \left[-\frac{\partial E}{\partial O^{(5)}} \right] \cdot \left[\frac{\partial O^{(5)}}{\partial O_j^{(3)}} \right] = \delta^{(5)} \cdot \left(O_{lj}^{(4)} \sum_j O_j^{(3)} - \sum_j O_{lj}^{(4)} O_j^{(3)} \right) \frac{1}{\left(\sum_j O_j^{(3)} \right)^2} \tag{12}$$

To eliminate the constraint $\gamma_j \in [0,1]$, we redefine γ_j as follows:

$$\gamma_j = \frac{c_j^2}{c_j^2 + d_j^2} \tag{13}$$

Then we have

$$\Delta c_j = \eta_c \left\{ \frac{2c_j d_j^2}{[c_j^2 + d_j^2]^2} \right\} \delta^{(3)} \left[\frac{1}{n} - 1 \right] \ln \left[\prod_i O_{ij}^{(2)} \right] O_j^{(3)} \tag{14}$$

$$\Delta d_j = -\eta_d \left\{ \frac{2c_j^2 d_j}{[c_j^2 + d_j^2]^2} \right\} \delta^{(3)} \left[\frac{1}{n} - 1 \right] \ln \left[\prod_i O_{ij}^{(2)} \right] O_j^{(3)} \tag{15}$$

In all above formulas, η_c and η_d are the learning rate of the parameter c_j and the parameter d_j.

Layer 2 : In this layer, the error term is computed as follows:

$$\delta^{(2)} = -\frac{\partial E}{\partial O_{ij}^{(2)}} = \left[-\frac{\partial E}{\partial O^{(5)}} \right] \cdot \left[\frac{\partial O^{(5)}}{\partial O_j^{(3)}} \right] \cdot \left[\frac{\partial O_j^{(3)}}{\partial O_{ij}^{(2)}} \right] = \delta^{(3)} \cdot \prod_{i \neq j} O_j^{(3)} \tag{16}$$

The updated law of m_{ij} is

$$\Delta m_{ij} = -\eta_m \frac{\partial E}{\partial m_{ij}} = \left[-\eta_m \frac{\partial E}{\partial O_{ij}^{(2)}} \right] \cdot \left[\frac{\partial O_{ij}^{(2)})}{\partial m_{ij}} \right] = \eta_m \delta^{(2)} \cdot \frac{\partial O_{ij}^{(2)}}{\partial m_{ij}} \tag{17}$$

The updated law of σ_{ij} is

$$\Delta \sigma_{ij} = -\eta_m \frac{\partial E}{\partial \sigma_{ij}} = \left[-\eta_\sigma \frac{\partial E}{\partial O_{ij}^{(2)}} \right] \cdot \left[\frac{\partial O_{ij}^{(2)}}{\partial \sigma_{ij}} \right] = \eta_\sigma \delta^{(2)} \cdot \frac{\partial O_{ij}^{(2)}}{\partial \sigma_{ij}} \tag{18}$$

where η_m and η_σ are the learning rate parameter of the mean and the standard deviation of the Gaussian function, respectively.

4. Control of water bath temperature system

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

$$\frac{dy(t)}{dt} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{RC} \tag{19}$$

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. (19) into discrete-time form with some reasonable approximation. The system

$$y(t+1) = e^{-\alpha Ts} y(k) + \frac{\frac{\beta}{\alpha}(1 - e^{-\alpha Ts})}{1 + e^{0.5y(k)-40}} u(k) + [1 - e^{-\alpha Ts}] y_0. \tag{20}$$

is obtained, where α and β are some constant values describing R and C. The system parameters used in this example are $\alpha = 1.0015e^{-4}$, $\beta = 8.67973e^{-3}$ and $Y_0 = 25.0$ (°C), which were obtained from a real water bath plant in [17]. The input$u(k)$ is limited to 0 and 5 V represent voltage unit. The sampling period is $Ts = 30$. The system configuration is shown in Fig. 4, where y_{ref} is the desired temperature of the controlled plant. Recently, many researchers [2,6,7,14] use various different methods for solving the temperature control problems. The control approach in this paper is different from [2,6]. Chen and Pao [2] compute the derivative of the model's output with respect to its input by means of the back-propagation process, which evaluates the transpose of the network Jacobian

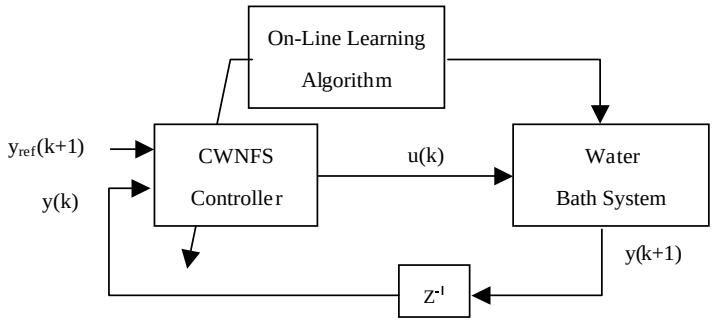


Fig. 4. Flow diagram of using CWNFS controller for solving the temperature control problem.

at the network's current input vector. This usually implies that we need a model for the plant and the Jacobian matrix obtained from the model, which could be a neural network, a neuro-fuzzy system, or another appropriate mathematical description of the plant. As a result, propagating errors between actual and desired plant outputs back through the forward model produces error in the control signal, which can be used to train another network to be a controller [6].

By implement the on-line training scheme for CWNFS, a sequence of random input signals $u_{rd}(k)$ limited to 0 and 5 V is injected directly into the simulated system described in Eq. (20). The 120 training patterns are chosen from the input-outputs characteristic in order to cover the entire reference output. The initial temperature of the water is 25°C, and the temperature rises progressively when random input signals are injected. For the CWNFS, the learning rate $\eta_w = \eta_m = \eta_\sigma = \eta_c = \eta_d = 0.05$, the initial variance $\sigma_{init} = 6$ and the prespecified threshold $\overline{F} = 0.05$ are chosen. After training, there are 12 fuzzy rules generated. The obtained fuzzy rules with a compensatory degree are shown as follows:

$$R^1: \text{If } [I_1 \text{ is } A_{1,1}(24.6024, 3.6861) \text{ and } I_2 \text{ is } A_{2,1}(26.9721, 7.1613)]^{0.37}$$
$$\text{Then } \hat{y}_1^1 = 0.7056\varphi_{0.0} + 0.0029\varphi_{1.0} - 0.0422\varphi_{1.1} + 0.0592\varphi_{2.0} + 0.0739\varphi_{2.1} - 0.0336\varphi_{2.2} + 0.0164\varphi_{3.0} + 0.0669\varphi_{3.1} + 0.0432\varphi_{3.2} - 0.0497\varphi_{3.3} + 0.0605\varphi_{4.0} - 0.0241\varphi_{4.1}$$

$$R^2: \text{If } [I_1 \text{ is } A_{1,2}(33.0827, 3.0011) \text{ and } I_2 \text{ is } A_{2,2}(31.5810, 7.7886)]^{0.93}$$
$$\text{Then } \hat{y}_2^1 = 0.0608\varphi_{0.0} + 2.9094\varphi_{1.0} + 0.0806\varphi_{1.1} - 0.0338\varphi_{2.0} - 0.0027\varphi_{2.1} - 0.0344\varphi_{2.2} - 0.0550\varphi_{3.0} + 0.0669\varphi_{3.1} - 0.0736\varphi_{3.2} - 0.0551\varphi_{3.3} - 0.0105\varphi_{4.0} + 0.0384\varphi_{4.1}$$

$$R^3: \text{If } [I_1 \text{ is } A_{1,3}(45.9519, 5.0566) \text{ and } I_2 \text{ is } A_{2,3}(37.5630, 5.1166)]^{0.73}$$
$$\text{Then } \hat{y}_3^1 = -0.0573\varphi_{0.0} + 0.0874\varphi_{1.0} + 17.1185\varphi_{1.1} - 0.0910\varphi_{2.0} - 0.0918\varphi_{2.1} + 0.0962\varphi_{2.2} - 0.0323\varphi_{3.0} + 0.0463\varphi_{3.1} - 0.0734\varphi_{3.2} - 0.0047\varphi_{3.3} - 0.0321\varphi_{4.0} + 0.0250\varphi_{4.1}$$

$$R^4: \text{If } [I_1 \text{ is } A_{1,4}(38.7137, 4.7003) \text{ and } I_2 \text{ is } A_{2,4}(45.9058, 5.1329)]^{0.94}$$
$$\text{Then } \hat{y}_4^1 = -0.0996\varphi_{0.0} - 0.0251\varphi_{1.0} - 0.0464\varphi_{1.1} - 12.5963\varphi_{2.0} + 0.0857\varphi_{2.1} + 0.0823\varphi_{2.2} - 0.0022\varphi_{3.0} + 0.0176\varphi_{3.1} - 0.0320\varphi_{3.2} + 0.0014\varphi_{3.3} - 0.0036\varphi_{4.0} + 0.0027\varphi_{4.1}$$

$$R^5: \text{If } [I_1 \text{ is } A_{1,5}(56.3778, 5.1274) \text{ and } I_2 \text{ is } A_{2,5}(48.0105, 4.7223)]^{0.25}$$
$$\text{Then } \hat{y}_5^1 = -0.0042\varphi_{0.0} + 0.0153\varphi_{1.0} + 0.0179\varphi_{1.1} + 0.0053\varphi_{2.0} + 13.7761\varphi_{2.1} + 0.0632\varphi_{2.2} - 0.0280\varphi_{3.0} - 0.0370\varphi_{3.1} - 0.0241\varphi_{3.2} - 0.0865\varphi_{3.3} - 0.0471\varphi_{4.0} + 0.0842\varphi_{4.1}$$

$$
\begin{aligned}
R^6 &: \text{If } [I_1 \text{ is } A_{1,6}(62.8080, 8.3051) \text{ and } I_2 \text{ is } A_{2,6}(66.1094, 6.6011)]^{0.41} \\
&\text{Then } \hat{y}_6^1 = -0.0763\varphi_{0.0} + 0.0380\varphi_{1.0} - 0.0943\varphi_{1.1} + 0.0567\varphi_{2.0} + 0.0369\varphi_{2.1} \\
&\quad + 2.2305\varphi_{2.2} - 0.0607\varphi_{3.0} - 0.0893\varphi_{3.1} + 0.0720\varphi_{3.2} - 0.0015\varphi_{3.3} \\
&\quad + 0.0747\varphi_{4.0} - 0.0416\varphi_{4.1}
\end{aligned}
$$

$$
\begin{aligned}
R^7 &: \text{If } [I_1 \text{ is } A_{1,7}(74.4883, 5.5007) \text{ and } I_2 \text{ is } A_{2,7}(68.1826, 7.6617)]^{0.95} \\
&\text{Then } \hat{y}_7^1 = 0.0028\varphi_{0.0} + 0.0722\varphi_{1.0} + 0.0858\varphi_{1.1} - 0.0837\varphi_{2.0} + 0.0739\varphi_{2.1} \\
&\quad - 0.0315\varphi_{2.2} + 3.5460\varphi_{3.0} - 0.0142\varphi_{3.1} + 0.0327\varphi_{3.2} + 0.0699\varphi_{3.3} \\
&\quad + 0.0614\varphi_{4.0} + 0.0655\varphi_{4.1}
\end{aligned}
$$

$$
\begin{aligned}
R^8 &: \text{If } [I_1 \text{ is } A_{1,8}(77.4191, 6.6648) \text{ and } I_2 \text{ is } A_{2,8}(78.0521, 5.7519)]^{0.35} \\
&\text{Then } \hat{y}_8^1 = 0.0553\varphi_{0.0} - 0.0123\varphi_{1.0} - 0.0703\varphi_{1.1} - 0.0936\varphi_{2.0} + 0.0774\varphi_{2.1} \\
&\quad - 0.0907\varphi_{2.2} - 0.0672\varphi_{3.0} + 2.6022\varphi_{3.1} - 0.0951\varphi_{3.2} - 0.0450\varphi_{3.3} \\
&\quad - 0.0597\varphi_{4.0} - 0.0852\varphi_{4.1}
\end{aligned}
$$

$$
\begin{aligned}
R^9 &: \text{If } [I_1 \text{ is } A_{1,9}(49.3033, 4.1272) \text{ and } I_2 \text{ is } A_{2,9}(54.9042, 3.9521)]^{0.12} \\
&\text{Then } \hat{y}_9^1 = -0.0126\varphi_{0.0} + 0.0186\varphi_{1.0} + 0.0649\varphi_{1.1} - 0.0651\varphi_{2.0} + 0.0903\varphi_{2.1} \\
&\quad - 0.0978\varphi_{2.2} + 0.0797\varphi_{3.0} + 0.0506\varphi_{3.1} - 7.3473\varphi_{3.2} - 0.0747\varphi_{3.3} \\
&\quad + 0.0500\varphi_{4.0} - 0.0143\varphi_{4.1}
\end{aligned}
$$

$$
\begin{aligned}
R^{10} &: \text{If } [I_{1.10} \text{ is } \mu(29.5346, 5.6278) \text{ and } I_{2.10} \text{ is } \mu(28.9791, 6.5138)]^{0.72} \\
&\text{Then } \hat{y}_{10}^1 = -0.0133\varphi_{0.0} - 0.0522\varphi_{1.0} + 0.0096\varphi_{1.1} + 0.0757\varphi_{2.0} + 0.0594\varphi_{2.1} \\
&\quad - 0.0107\varphi_{2.2} + 0.0405\varphi_{3.0} - 0.0732\varphi_{3.1} + 0.0019\varphi_{3.2} + 1.7451\varphi_{3.3} \\
&\quad + 0.0476\varphi_{4.0} + 0.0710\varphi_{4.1}
\end{aligned}
$$

$$
\begin{aligned}
R^{11} &: \text{If } [I_1 \text{ is } A_{1,11}(44.0306, 2.8736) \text{ and } I_2 \text{ is } A_{2,11}(48.5759, 1.2315)]^{0.96} \\
&\text{Then } \hat{y}_{11}^1 = 0.0830\varphi_{0.0} + 0.0382\varphi_{1.0} + 0.0861\varphi_{1.1} - 0.0607\varphi_{2.0} - 0.0223\varphi_{2.1} \\
&\quad - 0.0926\varphi_{2.2} + 0.0707\varphi_{3.0} - 0.0246\varphi_{3.1} - 0.0520\varphi_{3.2} + 0.0209\varphi_{3.3} \\
&\quad - 0.9817\varphi_{4.0} - 0.0653\varphi_{4.1}
\end{aligned}
$$

$$
\begin{aligned}
R^{12} &: \text{If } [I_1 \text{ is } A_{1,12}(34.8565, 3.4774) \text{ and } I_2 \text{ is } A_{2,12}(35.2144, 4.4854)]^{0.5} \\
&\text{Then } \hat{y}_{12}^1 = -0.0324\varphi_{0.0} - 0.0283\varphi_{1.0} + 0.0880\varphi_{1.1} + 0.0404\varphi_{2.0} + 0.0155\varphi_{2.1} \\
&\quad - 0.0193\varphi_{2.2} + 0.0520\varphi_{3.0} + 0.0252\varphi_{3.1} + 0.0809\varphi_{3.2} + 0.0851\varphi_{3.3} \\
&\quad - 0.0171\varphi_{4.0} + 2.1611\varphi_{4.1}
\end{aligned}
$$

In this paper, we compare the CWNFS controller to the PID controller [16], the manually designed fuzzy controller and the self-constructing fuzzy neural network (SCFNN) [15]. Each of the three controllers is applied to the water bath temperature control system. The comparison performance measures include set-points regulation, the influence of impulse noise, and a large parameter variation in the system.

For the PID control, a velocity-form discrete PID controller [16] is used and is described by

$$
\begin{aligned}
\Delta u(k) &= K \left\{ e(k) - e(k-1) + \frac{T_s}{2T_i}[e(k) + e(k-1)] + \frac{T_d}{T_s}[e(k) - 2e(k-1) + e(k-2)] \right\} \\
&= K_P[e(k) - e(k-1)] + K_I e(k) + K_D[e(k) - 2e(k-1) + e(k-2)]
\end{aligned} \tag{21}
$$

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

Table 1
Fuzzy rule table formulated for the water bath temperature control system

		Error, e(t)						
		NL	NM	NS	ZE	PS	PM	PL
Change error, ce(t)	PL				PL	PL	PL	PL
	PM				PM	PM	PM	PL
	PS			PS	PS	PS	PM	PL
	ZE	NL	NM	NS	ZE	PS	PM	PL
	NS			NS	NS	NS		
	NM				NM			
	NL				NL			

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 chosen as 80 and 70, respectively. For the above designed PID controller, we have tried our best to achieve their respective best performance through several trial-and-error experiments.

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: $GE = 1/15$, $GCE = 1/15$, $GU = 450$.

Recently, Lin et al. [15] presented a self-constructing fuzzy neural network (SCFNN) for control problems. The SCFNN controller is a standard four-layer structure. Each node in layer 3 performs the product operation. The consequence of each fuzzy rule is a singleton value. The output node sums all incoming signals to obtain inferred result. An on-line learning algorithm was proposed to decide the structure of fuzzy rules and turn the adjustable parameters through the backpropagation algorithm. The structure of the proposed CWNFS controller is difference from [15]. Our model is five-layer structure, using the compensatory operation in layer 3, and adopting the wavelet neural network as consequent part of each fuzzy rule.

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

The first task is to control the simulated system to follow three set-points.

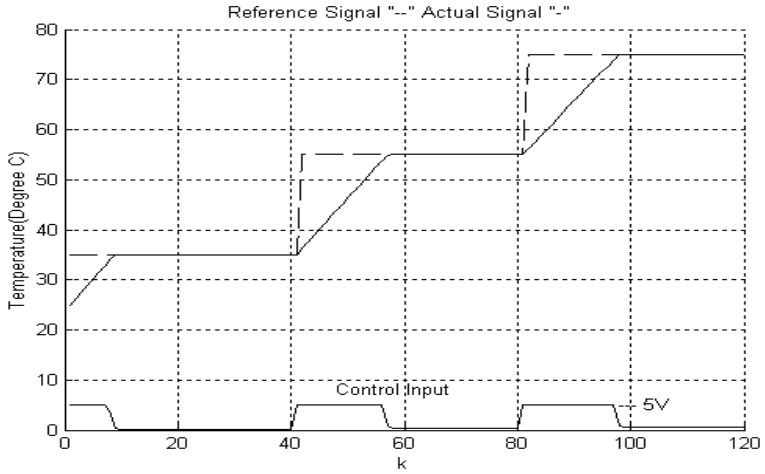
$$y_{ref}(k) = \begin{cases} 35^{\circ}\mathrm{C}, \text{ for } k \leqslant 40 \\ 55^{\circ}\mathrm{C}, \text{ for } 40 < k \leqslant 80 \\ 75^{\circ}\mathrm{C}, \text{ for } 80 < k \leqslant 120. \end{cases} \tag{22}$$

The regulation performance of the CWNFS model is shown in Fig. 5(a). We also test the regulation performance by using SCNNF controller [15]. The error curves of CWNFS controller and SCFNN controller between $k = 80$ and $k = 100$ are shown in Fig. 5(b). In this figure, the CWNFS controller obtains smaller errors than the SCNNF controller. To test their regulation performance, a performance index, sum of absolute error (SAE), is defined by

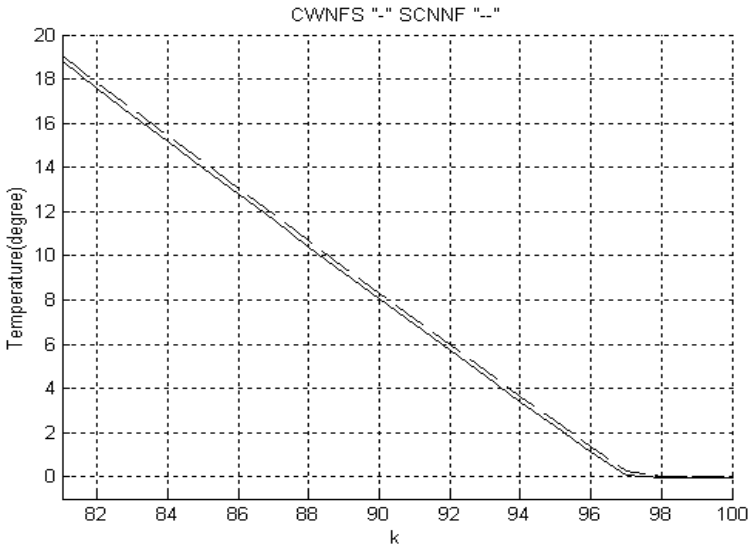
$$SAE = \sum_{k} |y_{ref}(k) - y(k)| \tag{23}$$

where $y_{ref}(k)$ and $y(k)$ are the reference output and the actual output of the simulated system, respectively. The SAE values of the CWNFS controller, the PID controller, the fuzzy controller and SCFNN controller are 352.95, 418.5, 401.5, and 356.41, which are shown in the first row of Table 2.

The second set of simulations is carried out for the purpose of studying the noise-rejection ability of the four controllers when some unknown impulse noise is imposed on the process. One impulse noise value $-5\ ^{\circ}\mathrm{C}$ is added



(a)



(b)

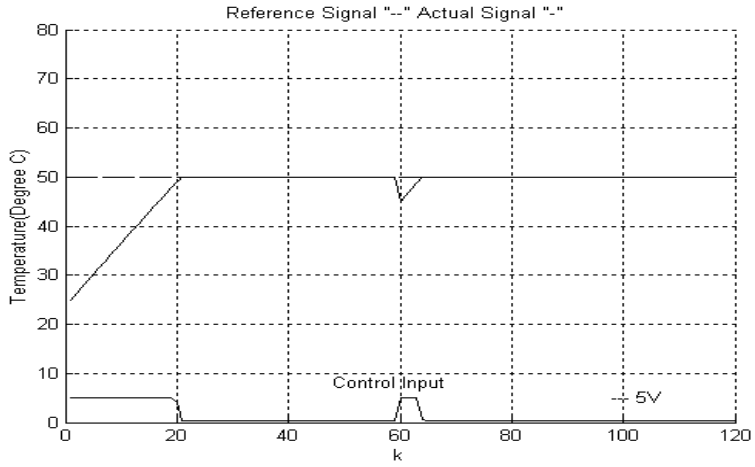
Fig. 5. (a) Final regulation performance of the CWNFS controller for water bath system. (b) The error curves of CWNFS controller and SCFNN controller between $k = 80$ and $k = 100$.

to the plant output at the sixtieth sampling instant. A set-point of $50\,^{\circ}$C is performed in this set of simulations. For the CWNFS 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 CWNFS controller under the influence of impulse noise and the corresponding errors are shown in Fig. 6(a)–(b). The SAE values of the CWNFS controller, the PID controller, the fuzzy controller, and SCFNN are 273.25, 311.5, 275.8, and 280.5, which are shown in the second row of Table 2. It is observed that the CWNFS controller performs quite well. It recovers very quickly and steadily after the presentation of the impulse noise.

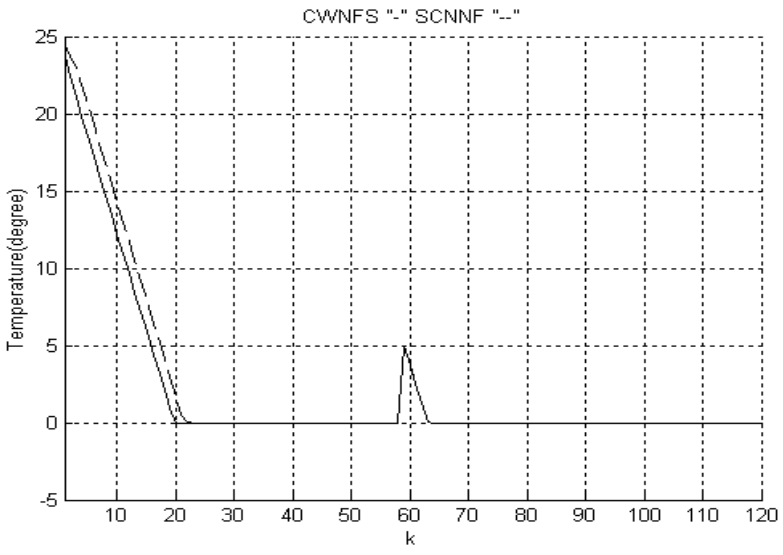
One common characteristic of many industrial-control processes is that their parameters tend to change in an unpredictable way. To test the robustness of the four controllers, a value of $0.7 * u(k-2)$ is added to the plant input after the sixtieth sample in the fourth set of simulations. A set-point of $50\,^{\circ}$C is used in this set of simulations. For

Table 2
Performance comparison of various controllers

$SAE = \sum_{k=1}^{120} \lvert y_{ref}(k) - y(k) \rvert$	**CWNFS controller**	PID controller [16]	manually designed fuzzy controller	SCFNN controller [15]
Regulation performance	**352.95**	418.5	401.5	356.41
Influence of impulse noise	**273.25**	311.5	275.8	280.50
Effect of change in plant dynamics	**262.51**	322.2	273.5	270.21



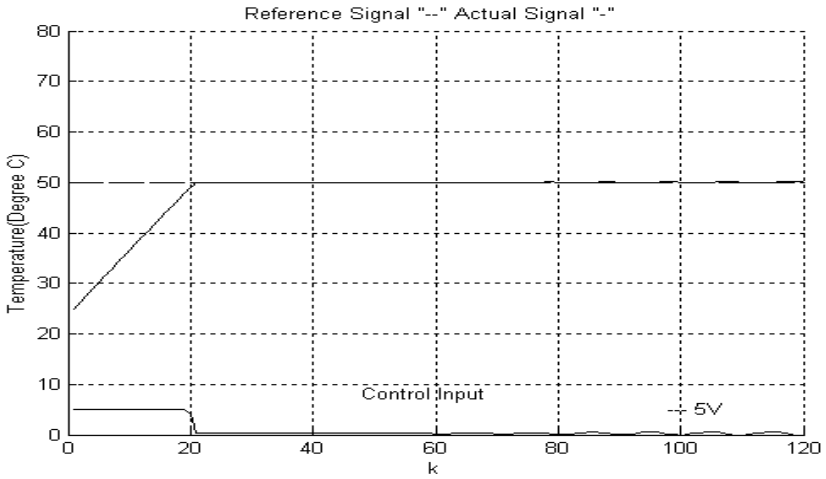
(a)



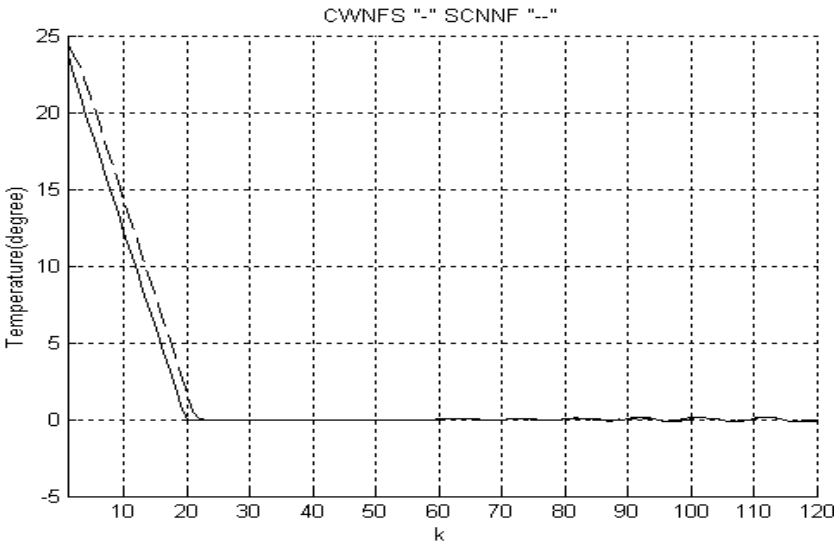
(b)

Fig. 6. (a) Behavior of the CWNFS controller under the impulse noise for water bath system. (b) The error curves of CWNFS controller and SCFNN controller.

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 CWNFS controller when there is a change in the plant dynamics are shown in Fig. 7(a). The corresponding errors of the CWNFS and SCNNF controllers are shown in Fig. 7(b). The SAE values of the CWNFS controller, the PID controller, the fuzzy controller, and SCFNN are 262.51, 322.2,



(a)



(b)

Fig. 7. (a) Behavior of the CWNFS controller when a change occurs in the water bath system. (b) The error curves of CWNFS controller and SCFNN controller.

273.5, and 270.21, which are shown in the third row of Table 2. The results show the good control and disturbance rejection capabilities of the trained CWNFS controller in the water bath system.

For the aforementioned simulation results, Table 2 has shown that the proposed CWNFS controller has better performance than that of other methods. For the fuzzy controller, the numbers 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 fuzzy and PID controllers, therefore, they usually require a long time in design for achieving good performance. In the CWNFS controller, however, no controller parameters have to be decided in advance. We only need to choose propose training patterns of the CWNFS controller. Although the structure of CWNFS controller is more complicated than the fuzzy and PID controllers, in general, the CWNFS controller usually spends a relatively short time in design for achieving good performance. This study attempts to emphasize

the methodology and control abilities of the proposed CWNFS model. In the future, we will apply the proposed CWNFS controller on a real water bath temperature control system.

5. Conclusion

In this paper, the new CWNFS controller, which combines TSK-type fuzzy model and wavelet neural networks, is proposed and applied to the water bath temperature system. The CWNFS controller can automatically construct and adjust free parameters itself by performing online supervised structure/parameter learning schemes concurrently. Finally, computer simulation results have shown that the proposed CWNFS controller has better performance than that of other methods.

Acknowledgement

This research is supported by the National Science Council of R.O.C. under grant NSC 93-2213-E-324-008.

References

[1] K.J. Åstrom and B. Wittenmark, *Adaptive Control*, Reading, MA: Addison-Wesley, 1989.
[2] V.C. Chen and Y.H. Pao, *Learning Control with Neural Networks*, Proc. of International Conf. on Robotics and Automation, 1989, 1448–1453.
[3] I. Daubechies, Orthonormal Bases of Compactly Supported Wavelets, *Comm. Pur. Appl. Math.* **41** (1998).
[4] D.W.C. Ho, P.A. Zhang and J. Xu, Fuzzy Wavelet Networks for Function Learning, *IEEE Trans. on Fuzzy Systems* **9** (2001), 200–211.
[5] J.-S.R. Jang, ANFIS: Adaptive-Network-Based Fuzzy Inference System, *IEEE Trans. on Syst., Man, and Cybern.* **23** (1993), 665–685.
[6] J.-S.R. Jang, C.T. Sun and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Ch. 17, Prentice-Hall, 1997.
[7] C.F. Juang, J.Y. Lin and C.T. Lin, Genetic reinforcement learning through symbiotic evolution for fuzzy controller design, *IEEE Trans. on Systems, Man, and Cybernetics* **B30** (2000), 290–302.
[8] C.F. Juang, A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms, *IEEE Trans. on Fuzzy Systems* **10** (2002), 155–170.
[9] C.L. Karr and E.J. Gentry, Fuzzy Control of pH Using Genetic Algorithms, *IEEE Trans. on Fuzzy Syst.* **1** (1993), 46–53.
[10] C.C. Lee, Fuzzy Logic in Control Systems: Fuzzy Logic Controllers – Parts I, II, *IEEE Trans. on Syst., Man, Cybern.* **20** (1990), 404–435.
[11] C.J. Lin and C.C. Chin, *A Wavelet-Based Neuro-Fuzzy System and Its Applications*, Proc. IEEE Int. Joint Conference on Neural Networks, Oregon, Portland, 2003, 20–24.
[12] C.J. Lin and C.H. Chen, Nonlinear System Control Using Compensatory Neuro-Fuzzy Networks, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **E86-A** (2003), 2309–2316.
[13] C.T. Lin and C.S.G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System*, NJ: Prentice-Hall, 1996.
[14] C.T. Lin, C.F. Juang and C.P. Li, Temperature control with a neural fuzzy inference network, *IEEE Trans. on Systems, Man, and Cybernetics* **C29** (1999), 440–451.
[15] F.J. Lin, C.H. Lin and P.H. Shen, Self-Constructing Fuzzy Neural Network Speed Controller for Permanent-Magnet Synchronous Motor Drive, *IEEE Trans. Fuzzy Syst.* **9** (2001), 751–759.
[16] C.L. Phillips and H.T. Nagle, *Digital Control System Analysis and Design*, Prentice Hall, 1995.
[17] J. Tanomaru and S. Omatu, Process Control by On-Line Trained Neural Controllers, *IEEE Trans. on Ind. Electron.* **39** (1992), 511–521.
[18] T. Yamakawa, E. Uchino and T. Samatsu, Wavelet Neural Networks Employing Over-Complete Number of Compactly Supported Non-Orthogonal Wavelets and Their Applications, *Proc. of IEEE Conf. on Neural Networks* **3** (1994), 1391–1396.
[19] Y.Q. Zhang and A. Kandel, Compensatory Neurofuzzy Systems with Fast Learning Algorithms, *IEEE Transactions on Neural Networks* **9** (1998), 83–105.