International Mathematical Forum, 1, 2006, no. 18, 853-866

A TSK-Type Quantum Neural Fuzzy Network for Temperature Control

Cheng-Jian Lin [1]

Dept. of Computer Science and Information Engineering
Chaoyang University of Technology
Taichung County,413 Taiwan,R. O. C.

Cheng-Hung Chen

Dept. of Electrical and Control Engineering
National Chiao-Tung University
Hsinchu, 300 Taiwan, R.O.C.

Chi-Yung Lee

Dept. of Computer Science and Information Engineering
Nankai Institute of Technology
Nantou County, 542 Taiwan,R.O.C.

Abstract. In this paper, a TSK-type quantum neural fuzzy network (TQNFN) for temperature control is proposed. The TQNFN model is a five-layer structure, which combines the traditional Takagi-Sugeno-Kang (TSK). Layer 2 of the TQNFN model contains quantum membership functions, which are multilevel activation functions. Each quantum membership function is composed of the sum of sigmoid functions shifted by quantum intervals. A self-constructing learning algorithm, which consists of the self-clustering algorithm (SCA) and the backpropagation algorithm, is also proposed. The proposed the SCA method is a fast, one-pass algorithm for a dynamic estimation of the number of clusters in an input data space. The backpropagation algorithm is used to tune the adjustable parameters. Simulation results have been given to illustrate the performance and effectiveness of the proposed model.

Keywords: Temperature control, TSK-type fuzzy model, quantum function, self-clustering method, neural fuzzy network.

[1] Corresponding Author. Tel: +886-4-23323000 Ext. 4408 Fax: +886-4-23742375 Email: cjlin@mail.cyut.edu.tw (C.-J. Lin)

1 Introduction

The concept of fuzzy logic and artificial neural network for control problem has been grown into a popular research topic in recent years [1]-[3]. 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 [4]. 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 neural fuzzy network representations have emerged as a powerful approach to the solution of many problems [5]-[9].

Recently, quantum neural networks (QNNs) for the limitations of conventional neural networks (NNs) were developed [10]-[12]. Conventional NNs and QNNs satisfy the requirements outlined in [13] for a universal function approximator. More specifically, QNNs can identify overlaps between data due to their ability to approximate any arbitrary membership profile up to any degree of accuracy. However, QNNs and NNs are generally disadvantaged by their "black box" format, lack a systematic way to determine the appropriate model structure, have no localizability, and converge slowly.

In this paper, a TSK-type quantum neural fuzzy network (TQNFN) is proposed. The TQNFN model is a five-layer structure, which combines the traditional Takagi-Sugeno-Kang (TSK). Layer 2 of the TQNFN model contains quantum membership functions, which are multilevel activation functions. Each quantum membership function is composed of the sum of sigmoid functions shifted by quantum intervals. The quantum intervals add an additional degree of freedom that can be exploited during the learning process to capture and quantify the structure of the input space.

A self-constructing learning algorithm for the TQNFN is proposed, as follows. First, a structure learning scheme is used to determine proper input space partitioning and to find the mean of each cluster. Second, a supervised learning scheme is used to adjust the parameters to obtain the desired outputs. The proposed learning algorithm uses the self-clustering algorithm (SCA) to perform structure learning and the backpropagation algorithm to perform parameter learning. We evaluate the performance of the proposed TQNFN model using series simulations of a water bath temperature control system.

2 The Structure of the TSK-Type Quantum Neural Fuzzy Network

The fuzzy if-then rule shown below is used by the TQNFN:

$$R_j: \text{ IF } x_1 \text{ is } Q_{1j} \cdots \text{ and } x_n \text{ is } Q_{nj}, \text{ THEN } y \text{ is } a_{0j} + \sum_{i=1}^{n} a_{ij} x_i \tag{1}$$

where x_i and y are the input and output variables, respectively; Q_{ij} is the linguistic term of the precondition part with quantum membership function μ_Q; a_{0j} and a_{ij} are the parameters of consequent part; n is the number of input dimensions; R_j is jth fuzzy rule.

The membership function of the precondition part discussed in this paper is different from the typical Gaussian membership function [9]-[13]. We adopt the quantum membership function to approximate desired results. Therefore, the response of the jth quantum membership function for the ith feature vector can be written as

$$Q_{ij} = \frac{1}{n_s} \sum_{r=1}^{n_s} [(\frac{1}{1 + exp(-\beta(x_i - m_{ij} + |\theta_j^r|))}) U(x_i; -\infty, m_{ij})$$

$$+ (\frac{exp(-\beta(x_i - m_{ij} + |\theta_j^r|))}{1 + exp(-\beta(x_i - m_{ij} + |\theta_j^r|))}) U(x_i; m_{ij}, \infty)] \tag{2}$$

where β is the slope factor, θ_j^r is the jump positions, m_{ij} is the center of the quantum membership function, and n_s is the number of levels in the quantum membership function. Figure 1 shows the response of a three-level quantum membership function.

The structure of the TSK-type quantum neural fuzzy network (TQNFN), which is systematized into n input variables, p-term nodes for each input variable, one output node, and np membership function nodes, is shown in Figure 2. We shall introduce the operation functions of the nodes in each layer of the TQNFN model. In the following description, $u^{(l)}$ denotes an output of a node in the lth layer.

$Layer1(InputNode)$: No computation is done in this layer. Each node in this layer is an input node, which corresponds to one input variable and which only transmits input values to the next layer directly.

$$u_i^{(1)} = x_i \tag{3}$$

$Layer2(MembershipFunctionNode)$: Nodes in this layer correspond to one linguistic label of the input variables in layer1 and a unit of memory, i.e., the membership value specified the degree to which an input value and a unit of

memory belongs a fuzzy set is calculated in layer 2. The quantum membership function, the operation performed in layer 2 is

$$u_{ij}^{(2)} = \frac{1}{n_s}\sum_{r=1}^{n_s}[(\frac{1}{1+exp(-\beta(u_i^{(1)} - m_{ij} + |\theta_j^r|))})U(u_i^{(1)}; -\infty, m_{ij})$$

$$+(\frac{exp(-\beta(u_i^{(1)} - m_{ij} + |\theta_j^r|))}{1+exp(-\beta(u_i^{(1)} - m_{ij} + |\theta_j^r|))})U(x_i; m_{ij}, \infty)] \tag{4}$$

where β is the slope factor, θ_j^r is the jump positions, m_{ij} is the center of the quantum membership function, and n_s is the number of levels in the quantum membership function.

*Layer*3(*RuleNode*): Nodes in this layer represent the preconditioned part of one fuzzy logic rule. They receive one-dimensional membership degrees of the associated rule from nodes of a set in layer 2. Here, we use a product operator mentioned above to perform IF-condition matching of fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = [\prod_i u_{ij}^{(2)}] \tag{5}$$

where the $\prod_i u_{ij}^{(2)}$ of a rule node represents the firing strength of its corresponding rule.

*Layer*4(*ConsequentNode*): Nodes in this layer are called the consequent nodes. The input to a node of layer 4 is the output delivered from layer 3, and the other inputs are the input variables from layer 1 as depicted in Figure 2. For this kind of node, we have

$$u_j^{(4)} = u_j^{(3)}(a_{0j} + \sum_{i=1}^{n} a_{ij}x_i) \tag{6}$$

where the summation is over all the inputs and a_{ij} are the corresponding parameter of consequent part.

*Layer*5(*OutputNode*): 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

$$y = u^{(5)} = \frac{\sum_{j=1}^{p} u_j^{(3)}(a_{0j} + \sum_{i=1}^{n} a_{ij}x_i)}{\sum_{j=1}^{p} u_j^{(3)}} \tag{7}$$

where p is the number of fuzzy rule.

3 A Learning Algorithm for the TQNFN Model

In this section, we present a learning algorithm for the proposed TQNFN model. The following two schemes are part of this learning algorithm. First, a structure learning scheme is used to determine proper input space partitioning and to find the mean of each cluster. Second, a supervised learning scheme is used to adjust the parameters for the desired outputs. The proposed learning algorithm uses the self-clustering algorithm (SCA) to perform structure learning and the backpropagation algorithm to perform parameter learning.

3.1 The Self-Clustering Algorithm

Layer 2 of the TQNFN model can be viewed as a function that maps input patterns. Hence, the discriminative ability of these new features is determined by the centers of the quantum membership function. To achieve good classification, centers are best selected based on their ability to provide large class separation.

A clustering method, called the self-clustering algorithm (SCA) [14], is proposed to implement scatter partitioning of the input space. Without any optimization, the online SCA is a fast, one-pass algorithm for a dynamic estimation of the number of clusters in a set of data and for finding the current centers of clusters in the input data space. It is a distance-based connection-clustering algorithm. In any cluster, the maximum distance between a sample point and the cluster center is less than a threshold value which has been set as a clustering parameter and which would affect the number of clusters to be estimated. The details of this step will be discussed in [14].

3.2 The Parameter Learning Algorithm

After the network structure is determined by the self-clustering algorithm, the network then enters the parameter learning phase to adjust the parameters of the network based on the training patterns. The learning process involves minimizing a given cost function. The gradient of the cost function is computed and adjusted along the negative gradient. The backpagation algorithm is used for this supervised learning method. When we consider the single output case for clarity, our goal to minimize the cost function E is defined as

$$E = \frac{1}{2}[y - y^d]^2 \tag{8}$$

where y^d is the desired output and y is the current output. Then the parameter learning algorithm based on backpropagation is described as follows: The error term to be propagated is calculated as

$$\delta_e = -\frac{\partial E}{\partial y} = y^d - y \tag{9}$$

The parameter of consequent part is updated by the amount

$$\Delta a_{0j} = -\frac{\partial E}{\partial a_{0j}} = [-\frac{\partial E}{\partial u^{(5)}}][\frac{\partial u^{(5)}}{\partial u_j^{(4)}}][\frac{\partial u_j^{(4)}}{\partial a_{0j}}] = \frac{\delta_e u_j^{(3)}}{\sum_{j=1}^{p} u_j^{(3)}} \tag{10}$$

and

$$\Delta a_{ij} = -\frac{\partial E}{\partial a_{ij}} = [-\frac{\partial E}{\partial u^{(5)}}][\frac{\partial u^{(5)}}{\partial u_j^{(4)}}][\frac{\partial u_j^{(4)}}{\partial a_{ij}}] = \frac{\delta_e u_j^{(3)} x_i}{\sum_{j=1}^{p} u_j^{(3)}} \tag{11}$$

The parameter of consequent part in the output layer is updated according to the following equation:

$$a_{0j}(t+1) = a_{0j}(t) + \eta_a \Delta a_{0j} \tag{12}$$

$$a_{ij}(t+1) = a_{ij}(t) + \eta_a \Delta a_{ij} \tag{13}$$

where factor η_a is the learning rate parameter of the parameter and t denotes the jth iteration number . The output error (i.e., the difference between the desired output and the current output) is then backpropagated to the quantum function neurons of the hidden layer to update their centers and jump positions. According to the chain rule, the updated center is as follows:

$$\Delta m_{ij} = -\frac{\partial E}{\partial m_{ij}} = [-\frac{\partial E}{\partial u^{(5)}}][\frac{\partial u^{(5)}}{\partial m_{ij}}] \tag{14}$$

The updated jump position is as follows:

$$\Delta \theta_j^r = -\frac{\partial E}{\partial m_{ij}} = [-\frac{\partial E}{\partial u^{(5)}}][\frac{\partial u^{(5)}}{\partial \theta_j^r}] \tag{15}$$

The centers and jump positions of the quantum function neurons in this layer are updated as follows:

$$m_{ij}(t+1) = m_{ij}(t) + \eta_m \Delta m_{ij} \tag{16}$$

$$\theta_j^r(t+1) = \theta_j^r(t) + \eta_a \Delta \theta_j^r \tag{17}$$

where η_m and η_θ are the learning rate parameters of the center and the jump positions of the quantum function neurons, 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

$$\frac{dy(t)}{dt} = \frac{u(k)}{C} + \frac{Y_0 - y(k)}{RC} \tag{18}$$

where $y(k)$ is system output temperature in $\dot{\mathrm{C}}$; $u(k)$ 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 Equation (19) into discrete-time form with some reasonable approximation. The system

$$y(k+1) = e^{\alpha T_s} y(k) + \frac{\frac{\delta}{\alpha}(1 - e^{\alpha Ts})}{1 + e^{0.5y(k)-40}} u(k) + (1 - e^{\alpha T_s}) y_0 \tag{19}$$

is obtained, where α and δ are some constant values describing R and C. The system parameters used in this example are α=1.0015e^{-4} , δ=8.67973e^{-3} and Y_0=25.0($\dot{\mathrm{C}}$), which were obtained from a real water bath plant in [3]. The input $u(k)$ is limited to 0 and 5V represent voltage unit. The sampling period is T_s=30. The system configuration is shown in Figure 3, where y_{ref} is the desired temperature of the controlled plant. The control approach in this paper is different from [15]-[16]. Chen and Pao [15] compute the derivative of the model's output with respect to its input by means of the backpropagation process, which evaluates the transpose of the network Jacobian 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 [16].

By implement the on-line training scheme for TQNFN, a sequence of random input signals $u_{rd}(k)$ limited to 0 and 5V is injected directly into the simulated system described in Equation (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$\dot{\mathrm{C}}$, and the temperature rises progressively when random input signals are injected. For the TQNFN, the initial parameters were set to η_a=η_m=η_θ=0.3, and the prespecified threshold D_{thr}=0.6 is used. After 10000 training iterations, there are 3 fuzzy rules generated.

In this paper, we compare the TQNFN controller to the PID controller [17], the manually designed fuzzy controller and the self-constructing fuzzy neural

network (SCFNN) [8]. 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 [17] is used and is described by

$$\Delta u(k) = K_p[e(k) - e(k-1)]K_I e(k)K_D[e(k) - 2e(k-1) + e(k-2)] \quad (20)$$

where K_p=K-$\frac{K_I}{2}$, K_I=$\frac{KT_s}{T_i}$, and 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 t, 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 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. [8] 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.

For the aforementioned controllers (TQNFN 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.

$$y_{ref}(k) = \begin{cases} f(35\dot{\mathrm{C}}), \text{for } k \leq 40 \\ f(55\dot{\mathrm{C}}), \text{for } 40 < k \leq 80 \\ f(75\dot{\mathrm{C}}), \text{for } 80 < k \leq 120 \end{cases} \tag{21}$$

The regulation performance of the TQNFN model is shown in Figure 4(a). We also test the regulation performance by using SCFNN controller [8]. The error curves of TQNFN controller and SCFNN controller between k=80 and k=100 are shown in Figure 4(b). In this figure, the TQNFN controller obtains smaller errors than the SCFNN controller. To test their regulation performance, a performance index, sum of absolute error (SAE), is defined by

$$SAE = \sum_{k} |y_{ref}(k) - y(k)| \tag{22}$$

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 TQNFN controller, the PID controller, the fuzzy controller, the NN controller and the SCFNN controller are 355.11, 418.5, 401.5, 364.52 and 356.41, which are shown in the first column of Table 2. The proposed TQNFN controller obtains much better SAE value of regulation performance than other methods.

The second set of simulations is carried out for the purpose of studying the noise-rejection ability of the five controllers when some unknown impulse noise is imposed on the process. One impulse noise value -5$\dot{\mathrm{C}}$ is added to the plant output at the sixtieth sampling instant. A set-point of 50$\dot{\mathrm{C}}$ is performed in this set of simulations. For the TQNFN 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 TQNFN controller under the influence of impulse noise and the corresponding errors are shown in Figure 5(a)-(b). The SAE values of the TQNFN controller, the PID controller, the fuzzy controller, the NN controller and the SCFNN are 270.85, 311.5, 275.8, 272.17 and 280.5, which are shown in the second column of Table 2. It is observed that the TQNFN 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 five controllers, a value of $0.7 \times u(k-2)$ is added to the plant input after the sixtieth sample in the third set of simulations. A set-point of 50$\dot{\mathrm{C}}$ is used in this set of simulations. For the TQNFN 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 TQNFN controller when there is a change in the plant dynamics are shown in Figure 6(a). The corresponding errors of the TQNFN and SCFNN controllers are shown in Figure 6(b). The

SAE values of the TQNFN controller, the PID controller, the fuzzy controller, the NN controller and the SCFNN controller are 262.77, 322.2, 273.5, 262.8 and 270.21, which are shown in the third column of Table 2. The results show the good control and disturbance rejection capabilities of the trained TQNFN controller in the water bath system.

For the aforementioned simulation results, Table 2 has shown that the proposed TQNFN 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 TQNFN controller, however, no controller parameters have to be decided in advance. We only need to choose propose training patterns of the TQNFN controller. Although the structure of TQNFN controller is more complicated than the fuzzy and PID controllers, in general, the TQNFN 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 TQNFN model. In the future, we will apply the proposed TQNFN controller on a real water bath temperature control system.

5 Conclusion

In this paper, a TSK-type quantum neural fuzzy network (TQNFN) was proposed for temperature control application. The TQNFN model is a five-layer structure. In the hidden layer, quantum function neurons are adopted. The proposed learning algorithm uses the self-clustering algorithm (SCA) to perform structure learning and the backpropagation algorithm to perform parameter learning. Finally, computer simulation results have shown that the proposed TQNFN controller has better performance than that of other methods.

Acknowledgement
This research is supported by the Ministry of Economic Affairs, Taiwan, R.O.C., under Grant 93-EC-17-A-02-S1-029.

References

1. C. C. Lee, Fuzzy Logic in Control Systems: Fuzzy Logic Controllers-Parts I, II, *IEEE Trans. Syst., Man, Cybern.*, 20 (1990), 404-435.
2. C. L. Karr and E. J. Gentry, Fuzzy Control of pH Using Genetic Algorithms, *IEEE Trans. Fuzzy Syst.*, 1 (1993), 46-53.
3. J. Tanomaru and S. Omatu, Process Control by On-Line Trained Neural Controllers, *IEEE Trans. Ind. Electron.*, 39 (1992), 511-521.

4. K.J. XAstrom and B. Wittenmark, *Adaptive Control. Reading*, MA: Addison-Wesley, 1989.
5. J.-S. R. Jang, ANFIS: Adaptive-Network-Based Fuzzy Inference System, *IEEE Trans. Syst., Man, and Cybern.*, 23 (1993), 665-685.
6. C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System*, NJ: Prentice-Hall, 1996.
7. C. J. Lin and C. T. Lin, An ART-Based Fuzzy Adaptive Learning Control Network, *IEEE Trans. Fuzzy Systems*, 5 (4) (1997), 477-496.
8. 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.
9. C. J. Lin and C. H. Chen, Nonlinear system control using compensatory neuro-fuzzy networks, *IEICE Fundamentals on Electronics, Communications and Computer Sciences*, E86-A (9) (2003), 2309-2316.
10. G. Purushothaman and N. B. Karayiannis, Quantum neural networks (QNNs): inherently fuzzy feedforward neural networks,*IEEE Trans. Neural Networks*, 8 (3) (1997), 679-693.
11. R. Kretzschmar, R. Bueler, N. B. Karayiannis and F. Eggimann, Quantum neural networks versus conventional feedforward neural networks: an experimental study, *Proc. IEEE Int. Conf. Signal Processing*, 1 (2000), 328-337.
12. L. Fei, Z. Shengmei and Z. Baoyu, Quantum neural network in speech recognition, *Proc. IEEE Int. Conf. Signal Processing*, 6 (2000), 1267-1270.
13. M. Leshno, V. Y. Lin, A. Pinkus and S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, *Neural Networks*, 6 (6) (1993), 861-867.
14. C. J. Lin, C. H. Chen and C. Y. Lee, A self-adaptive quantum radial basis function network for classification applications, *Proc. IEEE International Joint Conference Neural Networks*, 4 (2004), 25-29.
15. V. C. Chen and Y. H. Pao, Learning Control with Neural Networks, *Proc. of International Conf. on Robotics and Automation*, (1989), 1448-1453.
16. J.-S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing, Ch. 17*, Prentice-Hall, 1997.
17. C. L. Phillips and H. T. Nagle, *Digital Control System Analysis and Design*, Prentice-Hall, 1995.

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			

Table2: Performance comparison of various controllers.

$SAE = \sum_{k=1}^{120} \lvert y_{ref}(k) - y(k) \rvert$	TQNFN controller	PID controller[17]	Fuzzy controller	NN controller	SCFNN controller[8]
Regulation Performance	355.11	418.5	401.5	364.62	356.41
Influence of Impulse Noise	270.85	311.5	275.8	272.17	280.50
Effect of Change in Plant Dynamics	262.77	322.2	273.5	262.80	268.21
Number of adjustable parameters	33	3	150	241	205

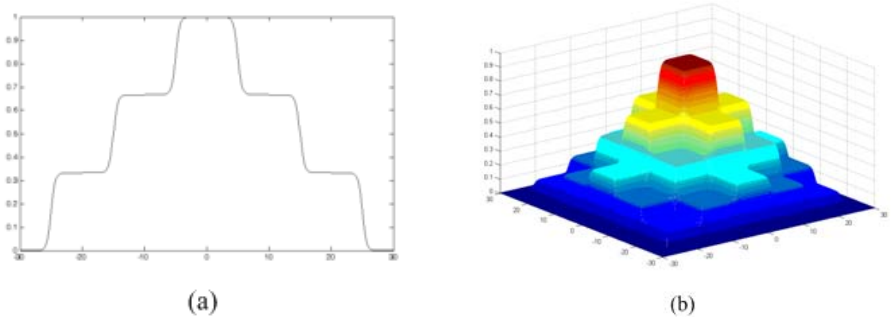

Fig. 1. Quantum membership function shown in (a) one-dimension (b) two-dimensions.

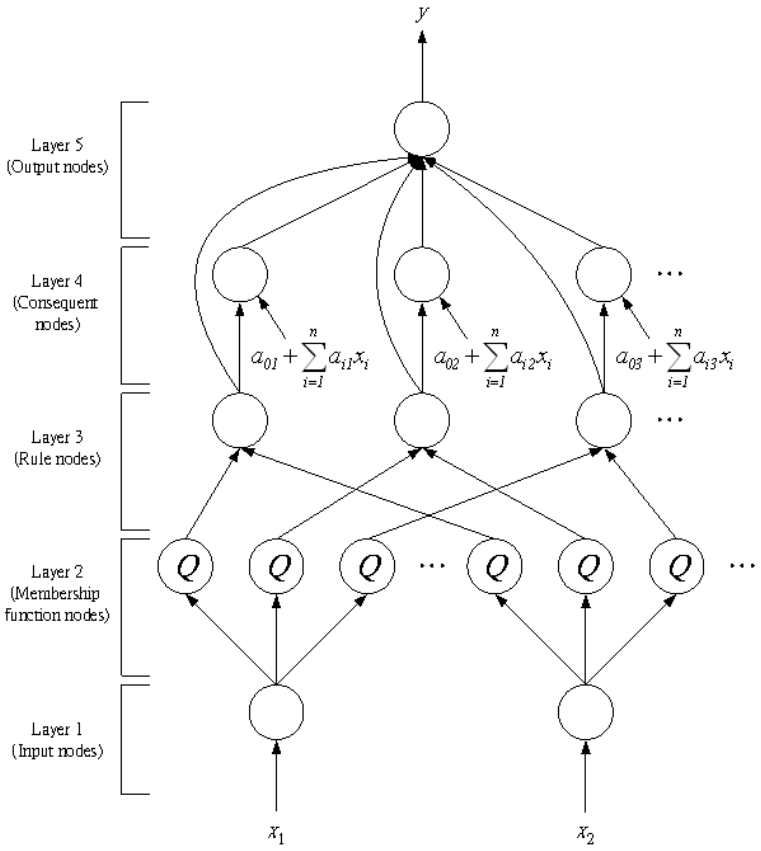

Fig. 2. Structure of the proposed TQNFN.

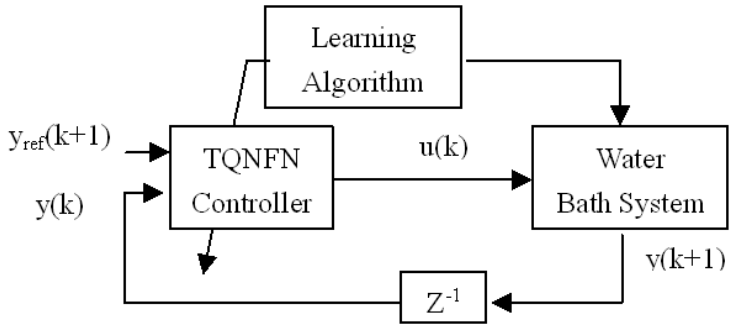

Fig. 3. Flow diagram of using TQNFN controller for solving the temperature control problem.

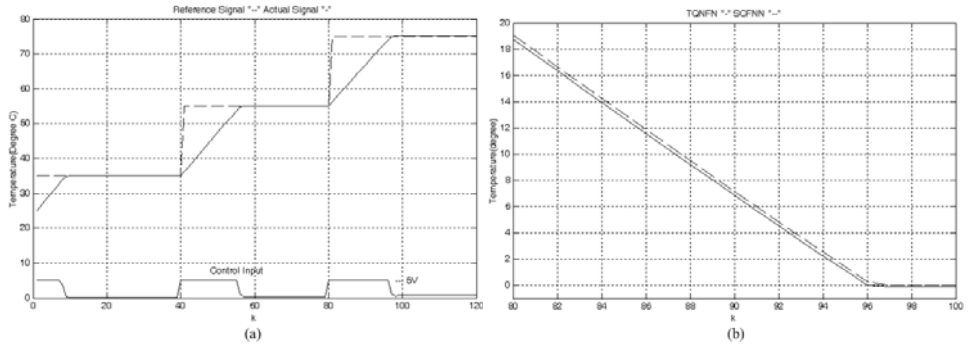

Fig. 4. (a) Final regulation performance of the TQNFN controller for water bath system. (b) The error curves of TQNFN controller and SCFNN controller [8] between k=80 and k=100.

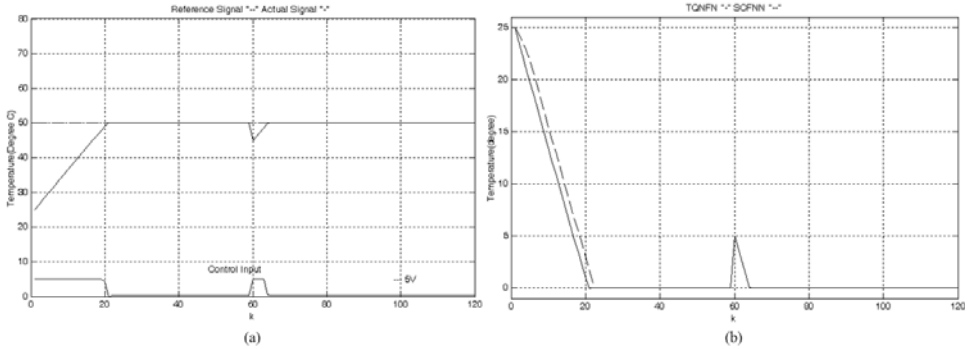

Fig. 5. (a) Behavior of the TQNFN controller under the impulse noise for water bath system. (b) The error curves of TQNFN controller and SCFNN controller [8].

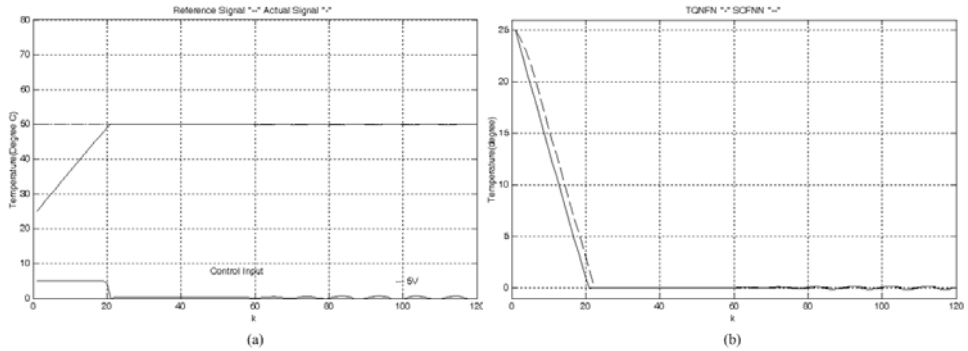

Fig. 6. (a) Behavior of the TQNFN controller when a change occurs in the water bath system. (b) The error curves of TQNFN controller and SCFNN controller [8].