An Neural Network Direct Controller for Nonlinear Systems

Kee-Hwan Nam*, Cheol-Soo Bae*, Hyeon-Seob Cho**, Sang-Dong Ra ***
*Dept of Electronic Communication Engineering, Kwandong University, Yangyang, Korea
Tel: +82-033-649-7194 Fax : +82-033-649-7959 E-mail:keelight@empal.com
** Dept of Electronic Engineering, Chungwoon University, Hongsung, Korea
Tel: +82-041-630-3242 E-mail : chohs@chungwoon.ac.kr
*** Dept of Computer Engineering, Chosun University, Kwangju, Korea
Tel: +82-062-230-7120 Fax: +82-062-230-7381 E-mail: sdna@mail.chosun.ac.kr

Abstract: In this paper, a direct controller for nonlinear plants using a neural network is presented. The controller is composed of an approximate controller and a neural network auxiliary controller. The approximate controller gives the rough control and the neural network controller gives the complementary signal to further reduce the output tracking error. This method does not put too much restriction on the type of nonlinear plant to be controlled. In this method, a RBF neural network is trained and the system has a stable performance for the inputs it has been trained for. Simulation results show that it is very effective and can realize a satisfactory control of the nonlinear system.
Keywords : RBF neural network, nonlinear system, direct controller

1. INTRODUCTION

The conventional design methods of a control system often require the construction of a mathematical model describing the dynamic behavior of the plant to be controlled. When such a mathematical model is difficult to obtain due to uncertainty or complexity of systems, these conventional techniques based on a mathematical model are not well suited for dealing with. Artifical neural network teachiques have been suggested for identification and control of nonlinear plants for which conventional techniques of control do not give satisfactory performance, such as the accuracy in matching the behavior of the physical system.

A good method of applying neural networks for control must have the following properties:

1. It must be applicable to nonlinear plants, since there are already good methods of control for linear plants.
2. It should not put too much restriction on the type of nonlinearity that it can handle.
3. It is preferable to have an Unsupervised Learning method for the neural network because the desired output form of a system for a given input may be known, but the input form of a plant that produces that desired output is not generally known. Unsupervised Training can avoid identification of the plant or its inverse model, which is generally not easy to obtain.
4. The system should be stable at least for the class of inputs it has been trained for.
5. In most cases open loop performance of a plant can be observed and a approximate controller can be devised for that. It would be desirable if we could put as much knowledge as possible in the design of this controller and only leave the extra fine tuning to the neural network controller.

According to the above requirements, a direct auxiliary controller for nonlinear plants using neural network is presented.

2. CONTROLLER DESIGN

The controller presented here is composed of an approximate controller and a neural network auxiliary controller. The structure of controller is shown in Figure 1. The approximate controller gives the approximate performance and the neural network auxiliary controller is used for the purpose of further fine tuning.

The approximate controller can be a PID or any other conventional controller. It can be designed by using the known dynamics of nonlinear plant. The neural network employed in this scheme is an Radial Basis Function Network (RBFN). It produces the complementary signal to further reduce the error **e** between output **y** and the reference input **r** . The structure of RBFN is showed in Figure 2. It is a network with two layers. A hidden layer of radial basis neurons and an output layer of linear neurons. A common choice for the basis function is a Gaussian given by the equation:

$$G_i(x) = \exp\left(-\frac{\|x - c_i\|^2}{2\sigma^2} \right), i = 1,2,...m \qquad (1)$$

Where C_i represents the center of the basis function and σ denotes its width. The norm $\|\cdot\|$ in equation can be expressed by Euclidean distance. The weights and biases of each neuron in the hidden layer define the position and width of a radial basis function. Each linear output neuron forms a weighted sum of these radial basis functions. With the correct weight and bias values for each layer, and enough hidden neurons, a RBFN can fit any function with any desired accuracy. The advantage of the RBFN is its rapid learning, generality and simplicity. RBFN finds the input to output map using local approximators. It can be trained faster than BP and have none of BP's

training problems such as saturation and local minima.

In the RBFN training stage we first observe the performance of the system with the approximate controller for a certain period of time and measure the range of error between the output of the plant and desired output. Then we divide this error span into certain sections and for each section we perform a perturbation test: We increased the input to the plant by δe whenever total square error between the output of the plant and desired output falls within a specified region. If this change of the input results in a lower value of the total square error, we modify the output weight of the neural network controller to work accordingly. This action is continued for all sections and the whole process is repeated until no modification can reduce the error.

Taking an * overlapping Gaussian activation function for kernel units supposedly provides a smoother response and better generalization. but in our case the amount of interference was so high and we obtained a better performance with non-overlapping regions. Nevertheless, smoothness of the output can be enhanced by dividing the correction for each section by modifying the cost function used for training from $J= e^2$ to $J= (e_{new}^2+k(e_{new}-e_{old})^2)$ for $k<1$.

During the training stage, each time only one kernel unit responds and one weight is adjusted, This results in a shorter training time compared with Multi-layer Perceptron (MLP) type networks. Since weights are adjusted by a small value δ each time, the number of necessary iterations depends on the size of error or the accuracy of the approximate controller.

The point that makes this method different is the way in obtaining the necessary corrections, which is based on the effect of weight perturbation on the total square error for a certain period of time. Perturbing the weights of a network is used in Madaline Rule III (MR III) training for analog implementation of neural networks. Because this method does not need prior knowledge about the transfer characteristics of the computing devices, it is not affected by the effects of neuron to neuron variations. Training the network based on its instantaneous result of the error will cause instability when used in a feedback loop. The performance of the system here is observed for a certain period of time and if any adjustment for any given amount of error increases the total square error, it would not be accepted. This proves the stability of the system for the class of inputs it has been trained for.

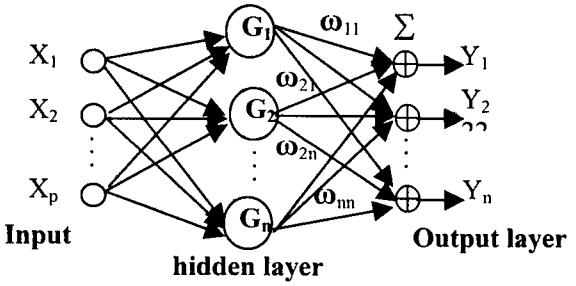

Figure 2. Radial Basis Function Network.

Generally optimization methods based on parameter perturbation are bound to failure when many parameters are involved in perturbation and that is because of the moving target effect of the other parameters. It is not the case in the proposed system for which only one or few related parameters are active at each time. At present, methods for directly adjusting the control parameters based on the output error are not available. The work presented in this paper is a step toward direct adaptive control.

3. SIMULATION STUDY

Different nonlinear systems were considered to evaluate the effect of the proposed method. Figure 3 gives the effect of controller when it is used for three different nonlinear plants. In this figure , y(.) denotes the plant output without controller,y_m(.) denotes the desired output , r(.) denotes the reference input and y_c(.) denotes the output of the plant with controller. An RBF type network with 30 kernel units each sensitive to different ranges of error was used. Input weights and activation function of the units were fixed. Output weights were perturbed and adjusted according to the effect of perturbation on the total square error for 400 sampling time. For a sinusoidal input (r(k)=sin(k/50)) and perturbation amount of δ=0.1, the amount of reduction in aggregated square error for a full cycle of input is example 1:from 3.69 to 1.33,expmple 2:from 105.84 to 26.44 and example 3:from 32.11 to 4.56. To observe the learning and generalization capability of the system, the controller was trained on a step response and its performance was observed on the sinusoidal input. Training the network with different types of input will enchance the generalization capability of the system.

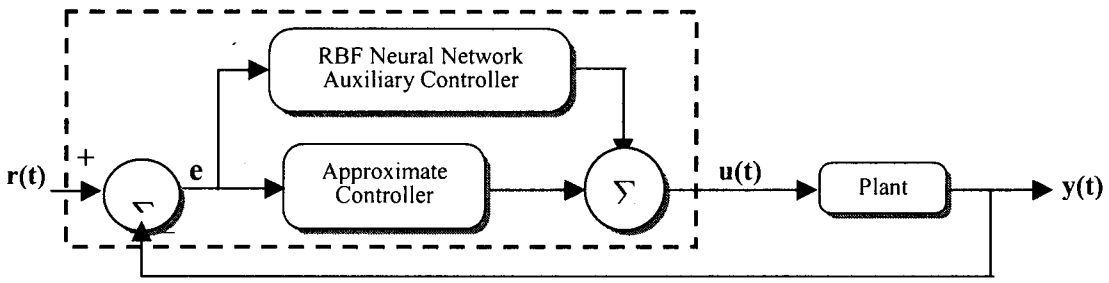

Figure 1. The structure of the controller.

Example 1

$$\begin{cases} y(k) = \dfrac{1.2y(k-1)}{1+y(k-1)^2} + 0.5u(k-1) \\ y_m(k) = 0.3y_m(k-1) + 0.2y_m(k-2) + 0.5r(k-1) \end{cases}$$

Example 2

$$\begin{cases} y(k) = \dfrac{y(k-1)y(k-2)(y(k-1)+2.5)}{1.3+y(k-1)^2+y(k-2)^2} + 1.5u(k-1) \\ y_m(k) = 0.7y_m(k-1) + 0.1y_m(k-2) + r(k-1) \end{cases}$$

based on neural networks.The structure used here can be view as a fuzzy controller implementation for which the control actions or rules which depend on the error between the plant output and the desired output are deduced in training time.It can also be viewed as a gain scheduling adaptive controller which can work for any unknown plant with no attempt to linearize the system at each region.

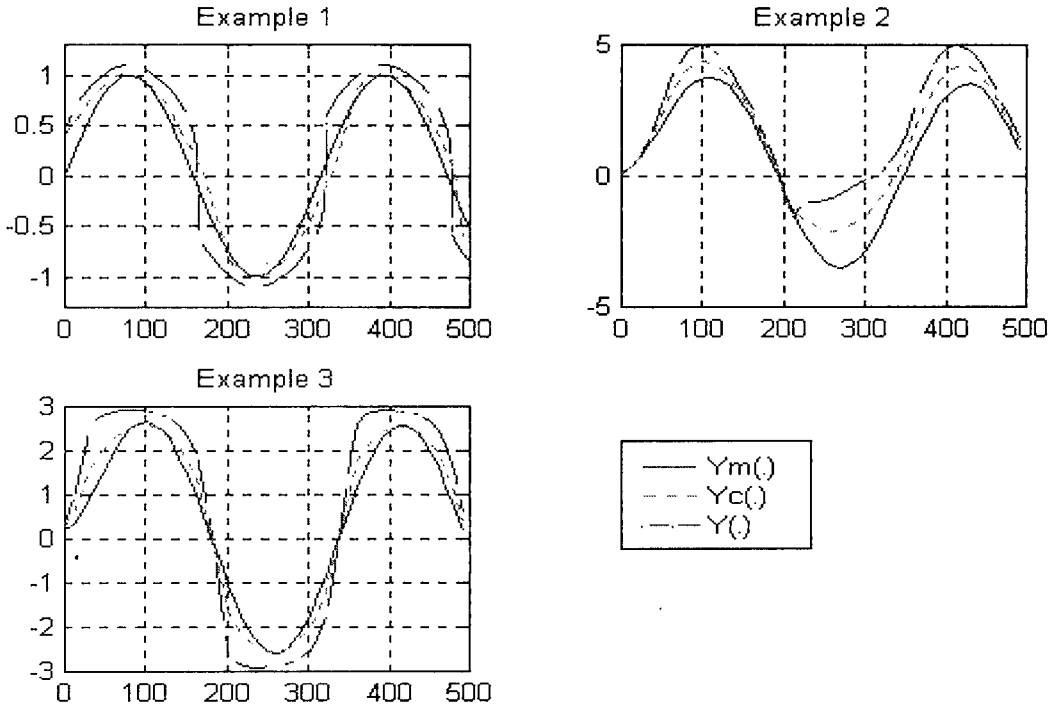

Figure. 3 $Y_m(.)$ is the output of the reference model, Y(.) is the output of system without NNC and $Y_c(.)$ is the output of system without NNC in the example1,2 and 3.

Example 3

$$\begin{cases} y(k) = sat(0.5y(k-1) + 0.2y(k-2) + 2.1u(k-1)) \\ y_m(k) = 0.48y_m(k-1) + 0.2y_m(k-2) + r(k-1) \end{cases}$$

$$(sat(x) \equiv -3 + \frac{6}{1+\exp(-x)})$$

4. CONCLUSIONS

A neural network controller for nonlinear plants is used in combination with an existing conventional controller, which removes the need for a generalized training scheme. The controller is guaranteed to perform stably for the class of inputs that it has been trained for. Using this method of control does not require assumption of a model for the plant and it makes it different from conventional control methodologies. Furthermore, since training of this network does not require backpropagation of error,it makes direct adaptive control possible, a structure beyond the capabilities of backpropagation

References

[1] .Moody and C.J.Darken, "Fast learning in Networks of Locally-Tuned Processing Units" Neural Computation,Vol.1,pp.281-194,1989.

[2] D.Andes,B.Widrow, M.Lehr and E.Wan,"MR II :A Robust Algorithm for Training Analog Neural Networks" International Joint Conference on Neural Networks,1990

[3] K.S.Narendra and K.Parthasarathy,"Identification and Control of Dynamical Systems Using Neural Networks" IEEE Trans on Neural Network Vol.1 No.1,pp.4-27,1990

[4] M.M.Gupta and D.H.Rao,"Dynamic Neural Units in the Control of Linear and Nonlinear Systems. 'In Proceedings of the International Joint Conf. On Neural Networks,June 1992,pp.100-105

[5] G.A.Montague, M.J.Willis and A.J.Morris, "Artificial Neural Network Model Based Control" Automatic Control Conference,1994

[6] Howard Demuth, Mark Beale, "Neural Network ToolBox for Use with MATLAB" The MathWorks Inc.,1994

[7] Chen S,Billing S A, "Grant P M. Recursive Hybrid Algorithm for Nonlinear System Identification using Radial Function Network",Int.J Control, 1992, 55(5): 1051-1070