An Adaptive Neuro-fuzzy Approach for Modeling and Control of Nonlinear Systems

Otman M. Ahtiwash and Mohd Zaki Abdulmuin
Center of Factory and Process Automation, Faculty of Engineering,
University of Malaya
50603 Kuala Lumpur, MALAYSIA.
ahtiwash@tm.net.my

Abstract. Fuzzy Inference Systems (FISs) and Artificial Neural Networks (ANNs), as two branches of Soft Computing Systems (SCSs) that pose a human-like inference and adaptation ability, have already proved their usefulness and have been found valuable for many applications [1],[2]. They share a common framework of trying to mimic the human way of thinking and provide an effective promising means of capturing the approximate, inexact nature of the real world process. In this paper we propose an Adaptive Neuro-Fuzzy Logic Control approach (ANFLC) based on the neural network learning capability and the fuzzy logic modeling ability. The approach combines the merits of the both systems, which can handle quantitative (numerical) and qualitative (linguistic) knowledge. The development of this system will be carried out in two phases: The first phase involves training a multi-layered Neuro-Emulator network (NE) for the forward dynamics of the plant to be controlled; the second phase involves on-line learning of the Neuro-Fuzzy Logic Controller (NFLC). Extensive simulation studies of nonlinear dynamic systems are carreid out to illustrate the effectiveness and applicability of the proposed scheme.

1 Introduction

In recent years, Fuzzy Inference Systems (FISs) and Artificial Neural Networks (ANNs) have attracted considerable attention as candidates for novel computational systems because of the variety of the advantages that they offer over conventional computational systems [1]-[5]. Unlike other classical control methods, Fuzzy Logic Control (FLC) and ANNs are more model free controllers, i.e. they do not require exact mathematical model of the controlled system. Moreover, they are becoming well-recognized tools of designing identifiers and controllers capable of perceiving the operating environment and imitating a human operator with high performance.

FLC has the strengths of linguistic control, parallelism, relaxation, flexibility, and robustness. But there has been no systematic approach in implementing the adaptive fuzzy control system. For example, the shape and location of membership function for each fuzzy variable must be obtained by trail-error or heuristic approach. Also, when an expert cannot easily express his knowledge or experience with the linguistic form of (*If-Then*) control rule, it is not easy to construct an efficient control rule base [6].

ANNs have the characteristics of high parallelism, fault-tolerance, and adaptive and learning abilities. But there exist some problems in the neural control; firstly, it is

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2074, pp. 198–207, 2001.

not easy to decide the optimal number of layers and neurons; secondly, the learning algorithm of the neural network has slow convergence speed and thirdly, the neural networks take numerical (quantitative) computations rather than symbolic or linguistic (qualitative) computations [7].

In order to overcome such problems, there have been considerable research efforts to integrate FLC and ANNs for developing what is known as neuro-fuzzy control systems [5]-[10]. The fusion of the two approaches, which can enlarge their individual strengths and overcome their drawbacks, will produce a powerful representation flexibility and numerical processing capability [11]-[13], [14]-[17].

In this paper we present another approach of an adaptive neuro-fuzzy logic control scheme (ANFLC) using the hybrid combination of fuzzy logic and neural networks. The proposed control scheme consists of a neuro-fuzzy logic controller (NFLC) and a neuro-emulator (NE). In the NFLC, the antecedent and consequent parts of the fuzzy rules are constructed using a multi-layered neural network with clustering method. The NFLC is trained to refine its parameters adaptively using error backpropagation learning algorithm (EBP). After constructing the adaptive neuro-fuzzy control system by NFLC and NE, the effectiveness of the proposed scheme will be demonstrated and evaluated by different nonlinear dynamic cases.

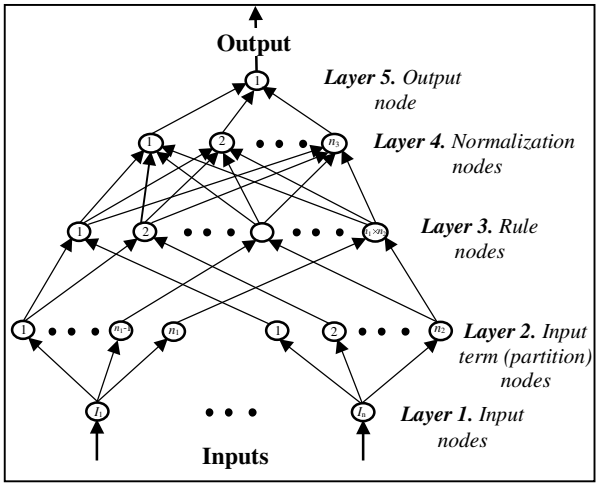

Fig. 1. Topology of the Neuro-Fuzzy model

2 Neuro-fuzzy Logic Controller

Both the FLC and ANNs have been employed together to design a neuro-fuzzy logic controller (NFLC). A fuzzy system with learning ability has been constructed and is trained directly from the input-output data of the plant. Since the NFLC has the property of learning; membership functions and fuzzy rules of the controller can be tuned automatically by the learning algorithm [6],[13],[15]. Learning is based on the performance error, which is evaluated by comparing the process output with the desired or required output.

The NFLC presented here is based on a self-learning FLC. The learning method is basically a special form of the error backpropagation (EPB), which is used for training ANNs. To train the controller, the EBP method is employed to propagate the plant output error signal through different stages in time.

The NFLC architecture is composed of five layers as shown in Fig. 1, where the layers are functionally described as: the input layer (L_1), the fuzzy partition layer (L_2), the firing strength layer (L_3), the normalized firing strength layer (L_4) and the output layer (L_5). The first four layers perform the fuzzy partition of the input space and construct the antecedent part while the last layer together with the weights and the results obtained by the partition carry out the implementation task of control and learning. This structure can update membership function and rule base parameters according to the gradient descent update procedure.

2.1 Fuzzy Elements of the Neuro-fuzzy Logic Controller

Since a Multi-Input-Multi-Output (MIMO) system can always be separated into group of a Multi-Input-Single-Output (MISO) systems, we only consider a multi-input (error and change in error)-single output (control action) neuro-fuzzy logic controller here.

Fig.2 shows the basic structure of a traditional FLC consisting of four major blocks [1],[12],[18]. These blocks can be described as: The *fuzzification interface* is a mapping from the observed non-fuzzy input space $U \subset \Re^n$ to the fuzzy set defined in U. The fuzzy set defined in U is characterized by a membership function $\mu_F : U \rightarrow [0,1]$, and is labelled by a linguistic term F such as "*big*" and "*small*". The *fuzzy rule base* is a set of the neuro-fuzzy controller rules in the form:
R_j: *If* ($x_1(t)$ is A_{1j}) and ($x_2(t)$ is A_{2j}) *Then* ($u(t)$ is B_j), For $j = 1, \ldots, N$, and, $t = 1,2,\ldots$
Where N is the number of rules, $x_1(t)$ and $x_2(t)$ are the input variables to the NFLC at time t given as [15]:

$$x_1(t) = y_r(t) - y_p(t) \tag{1}$$

is the error between the reference signal and the actual system output, and

$$x_2(t) = \frac{x_1(t+1) - x_1(t)}{\Delta T} \tag{2}$$

is the change in error. ΔT is the sampling period, A_{ij} and B_j are linguistic terms characterized by the membership functions $\mu_{A_{ij}}(x_i(t))$ and $\mu_{B_j}(u(t))$ respectively. Throughout this study, the A_{ij} uses the Gaussian shaped membership function, defined by [10],[17]:

$$\mu_{A_{ij}}(x_i(t)) = \mathbf{exp}\left\{-\left(\frac{(x_i(t) - c_{ij})}{\sigma_{ij}}\right)^2\right\} \tag{3}$$

The *fuzzy inference machine* is a decision making logic which employs fuzzy rules from the fuzzy rule base to determine the fuzzy outputs corresponding to its fuzzified inputs. Using the centroid defuzzification method, the *defuzzification interface* determines the non-fuzzy output of the neuro-fuzzy controller in the form [3]:

$$u(t) = \frac{\sum_{j=1}^{N} \bar{y}_j \left(\mu_{A_{1j}}(x_1(t)) \, \mu_{A_{2j}}(x_2(t)) \right)}{\sum_{j=1}^{N} \left(\mu_{A_{1j}}(x_1(t)) \, \mu_{A_{2j}}(x_2(t)) \right)} \tag{4}$$

In the previous equation, x_i is the input of node i, A_{ij} is linguistic label associated with this node, c_{ij} and σ_{ij} are adiustable real valued parameters representing the centre and width of the the Gaussian-shaped function; $\bar{y}_j$ is the point in the output space at which μ_{B_j} achieve its maximum.

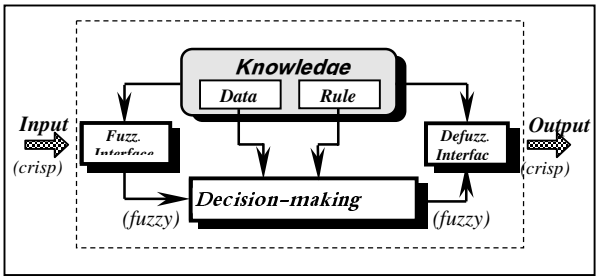

Fig. 2. General Structure of Fuzzy Systems

2.2 Neuro-emulator Model

In this paper, we consider the dynamic system goverened by the following relationship [2],[4]:

$$y(t) = f\{y(t-1),...,y(t-n_y),u(t),...,u(t-n_u)\} \tag{5}$$

where, $y(t)$ and $u(t)$ are the system output and input repectively; n_y, n_u are the corresponding lags in the output and input, $f(\bullet)$ is a nonlinear function. The task is to approximate and generalize $f(\bullet)$ using the multi-layer neuro-emulator with,

$$X(t) = [y(t-1) \ldots y(t-n_y) \; u(t) u(t-n_u)]^T \tag{6}$$

We may formulate the problem using the multi-layer neuro-emulator with the input-output response

$$\hat{y}(t) = \hat{f}(X(t)) \tag{7}$$

Training is done by using EBP algorithm whose MISO relationship forward stage is

$$net_i = \sum_{j=1}^{n_1} w_{ij}^E x_j + b_i \tag{8}$$

and, $$o_i(t) = g(net_i) \tag{9}$$

where, the superscript E stands for the neuro-emulator; w_{ij}^E and b_i are the hidden-input node connection weights and the threshold respectively, for $j = 1,\ldots,n_1$ and $i =$

1,…,n_2; n_1 is the number of nodes in the input layer and n_2 is the number nodes in the hidden layer. Furthermore, $g(\bullet)$ is the sigmoid activation function described as [4]:

$$g(\bullet) = \frac{1}{1+\exp(-(\bullet))} \tag{10}$$

$$y_E(t) = \sum_{j=1}^{n_2} w_{jk}^E \; o_j(t) \tag{11}$$

where, w_{jk}^E are the output-hidden node weights. The error function $e^E(t)$ utilized during the learning period can be defined as

$$e^E(t) = \frac{1}{2}(y_p(t) - y_E(t))^2 \tag{12}$$

In order to minimize the above error function, the weight variations Δw_{ij}^E and Δw_{jk}^E of the neuro-emulator can be determined as follows

$$w_{ij}^E(t+1) = w_{ij}^E(t) + \Delta w_{ij}^E(t) \tag{13}$$

and, $$w_{jk}^E(t+1) = w_{jk}^E(t) + \Delta w_{jk}^E(t) \tag{14}$$

with, $$\Delta w_{ij}^E(t) = -\eta \frac{\partial e^E(t)}{\partial w_{ij}^E(t)} = -\eta \delta_j^E x_j(t) \tag{15}$$

and, $$\Delta w_{jk}^E(t) = -\eta \frac{\partial e^E(t)}{\partial w_{jk}^E(t)} = -\eta \, \delta_k^E o_j^E(t) \tag{16}$$

where, $$\delta_k^E = y_p(t) - y_E(t) \tag{17}$$

and, $$\delta_j^E = g'(X(t)) \sum_j \delta_k^E w_{jk}^E \tag{18}$$

where, $g'(\bullet)$ denotes the activation function derivation. After the NE has been trained to emulate the plant exactly, the plant output $y_p(t)$ is replaced with the NE output $y_E(t)$. Then the error signal δ^c of the NFLC can be obtained as follows:

$$\delta_k'^E = y_r(t) - y_E(t) \tag{19}$$

$$\delta_j'^E = g'(X(t)) \sum_j \delta_k'^E w_{jk}^E \tag{20}$$

Thus the performance error at the output of the NFLC can be obtained as [2],[14]:

$$\delta^c = \sum_j \delta_j'^E w_{jk}^E \tag{21}$$

where, the superscript *c* stands for the neuro-fuzzy controller.

3 Adaptive Neuro-fuzzy Logic Control System

When there exist some variations in the internal or external environments of the controlled plant, it will be required for the controller to possess the adaptive ability to deal with such changes. Thus, in this section, an adaptive neuro-fuzzy logic control system (ANFLC) will be developed by using the NFLC described earlier in section 2.

But, when applying the previous NFLC, there are some difficulties in obtaining the performance error signal. To overcome this problem we use the neuro-emulator (NE) presented in the previous section, which can emulate the plant dynamics and backpropagate, the errors between the actual and desired outputs through the NE. Fig. 3 shows the proposed scheme constructed by the two input-single output NFLC and the NE, where k_1, k_2 and k_3 are the scaling gains for x_1, x_2 and u respectively.

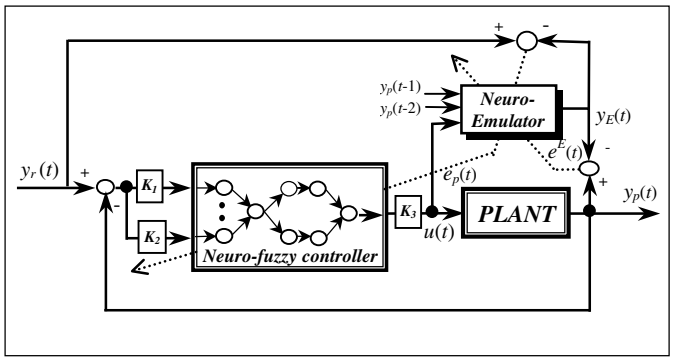

Fig. 3. The Proposed Adaptive Neuro-fuzzy Logic Controller Scheme

3.1 Learning Mechanism

At each time step we adjust the parameters of the NE before updating the controller. For this purpose, the EBP training algorithm is used to minimize the performance error e_p as follows:

$$e_p(t) = \frac{1}{2}\{(y_r(t) - y_E(t))^2\} \tag{21}$$

From equation (4), we train $\bar{y}_j$ as follows [6],[9]:

$$\bar{y}_j(t) = \bar{y}_j(t-1) + \Delta\bar{y}_j(t) \tag{22}$$

$$\Delta\bar{y}_j(t) = -\eta \frac{\partial e_p(t)}{\partial c_{ij}(t)} + \alpha\, \Delta\bar{y}_j(t-1) \tag{23}$$

where, η is the learning rate and α is the constant momentum term. Using the chain rule, we have:

$$\frac{\partial e_p(t)}{\partial \bar{y}_j(t)} = e_p(t) \frac{\mu_j}{\sum_j^N \mu_j} \sum_{i=1}^{n_2} w_{ij}^c\, w_{jk}^c\, g'(net_i) \tag{24}$$

Thus the training algorithm for $\bar{y}_j$

$$\bar{y}_j(t) = \bar{y}_j(t-1) - \eta\,[e_p(t) \frac{\mu_j}{\sum_j^N \mu_j} \sum_{i=1}^{n_2} w_{ij}^c w_{jk}^c\, g'(net_i)] + \alpha\, \Delta\bar{y}_j(t-1) \tag{25}$$

In order to train c_{ij} in (3) and (4), we use

$$c_{ij}(t) = c_{ij}(t-1) + \Delta c_{ij}(t) \tag{26}$$

$$\Delta c_{ij}(t) = -\eta \frac{\partial e_p(t)}{\partial c_{ij}(t)} + \alpha \, \Delta c_{ij}(t-1) \tag{27}$$

Again, using the chain rule, we have

$$\frac{\partial e_p(t)}{\partial c_{ij}(t)} = e_p(t) \sum_{i=1}^{n_2} w_{ij}^c \; w_{jk}^c \; g'(net_i) \left(\frac{\bar{y}_j(t) - u(t)}{\sum_{j=1}^{N} \mu_j} \right) \left(\frac{x_i(t) - c_{ij}(t)}{(\sigma_{ij})^2} \right) \tag{28}$$

and the training algorithm for c_{ij} is

$$c_{ij}(t) = c_{ij}(t-1) - \eta \left[e_p(t) \sum_{i=1}^{n_2} w_{ij}^c \; w_{jk}^c \; g'(net_i) \left(\frac{\bar{y}_j(t) - u(t)}{\sum_{j=1}^{N} \mu_j} \right) \left(\frac{x_i(t) - c_{ij}(t)}{(\sigma_{ij})^2} \right) \right] + \alpha \, \Delta c_{ij}(t-1) \tag{29}$$

By using the same above methods, we train σ_{ij} in (3) and (4) as:

$$\sigma_{ij}(t) = \sigma_{ij}(t-1) + \Delta \sigma_{ij}(t) \tag{30}$$

Thus the training algorithm for σ_{ij} is

$$\sigma_{ij}(t) = \sigma_{ij}(t-1) - \eta \left[e_p(t) \sum_{i=1}^{n_2} w_{ij}^c \; w_{jk}^c \; g'(net_i) \left(\frac{\bar{y}_j(t) - u(t)}{\sum_{j=1}^{N} \mu_j} \right) \left(\frac{(x_i(t) - c_{ij}(t))^2}{(\sigma_{ij})^3} \right) \right] + \alpha \, \Delta \sigma_{ij}(t-1) \tag{31}$$

4 Simulations

In this work, the proposed model is examined using two different applications. Firstly, the well known example of a nonlinear dynamic system given in [6],[19] was used to test for the predicted plant output. This is governed by the following difference equation:

$$y(t+1) = a_1 \; y(t) + a_2 \; y(t-1) + f[u(t)] \; ;$$

The nonlinear function has the form:

$f(u) = 0.6 \sin(\pi u) + 0.3 \sin(3\pi u) + 0.1 \sin(5\pi u)$; In order to predict the plant outputs, the following difference equation was used:

$$\hat{y}(k+1) = a_1 \; y(t) + a_2 \; y(t-1) + \hat{f}[u(t)] \; ;$$

where a_1 = 0.3 and a_2 = 0.6. For the prediction stage we select η = 0.01, n_1 = 3, n_2 = 12 , and $\alpha = 0.7$. The parameters w_{ij} and w_{kj} are initialized randomly and uniformly distributed over [-0.5, 0.5]. Training data of 500 samples are generated from the plant model and used to train w_{ij} and w_{kj} in order to minimize the performance error at each time interval. Fig.4(a) shows that the model is

approximated and converged to the plant output after only a few iterations. For the control stage, after the learning process is finished, the model is tested utilizing the same intilized parameters and the trained weights were being used to train NFLC in (25), (29) and (31) using (21). Fig.4(b) shows the improved results obtained using the combined scheme. Moreover, the adaptive capabilities and generalization abilities of the proposed scheme are further investigated by changing the input functions to: $f(u) = 0.3\cos(u(t)) + 0.25\,\mathbf{sin}(u(t))$ for $250 \le t \le 325$ and $f(u) = 0.45\,\mathbf{sin}(u(t))$ for $325 < t \le 500$, where $u(t) = \mathbf{sin}(2\pi t/100)$, the results are also shown in Fig. 4(b). the ANFLC model has a good match with the actual model with E_p = 0.02638, obviously, the proposed scheme can commendably identify the plant and assured its tracking performance.

Secondly, for further invetigations, the scheme was tested on a highly nonlinear system used in [20]:

$$y(t) = 0.8\,\mathbf{sin}(y(t-1)) + \frac{0.4\,y(t-1)\,\mathbf{sin}(y(t-2))}{1+y^2(t-2)} + 0.5u(t),$$

with the desired input sequence signal chosen as $\mathbf{[sin}(t/20) + \mathbf{sin}(t/50)\mathbf{]}$.

After extensive testing and simulations, the ANFLC model proved a good performance in forecasting the output of the complex-dynamic plant, it has a good match with the actual model where the performance error minimized from E_p = 0.355 to E_p = 0.00138, The results of the prediction and control stages of this nonlinear system are presented in Fig. 5(a) and (b), respectively. Comparable performance to the first plant were obtained.

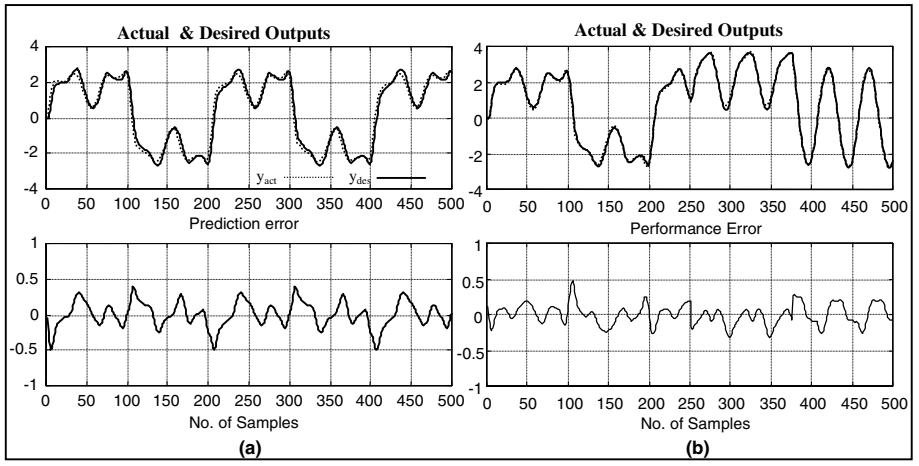

Fig. 4. Actual and Desired Outputs for 1st model: (a) For Prediction Stage: ➔ E_p = 0.2310; (b) For Control Stage: ➔ E_p = 2.638e-02

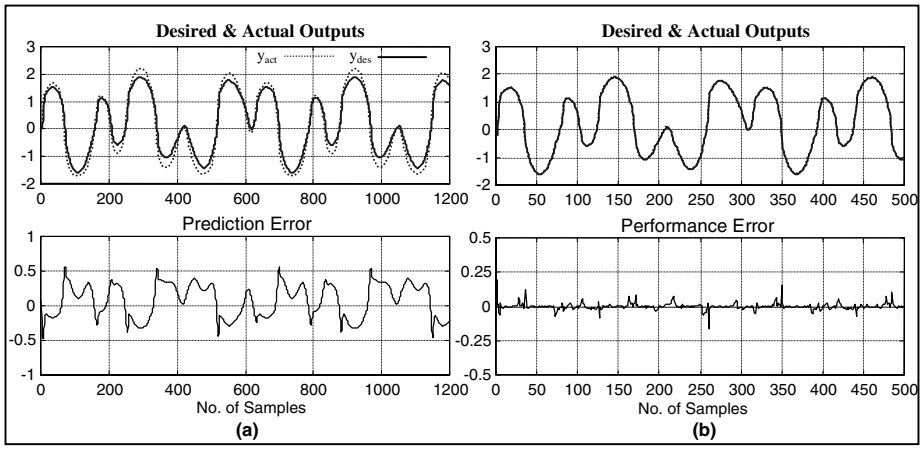

Fig. 5. Actual and Desired Outputs for 2nd model: (a) For Prediction Stage: ➔ E_p = 0.355; (b) For Control Stage: ➔ E_p = 1.38e-03

5 Conclusions

A flexible, adaptive neuro-fuzzy controller scheme (ANFLC) using the integration of FLC and ANNs has been proposed in this paper. The main features of the proposed control is that it does not require a reference model and it can be used to identify the unknown dynamics of the plant. The membership functions in the antecedent part and the real numbers in the consequent part of the NFLC are optimized by this method. The main advantages of the proposed model over FLC & ANNs are:

- A neural net mapping, of "Black Box" type, which is difficult to interpreted , is avoided.
- The tuning problem of fuzzy controllers is eliminated.

Two nonlinear examples are treated to demonstrate the potential applicability and usefulness of this approach in nonlinear dynamic processes.

References

1. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I & II ", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol.20, No.2, 1990, pp. 404-418.
2. V. Kasparian and C. Batur, "Model Reference Based Neural Network Adaptive Controller", *ISA Trans.*, Vol.37, 1998, pp. 21-39.
3. D. Psaltis; A. Sideris; and A. Yamamura, "A Multi-layered Neural Network Controller", *IEEE Control Systems Magazine*, 1988, pp. 17-21.
4. J. Tanomaru and S. Omatu, "Process Control by On-Line Trained Neural Controllers", *IEEE Transactions on Industrial Electronics*, Vol.39, No.6, 1992, pp. 511-521.

5. L-X. Wang, "*Adaptive Fuzzy Systems and Control: Design and Stability Analysis"*, Prentice Hall, NJ, 1994.
6. L-X. Wang and J. Mendel, "Back-propagation fuzzy system as nonlinear dynamic system identifiers", *IEEE International Conference on Fuzzy System.* 1992, pp. 1409 –1418.
7. C. Lin and C. Lee, "Neural-network-based fuzzy logic control and decision system"*, IEEE Trans. Computers,* vol. 40, no. 12, 1991, pp. 1320-1336,
8. C-T. Lin; C-F Juang and C-P Li, "Water bath temperature control with a neural fuzzy inference network", *Fuzzy Sets And Systems,* Vol.111, Issue 2. 2000, pp. 285-306.
9. D. Kaur and B. Lin, "On the Design of Neural-Fuzzy Control System", *Inter. Journal of Intelligent Systems*, Vol.13, 1998, pp.11-26.
10. J. Kim, and N. Kasabov, "HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems"*, Neural Networks,* Vol. 12, Issue 9, 1999, pp. 1301-1319.
11. P. Dash; S. Panda; T. Lee; J. Xu and A. Routray, "Fuzzy and Neural Controllers for Dynamic Systems: an Overview", *IEEE Proceedings.* 1997, pp. 810-816.
12. R-J Jang and T. Sun, "Neuro-Fuzzy Modeling and Control", *Proceeding of the IEEE*, Vol.83, No.3, 1995, pp. 378-404.
13. S. Horikawa; T. Furahashi; and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the backpropagation algorithm", *IEEE Trans. on Neural Networks,* vol.3, 1992, pp. 801-806.
14. T. Hasegawa; S-I Horikawa; T. Furahashi and Y. Uchikawa, "An Application of Fuzzy Neural Networks to Design Adaptive Fuzzy Controllers", *Proceedings of International joint Conference on Neural Networks,* 1993, pp.1761-1764.
15. Y. Hayashi; E. Czogala and J. Buckley, "Fuzzy neural controller"*, IEEE Intelligent. Conf. on Fuzzy Systems*, 1992, pp. 197-202.
16. Y. Shi & M. Mizumoto, "Some considerations on conventional neuro-fuzzy learning algorithms by gradient descent method", *Fuzzy Sets and Systems.* Vol.112, Issue 1, 2000, pp. 51-63.
17. Y. Wang and G. Rong, "A Self-Organizing Neural Network Based Fuzzy System", *Fuzzy Sets and Systems*, No.103, 1999, pp. 1-11.
18. P. Lindskog, "*Fuzzy Identification from a Grey Box Modeling Point of View"*, Technical Report, Department of Electrical Engineering, Linkoping University, Linkoping, Sweden, 1996.
19. K. S. Narendra and K. Parthsarathy, "Identification and control of dynamical systems using neural networks", *IEEE Trans. on Neural Networks,* Vol.1. No.1, 1990, pp. 4-27.
20. L. Pavel and M. Chelaru "Neural Fuzzy Architecture for Adaptive Control", *Proceedings of IEEE*, pp. 1115-1122. 1992.