Computers chem. Engng, Vol. 14, No. 4/5, pp. 573–583, 1990
Printed in Great Britain.

0098-1354/90 $3.00 + 0.00

USE OF NEURAL NETS FOR DYNAMIC MODELING AND CONTROL OF CHEMICAL PROCESS SYSTEMS

N. BHAT and T. J. MCAVOY
Department of Chemical Engineering, University of Maryland, College Park, MD 20742, U.S.A.

(*Received* 18 *October* 1989; *received for publication* 29 *November* 1989)

Abstract—Neural computing is one of the fastest growing areas of artificial intelligence. Neural nets are inherently parallel and they hold great promise because of their ability to "learn" nonlinear relationships. This paper discusses the use of backpropagation neural nets for dynamic modeling and control of chemical process systems. The backpropagation algorithm and its rationale are reviewed. The algorithm is applied to model the dynamic response of pH in a CSTR. Compared to traditional ARMA modeling, the backpropagation technique is shown to be able to pick up more of the nonlinear characteristics of the CSTR. The use of backpropagation models for control, including learning process inverses, is briefly discussed.

INTRODUCTION

Neural computing is one of the fastest growing areas of artificial intelligence. The reason for this growth is that neural nets hold great promise for solving problems that have proven to be extremely difficult for standard digital computers. There are two key differences between neural computers and digital computers. First, neural nets are inherently parallel machines and as a result they can solve problems much faster than a serial digital computer. Secondly, and perhaps more importantly, many neural nets have the ability to "learn". Rather than programming these nets, one presents them with a series of examples. From these examples the net learns the governing relationships involved in the training database. Since nonlinear governing relationships can be handled by neural nets, the nets may offer a cost effective approach to modeling chemical process systems.

Figure 1 presents an example of a typical neural net. The net consists of processing neurons (circles) and information flow channels between the neurons, called interconnects. The boxes are neurons that simply store inputs to the net. Each processing neuron typically has a small amount of local memory and it carries out a local computation which converts inputs to the neuron to outputs. This computation is called the transfer function of the neuron. Transfer functions can be linear or nonlinear and consist of algebraic or differential equations. For the net shown in Fig. 1, there are three layers of neurons: input, hidden and output. In some publications the input layer is not counted and the net in Fig. 1 would be considered a two-layered set. In this paper the input layer is counted.

According to Hecht-Nielsen (1988), in 1987 there were approx. 50 different types of neural nets being studied and/or used in applications. Some of the most common nets are: adaptive resonance (Carpenter and Grossberg, 1987; Grossberg, 1986) nets that form input categories for input data with the coarseness of the categories determined by the value of a selectable parameter; backpropagation (Werbos, 1974; Rumelhart and McClelland, 1986) mapping nets that minimize the mean squared mapping error; bidirectional associative memory (Kosko, 1987a, b), a class of single-stage heteroassociative networks, some capable of learning; counterpropagation (Hecht-Nielsen, 1987a, b) networks that function as a statistically optimal self-organizing lookup table and probability density function analyzer; and neocognitron (Fukushima and Miyake, 1984), a multilayer hierarchial character recognition network. The key part of most neural nets is the learning law which modifies internal net values in response to inputs and the transfer function. Hecht-Nielsen (1988) classified learning laws into seven categories: Grossberg (1982, 1986) competitive learning of weighted average inputs; Hebbian (Steinbuch and Piske, 1963) correlation learning of mutually-coincident inputs; Hinton/Sejnowski/Szu/Desieno (Hinton and Anderson, 1981; Szu, 1986) discovery of the absolute minimum of a performance surface function by means of a noise-based distributed statistical search; Kohonen (1987) development of a set of vectors conforming to a particular probability density function; Kosko/Klopf (Kosko, 1987a) formation of sequences of events in temporal order; Rosenblatt (Minskey and Papert, 1972) adjustment of a perceptron linear discriminant device via a performance grading input from an external teacher; and Widrow (Widrow and Hoff, 1960; Widrow and Stearns, 1985) minimization of a mean-squared error cost function. Finally, neural nets that learn can be categorized by whether the learning is supervised, graded or self-organized (Hecht-Nielsen, 1988).

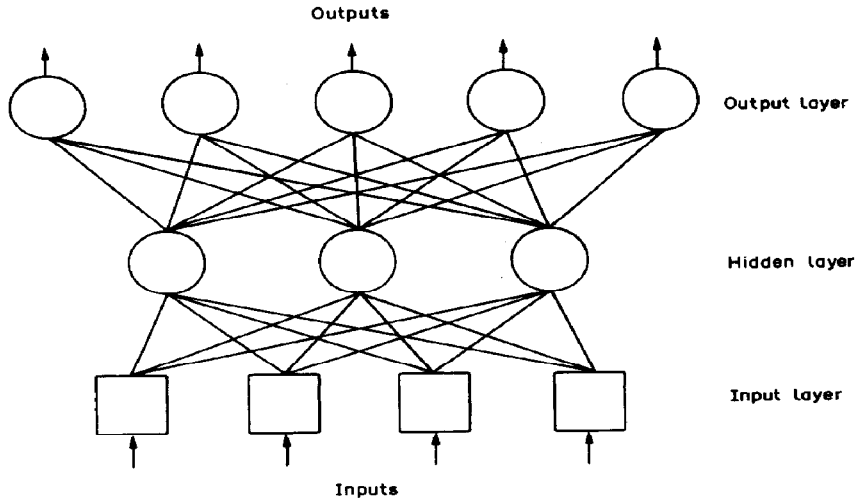

Fig. 1. Backpropagation hidden net.

As this brief overview shows, neural computation is a very broad area and interest in the field is growing rapidly. At present the most widely used neural net is backpropagation (Werbos, 1974; Rumelhart and McClelland, 1986). This paper discusses the use of backpropagation for dynamic modeling and model-based control of chemical process systems. After a discussion of the backpropagation algorithm, a comparison between neural net dynamic modeling and traditional convolution and ARMA modeling is presented. To illustrate how effectively backpropagation nets can learn nonlinear dynamic models, results for a pH stirred tank reactor are given. Finally, two approaches for using neural nets for model-based control are discussed.

BACKPROPAGATION

Overview

A typical backpropagation net is shown in Fig. 1. The following section discusses the backpropagation algorithm in detail. Backpropagation is an example of a mapping neural net that develops an approximation to the function $y = f(x)$ from example x, y pairs. Backpropagation has been applied to a wide variety of practical problems and it has proven very successful in its ability to model nonlinear relationships. Included in the applications are: speech synthesis and recognition (Sejnowski and Rosenberg, 1986; Peeling *et al.*, 1986; Elman and Zipser, 1987); visual pattern recognition (Rumelhart and McClelland, 1986); analysis of sonar signals (Gorman and Sejnowski, 1988); analysis of natural gas markets (Werbos, 1988); bond rating (Dutta and Shekar, 1988); defense applications (Castelaz, 1988); medical diagnosis (Bounds *et al.*, 1988); and learning in control systems (Elsley, 1988). We have used backpropagation successfully in deconvoluting nonlinear light-induced fluorescence spectra to predict the concentrations of fluophores (McAvoy *et al.*, 1989a) and to learn how to design distillation controls from examples (Birky and McAvoy, 1989). Essentially all of the published applications to date have dealt with steady-state functions, $y = f(x)$. In this paper an approach to using backpropagation to model transient relationships is presented.

Backpropagation is a generalization of the steepest descent learning law presented by Widrow and Hoff (1960). These authors analyzed a two-layered neural net, called the ADALINE and they developed a simple, local gradient descent method. This method was generalized to multilayered nets as the backpropagation algorithm (Werbos, 1974; Rumelhart and McClelland, 1986). The issue of the number of layers in a neural net is a very important one. In their classic text, Perceptrons, Minsky and Papert (1972) showed that a two-layered net of the type studied by Widrow and Hoff was limited in the types of problems it could solve. Minsky and Papert speculated that the study of multilayered nets would be a "sterile" area. However, this speculation proved to be incorrect. Multilayered nets can produce results that are impossible to achieve with two-layered nets. As discussed by Rumelhart and McClelland (1986) the addition of a hidden layer allows the backpropagation algorithm to develop an "internal representation" of the problem which can be vital for its solution. It appears that the hidden layer gives backpropagation a whole new dimension in terms of its ability to learn the function $f(x)$. One of the points that is developed in this paper is that traditional convolution and ARMA dynamic models are equivalent to a two-layered, linear backpropagation net model. Thus, these traditional models are probably more limiting than a three-layered backpropagation model in terms of their ability to model dynamic process systems.

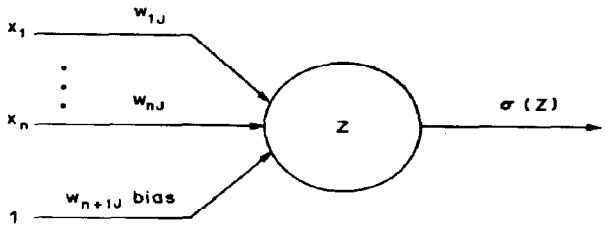

Fig. 2. jth neuron.

Algorithm

The governing equations for a backpropagation net, illustrated in Fig. 1, are briefly reviewed here. The inputs and outputs to the net have to be scaled into the range 0 to 1. As can be seen the net consists of three layers: an input, hidden and output layer. The neurons in the input layer simply store the scaled input values. The hidden layer and output layer neurons each carry out two calculations. To explain these calculations consider the general jth neuron shown in Fig. 2 and assume that this neuron is in the hidden layer. The inputs to this neuron consist of an N-dimensional vector x and a bias whose value is 1. Each of the inputs has a weight w_{ij} associated with it. The first calculation within the neuron consists in calculating the weighted sum S_j of the inputs as:

$$S_j = \sum_{i=1}^{N} w_{ij} x_i + w_{N+1j}. \tag{1}$$

Next the output of the neuron O_j is calculated as the sigma function of S_j as:

$$O_j = \sigma(S_j), \tag{2}$$

where

$$\sigma(z) = \frac{1}{1+e^{-z}}. \tag{3}$$

The sigma function is illustrated in Fig. 3. Once the outputs of the hidden layer are calculated, they are passed to the output layer. The output layer carries out the same calculations given by equations (1–3), except that the input vector x is replaced by the hidden layer output vector and the weights in equation (1) are those between the hidden and output layers w_{jk}. The governing equations for the output layer are:

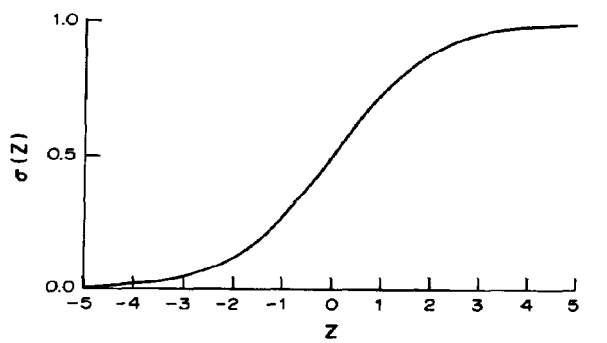

Fig. 3. Sigma function.

$$S_k = \sum_{j=1}^{M} w_{jk} O_j + w_{M+1k}, \tag{4}$$

$$O_k = o(S_k). \tag{5}$$

A backpropagation net learns by making changes in its weights. Assume that there are R input–output pairs $x^{(r)}$ $y^{(r)}$ available for training the net. Consider a mean-squared error performance criterion E defined as:

$$E = \sum_{m=1}^{R} \sum_{k=1}^{P} (y_k^{(m)} - O_k^{(m)})^2. \tag{6}$$

If a traditional gradient approach to minimizing E with respect to the w_{uv}s were taken, then $\partial E/w_{uv}$ would be calculated and one would move in a steepest descent direction. Such an approach requires using all of the input–output pairs to calculate the gradient. Backpropagation also uses gradient information to change the weights. However, the gradient is calculated only with respect to one input–output pair at a time. This input–output pair is presented to the net and then the weights are changed using the following expressions:

$$w_{uv}^{(m)} = w_{uv}^{(m-1)} + \Delta w_{uv}^{(m)}, \tag{7}$$

with

$$\Delta w_{uv}^{(m)} \propto -\frac{\partial E}{w_{uv}^{(m)}}. \tag{8}$$

Initially the weights in the net are randomized. Then the first input–output pair ($m = 1$) is presented to the net. Since the net is just beginning to learn, the net outputs O_k usually are quite different from the desired outputs y_k. Evaluation of equation (8) shows that the difference between O_k and y_k is involved in adjusting the weights. The specific versions of equation (8) which are used for each set of weights are:

$$w_{jk}^{(m)} = \eta \sigma'(S_k)(y_k^{(m)} - O_k^{(m)}) O_j^{(m)} \tag{9}$$

and

$$w_{ij}^{(m)} = \eta \sigma'(S_j) \left[\sum_{k=1}^{P} [\sigma'(S_k)(y_k^{(m)} - O_k^{(m)}) w_{jk}^{(m-1)} \right] x_i^{(m)}. \tag{10}$$

In adjusting the weights, first the weights between the hidden layer and the output layer w_{jk} are adjusted and then the weights between the input and hidden layers w_{ij} are changed. In adjusting w_{ij} the old values of w_{jk} are used, i.e. $w_{jk}^{(m-1)}$. After representation of the first input–output pair, one proceeds with the second pair, and so on. The weights are changed with each presentation. Results are presented later on which illustrate how a backpropagation net learns.

A mathematical insight into the capabilities of multilayered neural nets is given by Cybenko (1988, 1989). The first paper shows constructively that

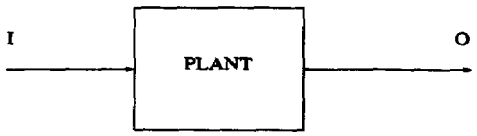

Fig. 4. SISO plant.

continuous valued neural networks with two hidden layers and any fixed continuous sigmoidal nonlinearity can approximate any continuous function arbitrarily well on a compact set. The second paper demonstrates the fact that arbitrary decision regions can be well approximated by continuous neural networks with only a single internal hidden layer and any continuous sigmoidal nonlinearity. This result confirms the belief that the common backpropagation network can indeed approximate any continuous function from R^m to R^n.

DYNAMIC MODELING APPROACHES USED IN MODEL-BASED PROCESS CONTROL

Model-based process control (MBPC) is an area that has received widespread attention recently (Garcia and Morari, 1982; McAvoy *et al.*, 1989a; Cutler and Ramaker, 1980). The reason for the interest is that MBPC has been used successfully to solve important, real-world process control problems. In MBPC linear, convolution dynamic models are typically used. Convolution modeling and its use for control are briefly reviewed here to develop insight into how backpropagation can be used for purposes of dynamic modeling and control. A single input–single output system, shown in Fig. 4 is treated for the sake of simplicity. Extension of the results to multivariable systems is straightforward. In developing a convolution model one would force the input, I, for example using a PRBS signal around the steady state value. The resulting output from the plant is illustrated in Fig. 5. Values of input and output would be recorded at discrete times and used to develop a model. An impulse response convolution model can be written to give the output at time step $k+1$ as (Garcia and Morari, 1982):

$$O_{k+1} = O_0 + \sum_{j=0}^{k} h_{k-j+1} I_j, \tag{11}$$

where the h_is are the impulse response coefficients, I_j is the input at time j and O_{k+1} is the output at time

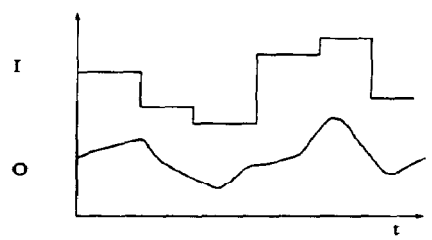

Fig. 5. Plant response to PRBS input.

$k+1$. Step response models have also been used (Cutler and Ramaker, 1980). In using equation (11) one would calculate the h_is by fitting the model to actual plant data using a least squares criterion. Typically a large number (~ 35) of h_is are used. These h_is are fitted offline and then updated perhaps in a few months.

It is interesting at this point to compare equation (11) to equation (1). If one interprets the weights as the impulse response coefficients and the bias term as the initial output, then these equations are identical. Both the weights in the net and the h_is are calculated from a least squares fit of data. Indeed, one can view convolution modeling as being equivalent to a two-layered backpropagation net without the sigmoid function being used. However, it is precisely the sigmoid and the three-layered structure that give backpropagation the ability to model nonlinear systems accurately. Since chemical processes are often nonlinear, using backpropagation for modeling holds the promise of producing more accurate models. This comparison between equations (1) and (11) also suggests how backpropagation can be used for dynamic modeling. The inputs to the net should be the plant inputs at discrete instants of time. This point is developed in the following section.

To use equation (11) for control, one splits the output up into one part that can be predicted from past inputs, and a part that depends on the current and future input. Assuming that the current time step is k, then the plant output in the future is:

$$\begin{bmatrix} \hat{O}_{k+1} \\ \hat{O}_{k+2} \\ \vdots \\ \hat{O}_{k+N} \end{bmatrix} = \begin{bmatrix} \bar{O}_{k+1} \\ \bar{O}_{k+2} \\ \vdots \\ \bar{O}_{k+N} \end{bmatrix} + A \begin{bmatrix} I_k \\ I_{k+1} \\ \vdots \\ I_{k+M} \end{bmatrix} + \begin{bmatrix} d \\ d \\ \vdots \\ d \end{bmatrix}, \tag{12}$$

where the A matrix is given by:

$$\begin{bmatrix} h_1 & & & \\ h_2 & h_1 & & \\ \vdots & \vdots & & \\ h_N & h_{N-1} & \cdots & h_1 \\ \vdots & \vdots & & \vdots \\ h_M & h_{M-1} & \cdots & h_{M-N+1} \\ \vdots & \vdots & & \vdots \\ h_M & h_M & \cdots & h_M \end{bmatrix}. \tag{13}$$

In equation (12) the $\bar{O}_i$s are the predicted outputs resulting from past plant inputs and the $\hat{O}_i$s are the predicted future outputs. The **d** vector accounts for differences between the actual and predicted plant outputs at $t = k$. This difference is:

$$d = O_k - \hat{O}_k \tag{14}$$

and it assumed to be constant in the future and it is added to the right-hand side of equation (12). If the

$\hat{O}_i$s in equation (12) are set equal to the desired set point O_D then one can solve for the current and future plant inputs. This solution typically has to be obtained in a least squares sense, since there are usually fewer values of I_i than there are future outputs.

In the actual implementation of MBPC, only the current plant input I_k is implemented. At the next time step, the entire calculation is repeated. The entire control procedure constitutes a one-step-ahead moving horizon algorithm. For additional details on the algorithm the interested reader is referred to the literature (Cutler and Ramaker, 1980; Garcia and Morari, 1982).

Adaptive control can also be considered as a form of model-based control. ARMA models are typically used in adaptive control. These models have the form:

$$O_{k+1} + \sum_{i=1}^{n} b_i O_{k\ i+1} = \sum_{j=1}^{n} a_j I_{k-j+1}. \qquad (15)$$

Usually the number of parameters in an ARMA model is significantly smaller than the number of h_i terms in equation (11). The a_i and b_i terms are estimated online and recursively from plant data. Equation (15) is also equivalent to a two-layered neural net (without the hidden node and having linear activation functions). Such a two-layered net is illustrated in Fig. 6. By using a backpropagation net with the sigmoid functions and three layers one should be able to achieve more accuracy than is possible with equation (15). In the following section both an ARMA and a convolution like backpropagation approach to dynamic modeling are discussed.

BACKPROPAGATION DYNAMIC MODELING (BDM)

In this section it is assumed that an historical database of plant inputs and outputs is available. To illustrate the neural net methodology, a specific pH CSTR is treated. After describing the system, the backpropagation methodology is presented.

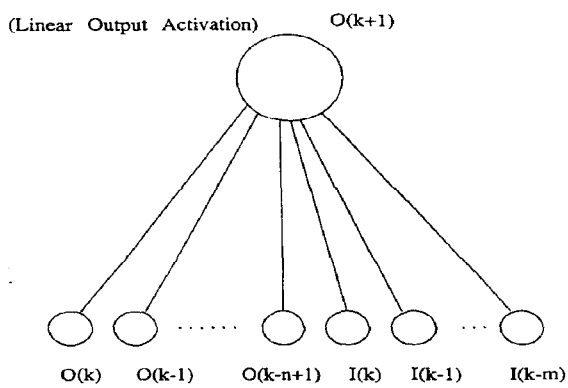

Fig. 6. ARMA model implemented as a two-layer backpropagation net.

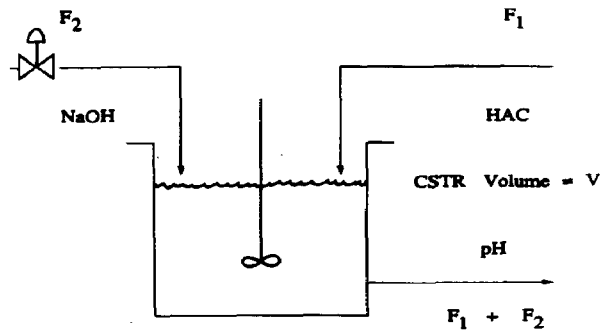

Fig. 7. pH CSTR.

System considered

We have studied the dynamic response of pH in a stirred tank reactor. The CSTR is shown schematically in Fig. 7. The CSTR has two input streams, one containing sodium hydroxide and the other acetic acid. A dynamic model for the pH in the tank can be obtained using the approach presented by McAvoy *et al.* (1972). By writing material balances on $Na^+(\zeta)$ and total acetate $(HAC + AC^-)(\xi)$ and assuming that acid–base equilibria and electroneutrality relationships hold one gets:

total acetate balance:

$$F_1 C_1 - (F_1 + F_2)\xi = V\frac{d\xi}{dt}; \qquad (16)$$

sodium ion balance:

$$F_2 C_2 - (F_1 + F_2)\zeta = V\frac{d\zeta}{dt}; \qquad (17)$$

HAC equilibrium:

$$\frac{[AC^-][H^+]}{[HAC]} = K_a; \qquad (18)$$

water equilibrium:

$$[H^+][OH^-] = K_w; \qquad (19)$$

electroneutrality:

$$\zeta + [H^+] = [OH^-] + [AC^-]. \qquad (20)$$

Parameters for the CSTR considered are given in Table 1. A training database was developed by forcing the F_2 stream with a 2% PRBS signal superimposed upon its steady state value. The F_2 and pH responses used for training are shown in Fig. 8. As can be seen, $[H^+]$ ranges over five orders of magnitude. In the pH model, equations (16) and (17) are essentially linear differential equations. Equations (18–20) are highly-nonlinear algebraic equations. Thus, one can view the model as having highly-nonlinear steady state characteristics and linear dynamic characteristics. Backpropagation dynamic modeling (BDM) is discussed next.

Table 1

Parameters used in the simulation	Value
Volume of the tank	1000 l.
Flow rate of acetic acid	81 l min^{-1}
Steady state flow rate of NaOH	515 l min^{-1}
Steady state pH	7
Concentration of acetic acid	0.32 mol l^{-1}
Concentration of NaOH	0.05 mol l^{-1}
Initial concentration of sodium in the CSTR	0.0432 mol l^{-1}
Initial concentration of acetate in the CSTR	0.0435 mol l^{-1}

Backpropagation dynamic modeling (BDM)

To model the pH response shown in Fig. 8, a backpropagation net was used. Various sizes for the input layer were tried. For the results given below, the input layer had 15 neurons and the hidden and output layers five neurons. The pH output was predicted one to five steps into the future. A time step Δt of 0.4 min is used. Methods similar to the ARMA and convolution type of modeling were tried. It was found that the ARMA approach, in which past pH values were used, gave better results and thus, the ARMA approach is discussed here.

The input to the net consisted of a moving window of pH and F_2 values, as illustrated in Fig. 9. The center of the window is taken as the current time t_0. Past and present values of pH and F_2 as well as future values of F_2 are fed to the net. The output from the net is the pH in the future. It should be emphasized here that since the database used is historical, the future values (relative to the center of the window) are known are these can be used for training. The following section on model-based control discusses how an optimizer can be used to calculate the future F_2 values.

At the beginning of the training process the window is placed at the beginning of the database. The first five pH and ten F_2 values are input to the net. The desired net outputs are the pH values at $t = t_0 + 1$, 2, 3, 4 and 5 Δts in the future. After the first presentation of data, the window moves Δt down the database. Again the current and past four pH and F_2 values and the future five F_2 values are input to the net. This process is continued until the end of the database is reached and then the process is repeated by starting at the beginning of the database. During training the proportionality factors η in equations (9) and (10) are lowered. After several cycles through the database the net converges as determined by the fact that the weights achieve fixed values.

Results

(a) BDM. Since the net's weights are initially randomized, the net does a very poor job in its predictions at first. After convergence the net gives excellent predictions, as the results in Fig. 10 show. Although the net predicts pH at one to five time steps into the future, only predictions for the first time step are shown for simplicity. The agreement between predicted and actual pH at the other four time steps is as good as that at one time step.

(b) ARMA. We have also tried traditional ARMA modeling on the data set in Fig. 8. To emphasize that ARMA modeling is equivalent to a two-layered neural net, we used such a net to carry out the calculations. The net used was a backpropagation net with no hidden layer and a linear output function and was the same as illustrated in Fig. 6. Initially the current and past four pH and F_2 values were fed to

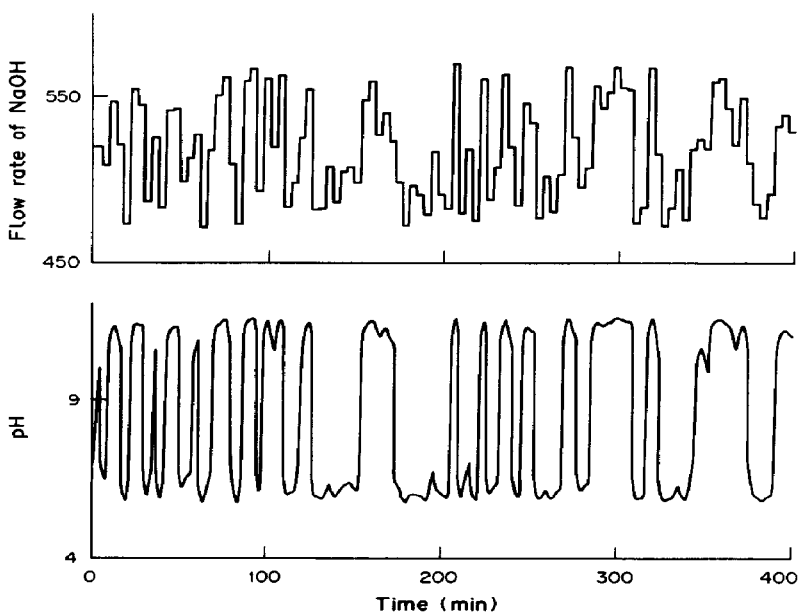

Fig. 8

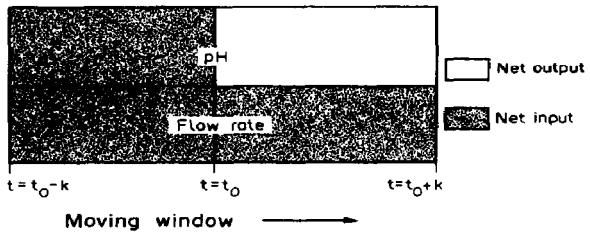

Fig. 9

current and past four pH and F_2 values were fed to the net. However, convergence problems were experienced with this configuration. When the present and past two pH and F_2 values were used, the results in Fig. 11 were obtained after convergence. As can be seen the ARMA model does reasonably well in predicting the pH response.

(c) Comparison of BDM and ARMA methods. To illustrate the ability of the BDM approach to learn the nonlinear characteristics of a model, the operating conditions of the CSTR were changed to correspond to a steady state pH of 7. Note that the original PRBS training encompassed pH equal to 7. Both the BDM and ARMA model parameters were frozen at the values obtained around a pH of 9. At a pH of 7 a 1% PRBS signal was added to the F_2 flow and the data shown in Fig. 12 were obtained. Also shown in Fig. 12 are the predictions of the BDM and ARMA models. As can be seen the BDM model is closer to the actual pH. In effect the backpropagation net has been able to learn the nonlinear model characteristics better than the ARMA model. Both modeling methods predict the shape of the dynamic responses reasonably well. However, the BDM method does better with the nonlinear steady state characteristics than the ARMA model does.

We have also tried the BDM approach with similar success for cases where both F_1 and F_2 were varied. Also, corrupting the pH outputs with noise presents no problem for the net. We have tried BDM modeling on linear systems with dead time and inverse response and have been successful. The following section discusses how a BDM model can be used for process control.

BACKPROPAGATION MODEL-BASED CONTROL (BDMC)

Once a trained backpropagation model is available, then it can be used in a straightforward manner for process control. Figure 13 presents a schematic diagram showing how the neural model is used. The approach is essentially the same as that used in dynamic matrix control (Cutler and Ramaker, 1980), except that the nonlinear neural net model is used in place of the linear convolution model. Since dynamic matrix control has been discussed extensively in the literature (Cutler and Ramaker, 1980; Garcia and Morari, 1982), only a brief discussion is given here. As Fig. 13 shows an optimizer is used to calculate the future F_2s. The optimization variables are $F_2(m)$, $m = k$ to $k + m$. After M time steps the F_2s are held constant. If the desired pH is pH_D, then the F_2s can be chosen to minimize a weighted sum of desired minus predicted pHs as:

$$\min_{F_2(m)} J = \sum_{n=1}^{N} [pH_D - pH(t_0 + nt)]^2. \qquad (21)$$

Typically, several future F_2s would be calculated, but only the first, $F_2(k)$, would be implemented. Constraints on the size of F_2 moves and high and low

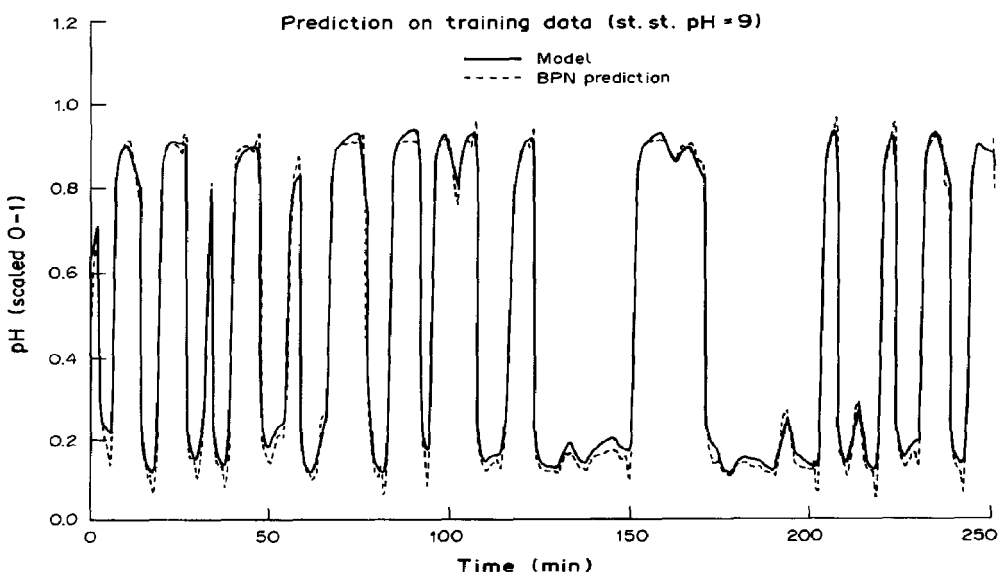

Fig. 10

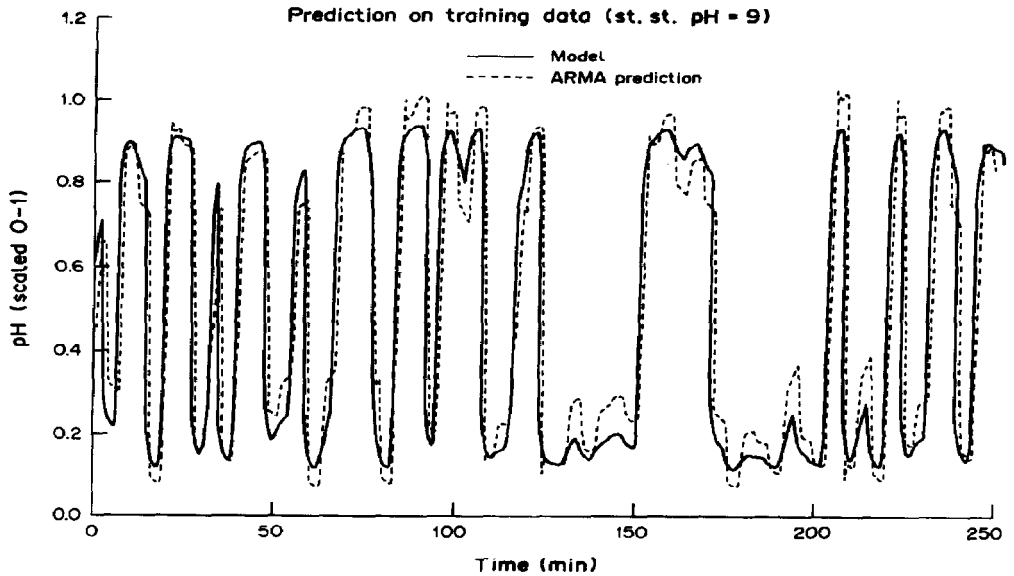

Fig. 11

values for F_2 and pH can be incorporated into the problem. This procedure constitutes a one-step-ahead moving horizon problem. Since the BDM models are nonlinear, the optimization equation (21) is more difficult to solve than if the linear convolution or ARMA models were used.

An alternative approach to using neural nets for control involves determining the inverse process model. To illustrate this approach, the data shown in Fig. 8 will be used. Referring to Fig. 9, one could switch the future pHs (outputs) with the future F_2s (inputs) and develop an inverse plant model. Such a net would predict what flow rates are necessary to achieve a desired pH. This approach to modeling was tried on the data in Fig. 8 using 15 input, and five hidden and output neurons. The results are shown in Fig. 14. As can be seen, a backpropagation net does a good job of learning the inverse process model. Once the inverse model is available then it can be used in an internal model control structure, as shown in Fig. 15.

One last point should be made about using neural net dynamic models for control. The nature of back-propagation modeling is such that it lends itself

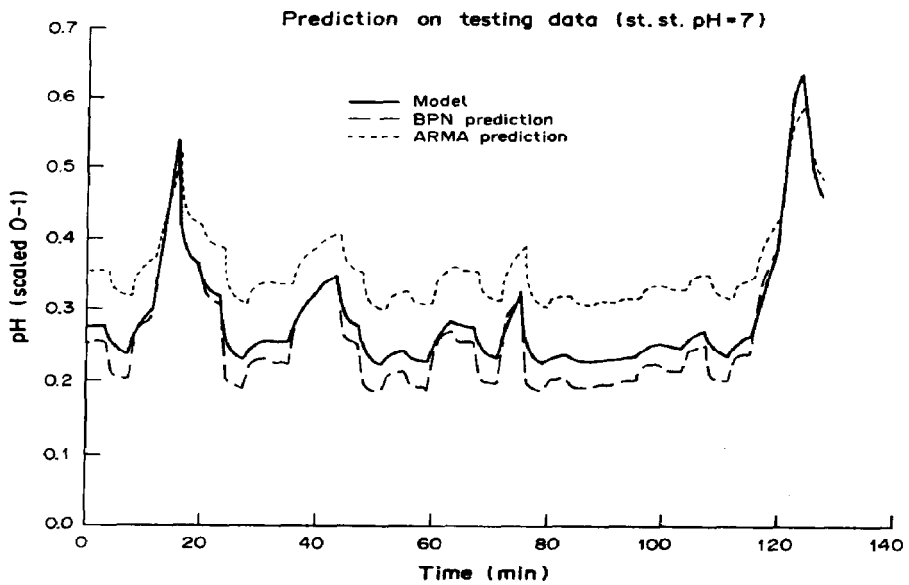

Fig. 12

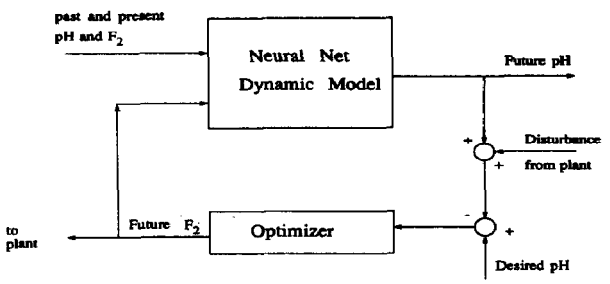

Fig. 13

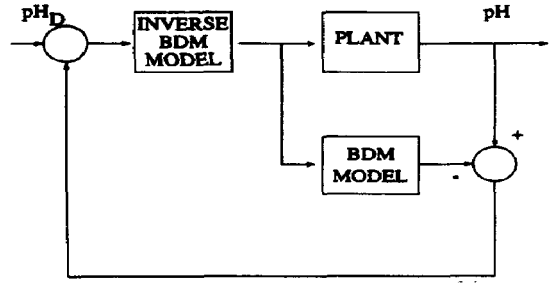

Fig. 15. IMC structure employing inverse and normal neural net models.

naturally to an adaptive implementation. Many different schemes are possible. For example, one could have two nets and use one for control and one for learning. Both nets would start out the same and they would be converged on an historical data base. The net used for control would be frozen. The other net would continue to learn as new data became available. Such learning could only take place when there was sufficient information in the plant inputs and outputs. When there was a significant difference between the two nets, the one being trained would replace the frozen net and the process continued. We are currently carrying out research on several approaches to model-based control and expect to report our results in the near future.

CONCLUSIONS

This paper has discussed the use of backpropagation neural nets for learning nonlinear dynamic models from plant input–output data. Because of its ability to identify nonlinear relationships and because it is fundamentally a parallel technique, backpropagation is a very promising neural algorithm. The research reported here differs from most published work on backpropagation which is primarily concerned with steady state modeling. It has been pointed out that traditional ARMA and convolution models are special cases of backpropagation nets. These traditional models are equivalent to two-layered linear backpropagation nets and as such they are more limited in their ability than the complete backpropagation algorithm.

The backpropagation dynamic modeling (BDM) approach was illustrated on a pH CSTR that was forced with a PRBS input. A BDM model trained at a steady state pH of 9 was shown to be more accurate than an ARMA model when the pH was lowered to 7. In effect the BDM model did a better job of learning the governing nonlinear pH relationships. The use of BDM models for process control was also outlined. The BDM models can be used in place of traditional ARMA and convolution models. Alternatively, one can train a BDM net to learn the inverse of the process model and then employ the inverse in

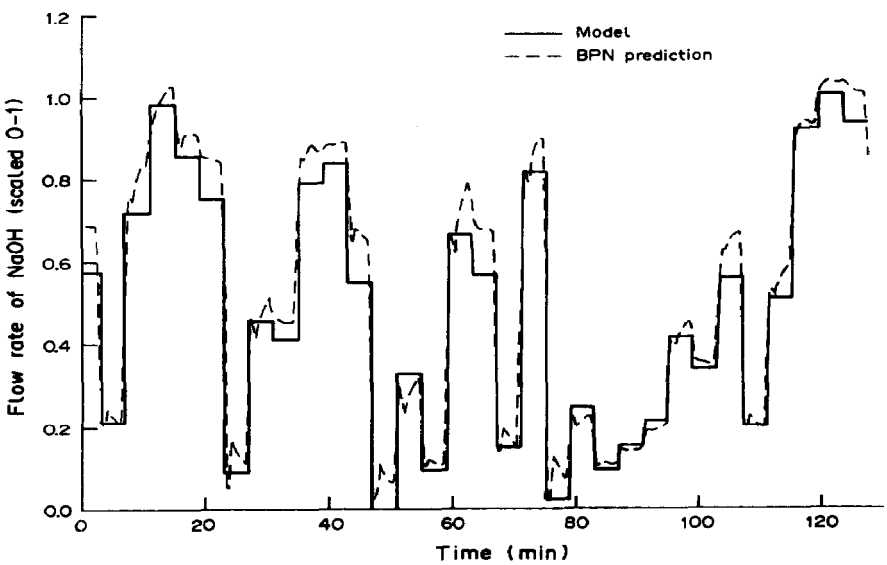

Fig. 14. Inverse process model.

an internal model control structure. Finally BDM models lend themselves naturally to adaptive control implementation.

REFERENCES

Bhat N. and T. McAvoy, Use of neural nets for dynamic modeling and control of chemical process systems. *1989 Am. Automatic Control Conf.*, Pittsburgh (1989).

Birky G. and T. McAvoy, A neural net to learn the design of distillation controls. *IFAC DYCORD Symp.*, Maastricht, The Netherlands (1989).

Bounds D., P. Lloyd, B. Mathew and A. Waddell, Multilayer perceptron for the diagnosis of low back pain. *IEEE Int. Conf. on Neural Networks*, San Diego (1988).

Carpenter G. and S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine. *Comput. Vision Graphics Image Process.* **37**, 54–115 (1987).

Castelaz P., Neural networks in defense applications. *IEEE Int. Conf. on Neural Networks*, San Diego (1988).

Cutler C. and B. Ramaker, Dynamic matrix control—a computer control algorithm. *AIChE 86th National Meeting*, Houston (April 1979); also in *Joint Automatic Control Conf. Proc.*, San Francisco (1980).

Cybenko G., Continuous valued neural networks with two hidden layers are sufficient. Report, Department of Computer Science, Tufts University, Medford (1988).

Cybenko G., Approximations by superpositions of a sigmoidal function. *Math. Control Signal Systems* **2**, 303–314 (1989).

Dutta S. and S. Shekar, Bond rating: a non-conservative application of neural networks. *IEEE Int. Conf. on Neural Networks*, San Diego, (1988).

Elman J. and D. Zipser, Learning the hidden structure of speech. Institute for Cognitive Science, University of California at San Diego, ICS Report 8701 (1987).

Elsley R., A learning architecture for control based on backpropagation neural networks. *IEEE Int. Conf. on Neural Networks*, San Diego (1988).

Fukushima K. and S. Miyake, Necognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition.* **15**, 455–469 (1984).

Garcia C. and M. Morari, Internal model control. 1. A unifying review and some new results. *Ind. Engng Chem. Process Des. Dev.* **21**, 308–323 (1982).

Gorman R. and T. Sejnowski, Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks* **1**, 75–89 (1988).

Grossberg S., *Studies of Mind and Brain*. Reidel, Dordrecht, Holland (1982).

Grossberg S., *The Adaptive Brain*, Vols I and II. Elsevier/North-Holland, Amsterdam (1986).

Hecht-Nielsen R., Counterpropagation networks. *Appl. Optics* Dec, (1987a).

Hecht-Nielsen R., Counterpropagation networks. *Proc. IEEE Int. Conf. on Neural Networks* (1987b).

Hecht-Nielsen R., Neurocomputer applications. *Neural Computers* (Eckmiller *et al.*, Eds). Springer-Verlag, New York (1988).

Hinton G. and J. Anderson (Eds), *Parallel Models of Associative Memory*. Erlbaum, Hillsdale, New Jersey (1981).

Kohonen T., *Self-Organization and Associative Memory*, Second Edition. Springer-Verlag, Berlin (1987).

Kosko B., Bidirectional associative memories. *BYTE Mag.* Sept, (1987a).

Kosko B., Fuzzy associative memories. *Fuzzy Expert Systems* (A. Kandel, Ed.). Addison-Wesley, Reading, Mass. (1987b).

McAvoy T., E. Hsu and S. Lowenthal, Dynamics of pH in CSTRs. *Ind. Engng Chem. Process Des. Dev.* **11**, 68–70 (1972).

McAvoy T., Y. Arkun and Zafiriou E. (Eds), *Proc. of IFAC Workshop on Model Based Control*, Atlanta. Pergamon Press, London (1989a).

McAvoy T., N. Wang and N. Bhat, Use of neural nets for interpreting biosensor data. *1989 American Automatic Control Conf.*, Pittsburgh (1989b).

Minskey M. and S. Papert, *Perceptrons*. MIT, Cambridge, Mass. (1972).

Peeling S., R. Moore and M. Tomlinson, The multi-layer perceptron as a tool for speech pattern processing research. *Proc. IoA Autumn Conf. on Speech and Hearing* (1986).

Rumelhart D. and J. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. I, Chap. 8. MIT, Cambridge, Mass. (1986).

Sejnowski T. and C. Rosenberg, NETtalk: a parallel network that learns to read aloud. Johns Hopkins University, Technical Report JHU/EECS-86/01 (1986).

Steinbuch K. and U. Piske, Learning matrices and their applications. *IEEE Trans. Elec. Comput.* **12**, (1963).

Szu H. (Ed.), *Optical and Hybrid Computing*, *SPIE Proc.* **634**, (1986).

Werbos P., Beyond regression: new tools for prediction and analysis in behavioral sciences. Ph.D Thesis, Harvard University (1974).

Werbos P., Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks* **1**, 339–356 (1988).

Widrow B. and M. Hoff, Adaptive switching circuits. IRE WESCON Conv. Record, Part 4, pp. 96–104 (1960).

Widrow B. and S. Stearns, *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, New Jersey (1985).