A STUDY ON FUZZY-NEURAL CONTROL OF NONLINEAR SYSTEM

°Jae-Chul Oh†, Jin-Hwan Kim†, Uk-Youl Huh†

†Dep.of Electrical Eng., INHA University, INCHON, KOREA, E-mail:uyhuh@dragon.inha.ac.kr.

Abstracts - This paper proposes identification and control algorithm of nonlinear systems and the proposed fuzzy-neural network has following characteristics. The network is roughly divided into premise and consequence. The consequence function is nonlinear function which consists of three parameters and the membership function in the premise contains of two parameters. The parameters in premise and consequence are learned by the extended back-propagation algorithm which has a modified form of the generalized delta rule. Simulation results on the identification show that this method is more effective than that of Narendra [3]. The indirect fuzzy-neural control is made of the fuzzy-neural identification and controller. Result on the indirect fuzzy-neural control shows that the proposed fuzzy-neural network can be efficiently applied to nonlinear systems.

Keywords Fuzzy-Neural network, Membership function, Premise, Consequence, Nonlinear system

1. INTRODUCTION

In recent years, there have been a number of ways of identification and control with fuzzy-neural network [1], [2]. Fuzzy relationship models are expressed by a set of fuzzy linguistic propositions which is derived from the experience of the skilled operators or a group of observed input-output data. Fuzzy model identification which is based on the fuzzy set theory proposed by Zadeh has been developed and widely investigated.

However, for some large complex systems, it is almost impossible to establish such a fuzzy relationship model due to the large amount of the fuzzy propositions and the highly complicated multidimensional fuzzy relationships.

Artificial Neural Network has been developed because of the strong nonlinear mapping and learning abilities. Use of neural networks for adjustment of fuzzy membership functions and modification of fuzzy rules makes it be practical to design adaptive fuzzy models and self-organizing fuzzy controllers.

In this paper, nonlinear systems are identified by using fuzzy and multilayed neural network. The neural network contains not only summing and product neurons but also fuzzy neurons that perform minimum operation which selects the minimum value of the membership grades for all premise variables. In order to adjust the consequence and premise parameters, the error back propagation algorithm is extended.

Before control is trained on-line, the identification of system is required to be trained off-line in the indirect fuzzy-neural control. The pretrained emulator is crucial to the performance of the indirect fuzzy-neural control.

The paper is organized as follows: In section 2, the proposed fuzzy-neural network structure is described. section 3 introduces control schemes about direct and indirect fuzzy-neural control. Section 4 deals with method for adjustment of parameters of fuzzy-neural network. Section 5 shows about Identification and indirect fuzzy-neural control. Finally, section 6 represents conclusions and research for the future.

2. FUZZY-NEURAL NETWORK SYSTEM

The proposed fuzzy-neural network architecture is shown in Fig.1. In Fig. 1, the network is divided into premise and consequence. The neural networks contain summing, multiplying neurons and fuzzy neurons, which perform fuzzy operations such as the minimum or maximum operation. Σ represents the summing neurons, Π means the products neurons and Λ describes the fuzzy neurons which are minimum operators.

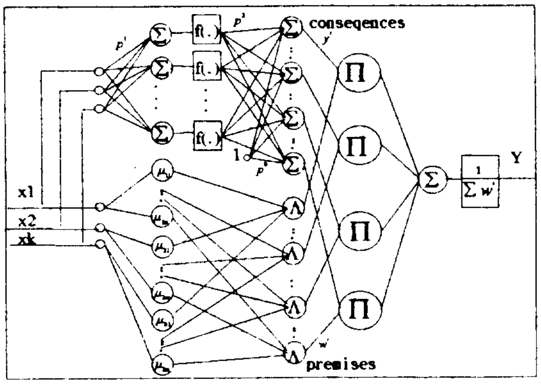

Fig. 1 Fuzzy-Neural Network Architecture

In order to update the parameters in premise and consequence, error back propagation algorithm is applied to nonlinear system. The final output of fuzzy model is expressed by (1).

$$Y=\sum_{i=1}^{m} w^i y^i / \sum_{i=1}^{m} w^i \ , w^i = \wedge^{k}{}_{l=1}\, \mu_{li}(x_l) \tag{1}$$

$$y^i = P^0{}_i + \sum_{j=1}^{n}(P^2{}_{ji} \cdot f(\sum_{l=1}^{k} P^1{}_{lj} \cdot x_l)) \tag{2}$$

$$f(x) = \frac{1}{1+\exp(-x)} \tag{3}$$

$$\mu_{li}(x_l) = \exp(-(x_l - a^i{}_l)^2 / \ b^i{}_l{}^2) \tag{4}$$

where, w^i is the overall truth value for the premises of the i-th implication and $\mu_{li}(x_l)$ is the membership function of the fuzzy subset which is supposed to Gaussian function.

Assume that the consequence sub-network has n hidden neurons, then the output of consequence is made by (2) and (3) where, $f(\cdot)$ is a sigmoid type nonlinear function.

If input variables are large numbers, values of sigmoid and membership function become 0 or 1. Also, output of fuzzy-neural identifier is saturation which means its output don't follow desired output after the training is stopped. Therefore, input variables must be normalized. The truth value of the premise for each rule is determined by the minimum value of the membership grades for all premise variables and it will be updated indirectly by changing the form of the membership function. The membership function is used to gaussian type.

3. FUZZY-NEURAL CONTROL

3.1 Direct Fuzzy-Neural Control

Structure of direct fuzzy-neural control is depicted as Fig. 2 .

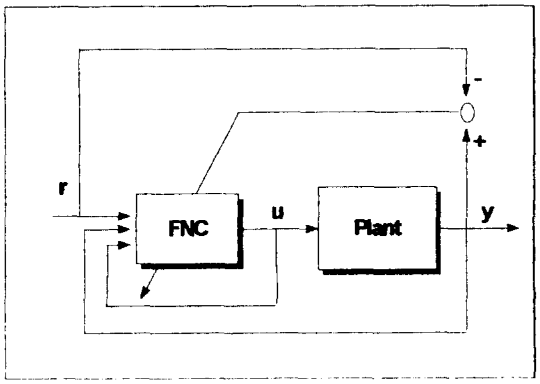

Fig. 2 Direct Fuzzy-Neural Control Structure

The square error function E_r is defined as

$$E_r = \frac{1}{2}(r-y)^2$$

The problem of direct fuzzy-neural control is that the exact calculation of δ_k (sensitivity of E_r w. r. t. the output of the FNC) requires knowledge of the Jacobian of the plant.

δ_k term can be represented as

$$\delta_k = \varepsilon_k e(k) \frac{dy(k)}{du(k)}$$

$$\hat{e} = e(k)\, dy(k)/du(k)$$

$$\varepsilon_k = \begin{cases} 0 \ , \text{if} \begin{cases} \hat{e}(k) > 0 \ \text{ and } \ u(k-1) = u_M \\ \text{or} \\ \hat{e}(k) < 0 \ \text{ and } \ u(k-1) = u_m \end{cases} \\ 1 \ , \textit{otherwise}. \end{cases}$$

$$-\frac{\partial E_r}{\partial w} = \varepsilon_k \cdot e(k) \cdot sgn[\, dy(k)/du(k)] \cdot \frac{\partial u}{\partial w} \tag{5}$$

where e(k)=r(k)-y(k) and the binary factor ε_k is introduced to account for the constraint on the input u(k). The role of ε_k is to prevent the FNC(Fuzzy-Neural Controller) from mistraining to the references that cannot be tracked. The inclusion of ε_k is equivalent to considering the existence of a limiter between the output of the FNC and the input of the plant. Parameters on direct FNC is approximately updated by eqn. (5) and w is weights of FNC.

This control scheme mainly suffers from the problem of estimating the derivation of the model output with respect to the control input signal, i. e. , the sensitivity of the unknown plant ($\partial y / \partial u$) can not be back propagated to the feed forward inverse dynamic fuzzy-neural controller. So, this term can be replaced by sgn(dy/du).

3.2 Indirect Fuzzy-Neural Control

Scheme of the indirect fuzzy-neural control is shown in Fig.3.

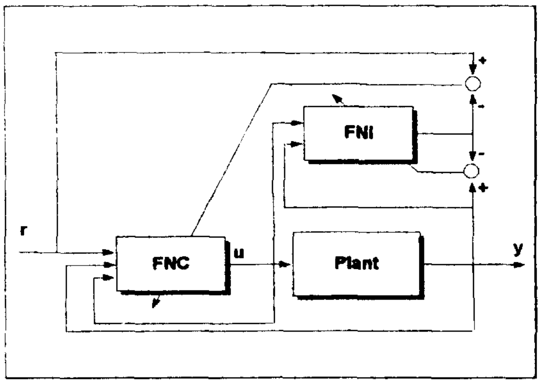

Fig. 3 Indirect Fuzzy-Neural Controller

The square error function E_r between reference and model error function is defined as

$$E_r = \frac{1}{2}(r-Y)^2$$

FNI (Fuzzy-Neural Identification) is used to compute the sensitivity of the error function E_r w. r. t. the controller's output. The desired sensitivity output is calculated by using BP.

The difference between direct FNC and indirect FNC is that y(output of the plant) is changed into Y (output of FNI) in eqn. 5. FNI should be off-line trained with a data set sufficiently rich to allow plant identification, and then both FNC and FNI are on-line trained.

4. LEARNING ALGORITHM

Supposed that the actual plant output is y and output of fuzzy neural network is Y, we define the error function as

$$E_y = \frac{1}{2}(y - Y)^2 \qquad (6)$$

To adjust the consequence and premise parameters, the error back propagation algorithm should be slightly extended. The consequence and premise parameters are updated as

$$\begin{aligned} w_{ij}(n+1) &= w_{ij}(n) - \eta \cdot \frac{\partial E_y}{\partial w_{ij}(n)} + \alpha \cdot \Delta w_{ij}(n-1) \\ &= w_{ij}(n) + \Delta w_{ij}(n) \end{aligned} \qquad (7)$$

where η is the learning rate, α is the momentum factor, and n indicates the number of training iterations.

Using the chain rule, in eqn.(7), gradients of the required E_y w. r. t. consequence(w_{con}) and premise(w_{pre}) parameters are determined by

$$\begin{aligned} \frac{\partial E_y}{\partial w_{con}} &= \frac{\partial E_y}{\partial Y} \cdot \frac{\partial Y}{\partial y^i} \cdot \frac{\partial y^i}{\partial w_{con}} \\ \frac{\partial E_y}{\partial w_{pre}} &= \frac{\partial E_y}{\partial Y} \cdot \frac{\partial Y}{\partial w^i} \cdot \frac{\partial w^i}{\partial w_{pre}} \end{aligned} \qquad (8)$$

Then, we obtain the training algorithm for adjusting the consequence parameters (ΔP^0_i ,ΔP^1_{lj} ,ΔP^2_{ji}) as the fol lowing form

$$\begin{aligned} \Delta P^0{}_i(n) &= -\eta \frac{\partial E_y}{\partial P^0{}_i} + \alpha 1 \Delta P^0{}_i(n-1) \\ &= \eta\ \delta^i{}_1 + \alpha 1 \Delta P^0{}_i(n-1) \\ \Delta P^1{}_{lj}(n) &= -\eta \frac{\partial E_y}{\partial P^1{}_{lj}} + \alpha 1\ \Delta P^1{}_{lj}(n-1) \\ &= \eta \sum_{i=1}^{m} \delta^i{}_1\ P^2{}_{ji} f'(\sum_{l=1}^{k} \Delta P^1{}_{lj} \cdot x_l) \cdot x_l + \alpha 1\ \Delta P^1{}_{lj}(n-1) \\ \Delta P^2{}_{ji}(n) &= -\eta \frac{\partial E_y}{\partial P^2{}_{ji}} + \alpha 1 \Delta P^2{}_{ji}(n-1) \\ &= \eta\ \delta^i{}_1 f(\sum_{l=1}^{k} P^1{}_{lj} \cdot x_l) + \alpha 1 \Delta P^2{}_{ji}(n-1) \end{aligned} \qquad (9)$$

where $\delta^i{}_1$ is $(y-Y)w^i / \sum_{i=1}^{m} w^i$, η is learning rate, m is the number of rule, k is the number of input, $f'(\cdot)$ is f(1-f) and $\alpha 1$ is momentum coefficient.

Also, the parameters of the membership function(Δa^i_l , Δb^i_l) are modified in the following form

$$\begin{aligned} \Delta a^i{}_l(n) &= -\beta \frac{\partial E_y}{\partial a^i{}_l} + \alpha 2 \Delta a^i{}_l(n-1) \\ &= \beta\ \delta^i{}_2 2(x_l - a^i{}_l) /\ b^i{}_l{}^2 + \alpha 2 \Delta a^i{}_l(n-1) \\ \Delta b^i{}_l(n) &= -\beta \frac{\partial E_y}{\partial b^i{}_l} + \alpha 2 \Delta b^i{}_l(n-1) \\ &= \beta\ \delta^i{}_2 2\ (x_l - a^i{}_l)^2 /\ b^i{}_l{}^3 + \alpha 2 \Delta b^i{}_l(n-1) \end{aligned} \qquad (10)$$

where β is learning rate ,$\alpha 2$ is momentum coefficient and δ^i_2 is $(y-Y) / \sum_{i=1}^{m} w^i\ (y^i - Y)$.

5. SIMULATIONS

Example 1: Plant to be identified is governed by

y(k+1)=0.3y(k)+0.6y(k-1)+f[u(k)]

where the unknown function has the form f[u(k)]=0.6sin(πu(k))+0.3sin(3πu(k))+0.1sin(5πu(k)). Input vectors [y(k-1),y(k-2),y(k-3),y(k-4),u(k-1),u(k-2)], and rule size is 40, η is 0.05 and α is 0.8. In Fig. 4, the identification model follows the plant more and more accurately as more and more training is performed. When the adaptation process is stopped at k=500, we can see the identification model approximately follows the plant. In Narendra and Parthasarathy[3], the same plant is identified using a neural network identifier which failed to follow the plant when the training is stopped at k=500. It is shown in Fig. 5.

Example 2: Nonlinear plant is described by

$$y(k) = \frac{y(k-1)y(k-2)y(k-3)u(k-2)(y(k-3)-1)+u(k-1)}{1+y^2(k-2)+y^2(k-3)}$$

where u(k)=sin(2πk/250) (1≤k≤500),u(k)=0.8sin(2πk/250) +0.2sin(2πk/25) (k>500), rule size is 20 ,η is 0.01, and α is 0.8. Comparing Fig.6 and Fig. 7,although Narendra's neural identifier was trained for 100,000 steps, its performance is worse than that of fuzzy-neural identifier which was trained for 16,000 steps.

Example 3: Nonlinear plant is described by

$$\begin{aligned} y(k+1) &= \frac{y(k)}{1+y(k)^2} + u(k)^3 \\ r(k+1) &= 0.25 \cdot (\sin(2*3.14*(k+1)/25) \\ &\quad + \sin(2*3.14*(k+1)/10)) \end{aligned}$$

where, r(k) is reference function, control input vectors [y(k),y(k-1),y(k-2),r(k+1), u(k-1)], rule size is 5, η is 0.01 and α is 0.8, rule size is 5.

The output of reference and the actual output of plant was depicted in Fig. 8.

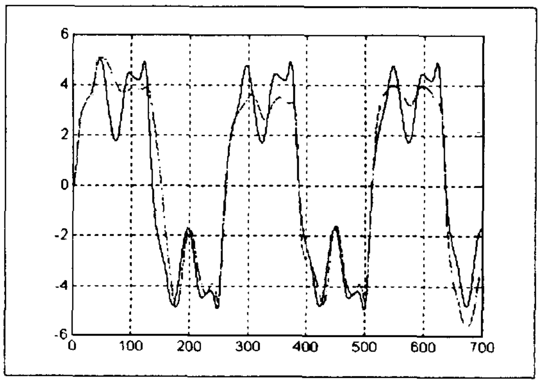

Fig. 4. Fuzzy-Neural identification for Example 1 (solid line: desired output, dashed line: model output)

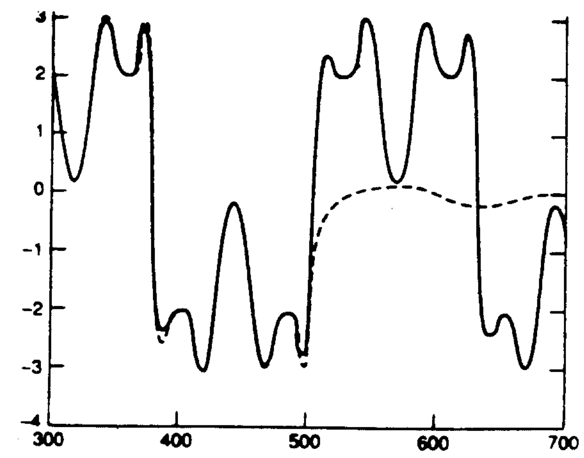

Fig.5.Neural network identification of Narendra when the training is stopped at k=500(solid line: desired output, dashed line: model output)

6. CONCLUSIONS

In this paper, we proposed the fuzzy-neural network which can be largely divided into consequence and premise. In learning algorithm, back propagation algorithm contains momentum term in order to reduce significantly training time. Identifier and controller of the proposed fuzzy-neural are introduced. Simulation results of Identifier show that the proposed fuzzy-neural network is more effectively than that of Narendra [3] and simulation result of the indirect fuzzy-neural control shows that it can be effectively applied to nonlinear systems. For the research of future, we must study method for selecting suitable initialization of weights and normalization of input variable and we must develope the better learning algorithm.

7. REFERENCES

[1] Yaochu Jin, Jingping Jiang, and Jing Zhu,"Neural Network Based Fuzzy Identification and Its Application to identification and Control of Complex Systems," IEEE Trans. on Sys. Man and Cybern.,vol. 25, no, 6 pp. 990-997,1995.

[2] Yinghua Lin and GeorgeA."A New Approach to Fuzzy-Neural System Modeling,"IEEE Trans on Fuzzy Systems, vol. 3, no. 2, pp. 190-198, 1995.

[3] KUMPATIS. NARENDRA AND KANNAN PARTHASARATHY, "Identification and Control of Dynamical Systems Using Neural Networks," IEEE Trans. on Neural network, vol.1, no.1, pp. 4-26,1990.

[4] Yao Zhang , Pratyush Sen, and Grant E. Hearn, "An On-Line Trained Adaptive Neural Controller," IEEE control system, pp. 67-75,1995.

[5] Julio Tanomaru and Sigeru Omatu,"Process Control by On-line Trained Neural Controllers," IEEE Tans. on Industrial Electronics, vol. 39 no. 6, pp. 511-521,1992.

[6] Martin G.Bello, "Enhanced Training Algorithms, and Integrated Training/Architecture Selection for Multilayer Perceptron Networks ,"IEEE Trans. on Neural Networks, vol. 3, no .6, pp. 864-875, 1992

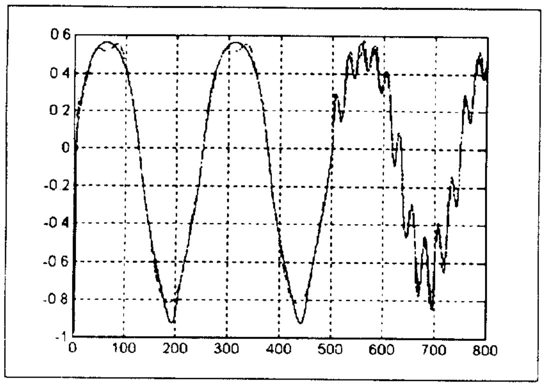

Fig.6.Fuzzy-Neural identification for Example 2 when training stops at k=16,000(solid line: desired output, dashed line: model output)

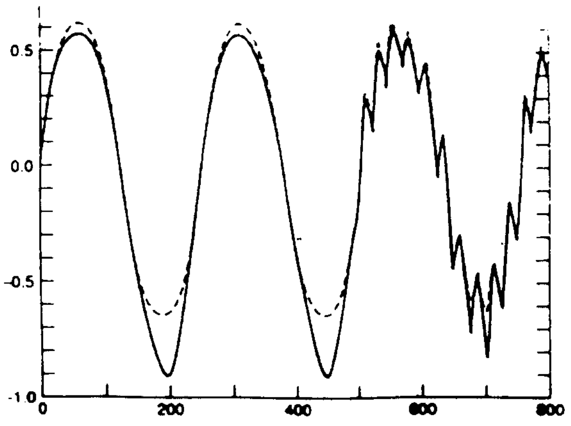

Fig.7. Neural network identification of Narendra when the training is stopped at k=100,000(solid line: desired output, dashed line: model output)

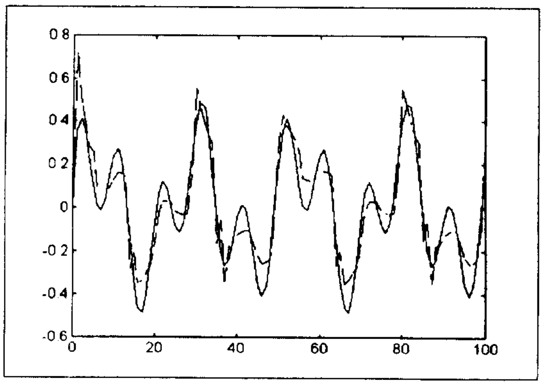

Fig.8. Indirect fuzzy-neural control for Example 3 (solid line: desired output, dashed line: model output)