A New Scheme Combining Neural Feedforward Control with Model-Predictive Control

Moonyong Lee and Sunwon Park
Dept. of Chemical Engineering, KAIST,
Taejon 305-701, Korea

A new control scheme is presented for feedforward control of unknown disturbances in the model-predictive control (MPC) scheme. In this control scheme, a neural network is connected in parallel with the MPC controller and trained on-line by minimizing the MPC controller output corresponding to the unmodeled effect. It is applied to distillation column control and nonlinear reactor control to illustrate its effectiveness. The result shows that the neural feedforward controller can cope well with strong interactions, time delays, nonlinearities, and process/model mismatch. The controller also offers such advantages as fault tolerance, generalization capability by interpolation, and learning capability by random input patterns.

Introduction

During the past decade, there has been considerable interest in model-predictive control (MPC) techniques. Typical algorithms such as dynamic matrix control (DMC) (Cutler and Ramaker, 1980) and model algorithmic control (MAC) (Richalet et al., 1978; Rouhani and Mehra, 1982) have been used in various industrial processes and have proved to be effective control algorithms.

Since MPC techniques are based on linear models, their performance is affected largely by modeling errors, which are caused mainly by system uncertainties and nonlinearities. System uncertainties exist almost always in the real plant. Lack of precision of the model makes it difficult to design model-based controllers even for linear systems. Many chemical processes have nonlinear dynamics. The linear approximation results in performance loss and limits the operating range of the MPC controller. Therefore, there have been efforts to develop robust nonlinear MPC controllers in recent years (Brengel and Seider, 1989; Li and Biegler, 1989; Peterson et al., 1989; Petwardhan et al., 1990). Most of these techniques, however, are based on nonlinear models, thus often requiring nonlinear identification techniques. For this reason, extensive research efforts have been made to the application of artificial neural networks to MPC, which appears to have the potential to identify complex nonlinear systems. Some recent attempts have shown considerable promise (Bhat and McAvoy, 1990; Hernandez and Arkun, 1990; Roat and Moore, 1990; Psichogios and Ungar, 1990; Ydsite, 1990).

In this article, we focus only on feedforward control using a neural network in the MPC scheme. Load disturbance rejection is important especially in chemical process control. In the MPC scheme, this regulatory performance can be greatly improved by incorporating the disturbance model into MPC (Cutler and Ramaker, 1980; Garcia and Morari, 1985). The disturbance dynamics of many processes, however, are not well-known and/or not characterized in a simple model due to their uncertainties and complexities. In addition, the nonlinear characteristics of the disturbance dynamics of many chemical processes also limit the use of the linearized disturbance model. We present here a simple control scheme for neural-network-based feedforward control for unknown disturbances in the MPC scheme. In this controller, a neural network is connected in parallel with the MPC controller. Since the neural network learns a relationship between disturbance patterns and desired control actions by minimizing the MPC controller output corresponding to the unmodeled effect through on-line training, any prior knowledge of the disturbance dynamics is not required. Also, any steps to solve nonlinear problems are not required, since the explicit solution of the standard MPC controller is still employed.

The back-propagation algorithm and the DMC algorithm are briefly reviewed, followed by the presentation of a control scheme combining the neural network with MPC. Subse-

Correspondence concerning this article should be addressed to S. Park.

quently, simulation results for two applications are presented using an example of distillation column control, various properties of the controller and the effect of the process/model mismatch are tested; using an example of nonlinear reactor control, its effectiveness on the nonlinear system is shown.

Neural Networks and Error Back-Propagation

The neural network model used is the multilayered feedforward neural network with the back-propagation algorithm (Rumelhart and McClelland, 1986), which is most relevant to control applications. Figure 1 shows a schematic of this model.

The back-propagation algorithm involves two phases. In the first phase, input signals are propagated in a feedforward manner through the network to produce actual outputs. The governing equation of this phase is given by:

$$O^l_{pj}=F\left(\sum_{i=1}^{N^{l-1}} W^l_{ij}O^{l-1}_{pi}\right) \quad \text{for } j=1, \ldots, N^l, \text{ and } 1<l\leq L \qquad (1)$$

where O^l_{pj} is the output of the jth neuron in the lth layer for pattern p, W^l_{ij} is the connection weight between the ith neuron in the $(l-1)$th layer and the jth neuron in the lth layer, $F(x)$ is a nonlinear activation function, N^l is the number of neurons in the lth layer, and L is the total number of layers or the output layer number.

In the second phase, the actual outputs are compared to the target outputs, and the errors are propagated backward through the network and used to adjust the connection weights. W^l_{ij} at the $(k+1)$th learning step is adjusted as follows:

$$W^l_{ij}(k+1)=W^l_{ij}(k)+\eta\Delta W^l_{ij} \qquad (2)$$

where ΔW^l_{ij} is:

$$\begin{aligned}\Delta W^l_{ij}&=(T_{pj}-O^l_{pj})F'\left\{\sum_{i=1}^{N^{l-1}} W^l_{ij}(k)O^{l-1}_{pi}\right\}O^{l-1}_{pi}\\&=\delta^l_{pj}O^{l-1}_{pi} \quad \text{for the output layer, that is } l=L \qquad (3)\end{aligned}$$

$$\begin{aligned}\Delta W^l_{ij}&=\sum_{k=1}^{N^{l+1}}\delta^{l+1}_{pk}W^{l+1}_{jk}(k)F'\left\{\sum_{i=1}^{N^{l-1}} W^l_{ij}(k)O^{l-1}_{pi}\right\}O^{l-1}_{pi}\\&=\delta^l_{pj}O^{l-1}_{pi} \quad \text{for the hidden layers, that is, } 1<l<L \qquad (4)\end{aligned}$$

where T_{pj} is the target output of the jth neuron in the output layer for pattern p, $F'(X)$ is the derivative of the activation function with respect to x, and η is the learning rate coefficient.

The entire process is repeated over the training input-output patterns until the network produces the desired responses.

Dynamic Matrix Control and Disturbance Error

Various MPC algorithms may seem to differ from one another, but their main strategies are identical: (1) at the present moment, k, the future outputs of the plant are predicted based on a given model; (2) the present and future control actions are computed in such a way that they minimize the given prediction horizon cost function; and (3) only the present control action is actually implemented, and the whole procedure is repeated at the subsequent time period, $k+1$.

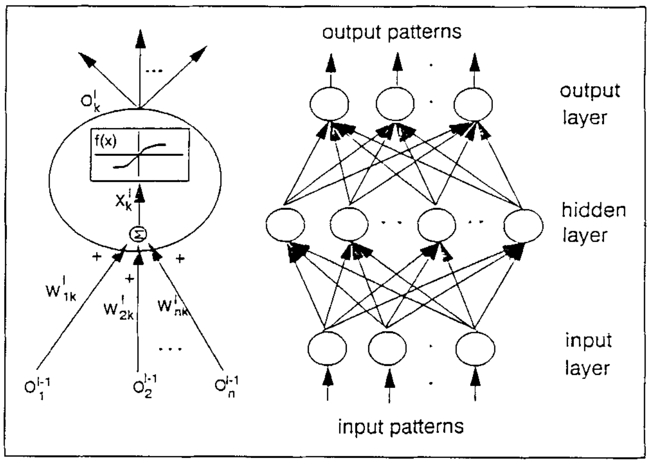

Figure 1. Architecture of the multilayered feedforward neural network.

The DMC algorithm is chosen as an example of MPC, and DMC for a single-input/single-output system will be briefly described. Extension of the results to multivariable systems and/or other MPC algorithms is straightforward.

In DMC, the cost function to be minimized can be written as:

$$J(\Delta U_c)=(A\Delta U_c-e)^T\Gamma^T\Gamma(A\Delta U_c-e)+\Delta U_c^T\Lambda^T\Lambda\Delta U_c, \qquad (5)$$

where

$$\Delta U_c=\begin{bmatrix}\Delta U_c(\bar{k})\\ \vdots \\ \Delta U_c(\bar{k}+N-1)\end{bmatrix} \qquad (6)$$

is the $N\times 1$ vector of the future control moves for the present time, $\bar{k}$, up to the future time, $\bar{k}+N-1$, where N is the control horizon, A is the $P\times N$ matrix called "dynamic matrix" (Cutler and Ramaker, 1980), in which P is the prediction horizon, Γ is the positive weighting matrix for selective weighting of controlled variables, Λ is the positive weighting matrix for the move suppression of manipulated variables, and

$$e=\begin{bmatrix}R-\hat{Y}^*(\bar{k}+1)-d(\bar{k}+1)\\ \vdots \\ R-\hat{Y}^*(\bar{k}+P)-d(\bar{k}+P)\end{bmatrix} \qquad (7)$$

is the $P\times 1$ vector of predicted deviations from the set point for future times, $\bar{k}+1$ up to $\bar{k}+P$, $\hat{Y}^*$ is the contribution of past control inputs to the projection, and d accounts for unmodeled factors.

The resulting control law that minimizes the cost function J is:

$$\Delta U_c=(A^T\Gamma^T\Gamma A+\Lambda^T\Lambda)^{-1}A^T\Gamma^T\Gamma e=Ke \qquad (8)$$

where K is the $N\times P$ matrix of feedback gains. Only the first element $\Delta U_c(\bar{k})$ in Eq. 8 is implemented.

At this point, let us focus on the term d that accounts for the unmodeled effect on the measured output. The unmodeled effect includes both unmodeled disturbances and modeling errors. Modeling errors are due mainly to nonlinearities and uncertainties in the system. The present $d(k)$ can be estimated using the current feedback measurement $Y(k)$ as follows:

$$d(k) = Y(k) - \hat{Y}^*(k) \qquad (9)$$

In DMC, each predicted disturbance $d(\bar{k}+l)$ for any future time $\bar{k}+l$ is assumed to be equal to the present $d(k)$, since future values of $d(k)$ are not available. In MAC, on the other hand, each predicted disturbance $d(\bar{k}+l)$ is filtered through a first-order exponential filter.

Breaking up $\boldsymbol{e}$ into the modeled and unmodeled contributions, Eq. 7 can be expressed as:

$$\boldsymbol{e} = \begin{bmatrix} R - \hat{Y}^*(\bar{k}+1) \\ \vdots \\ R - \hat{Y}^*(\bar{k}+P) \end{bmatrix} + \begin{bmatrix} -d(\bar{k}+1) \\ \vdots \\ -d(\bar{k}+P) \end{bmatrix} = \boldsymbol{e}_m + \boldsymbol{e}_d \qquad (10)$$

where $\boldsymbol{e}_m$ and $\boldsymbol{e}_d$ are the predicted error vectors caused by the modeled and unmodeled effects, respectively. For simplicity, we will call $\boldsymbol{e}_d$ "disturbance error." Then, Eq. 8 becomes:

$$\Delta \boldsymbol{U}_c = \boldsymbol{K}(\boldsymbol{e}_m + \boldsymbol{e}_d) = \Delta \boldsymbol{U}_m + \Delta \boldsymbol{U}_d \qquad (11)$$

where $\Delta \boldsymbol{U}_m$ and $\Delta \boldsymbol{U}_d$ are the vectors for the control input change at different time intervals due to the modeled and unmodeled effects, respectively.

From Eq. 11, we can interpret the control action in DMC as the sum of two factors, each of which plays a totally different role. Clearly, the control action by $\Delta \boldsymbol{U}_d$ results from the unmodeled effect and plays a role of compensating the unmodeled disturbances and modeling errors. In the ideal case where there are no disturbances and no modeling errors, $\Delta \boldsymbol{U}_d$ is equal to zero and DMC should be operated according to only $\Delta \boldsymbol{U}_m$. As the unmodeled disturbances and modeling errors increase, the effect of $\Delta \boldsymbol{U}_d$ increases. As will be shown later, $\Delta \boldsymbol{U}_d$ plays an important role in training the neural network in the proposed control scheme.

Control Scheme Using Disturbance Error Learning

In this section, we propose a new scheme that combines neural-network-based feedforward control with MPC. All major disturbances are assumed measurable. Figure 2 shows the proposed control scheme. The objective of training the network is to find the connection weights with which the network produces the control action $\boldsymbol{u}$ to minimize the square error $\|\boldsymbol{R} - \boldsymbol{Y}\|_2^2$ over a specific disturbance domain. However, we cannot apply the error back-propagation algorithm directly to this control scheme, because the target output (the control input that minimizes the square error) is not known *a priori*. Information about the correctness of the network output is available only through the plant output.

This problem can be avoided by using an appropriate target signal (Lee and Park, 1991). This method is based on the work by Kawato et al. (1988) and Setoyama et al. (1988), who showed that a conventional PD controller output could be used as the error signal for training neural-network-based robot controllers. In fact, their results can be extended to general linear feedback controllers. The success of training depends directly on the proper choice of the feedback controller output such as how to decompose the feedback controller output and which portion to use as the error signal (Lee and Park, 1991).

Based on the analysis in the earlier section, the sum of the network output, U_n, and the DMC controller output by the unmodeled effect, ΔU_d, is used as the target signal. The target signal at the kth time step, $T_j(k)$, is:

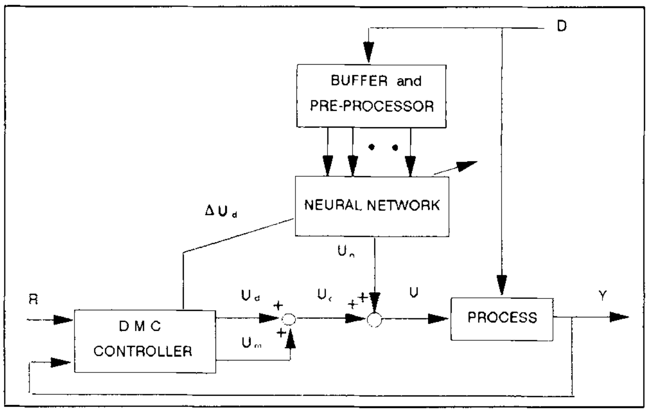

Figure 2. Control scheme combining the neural network with DMC.

$$T_j(k) = U_{nj}(k-1) + \Delta U_{dj}(k) \qquad (12)$$

The target signal $T_j(k)$ is then compared with the network output at the $(k-1)$th time step, $U_{nj}(k-1)$. Thus, at the kth time step, δ_j^L of the output layer in Eq. 3 becomes:

$$\delta_j^l = \Delta U_{dj}(k) F'\left(\sum_{i=1}^{N^{l-1}} W_{ij}^l O_i^{l-1}\right) \text{ for } l = L \qquad (13)$$

As shown, the network compares its actual output with the target signal, instead of the unknown target output, thus adjusting its weights according to ΔU_d. Initially, the target signal may quite differ from the desired target output, but it gradually approaches the target output as learning proceeds and becomes identical with the target output once learning is successfully accomplished. We call this learning technique *disturbance error learning*. Since the proposed target signal provides the correct gradient direction for the network training, learning is achieved in such a way that for a specific disturbance domain, the square of $\Delta \boldsymbol{U}_d$ is minimized and consequently the square error $\|\boldsymbol{R} - \boldsymbol{Y}\|_2^2$ is minimized.

Currently we cannot prove mathematically the stability of the proposed learning scheme. It, however, is intuitively clear that no stability problems are expected as long as the learning rate is sufficiently slower than the time constants of the other components of the control system, as mentioned by Kawato et al. (1988) and Psaltis et al. (1988).

The overall learning process is as follows. At each time step k, the disturbance $\boldsymbol{D}(k)$ is measured, and the DMC controller outputs $\Delta \boldsymbol{U}_d(k)$ and $\Delta \boldsymbol{U}_m(k)$ are computed. Since the state of a dynamic system depends on the past as well as the current disturbance, the current and dominant past values of the disturbance are necessary for the dynamic relationship. These disturbance signals are scaled and stored in the buffer and preprocessor (BPP). After one learning step is performed using $\Delta U_d(k)$ according to the disturbance error learning, the network receives the scaled past and current values of the disturbances from the BPP and produces the network output $U_n(k)$. $U_n(k)$ is then added to the DMC controller output $U_c(k)$ to form the manipulated variable $U(k)$, which is applied to the process. This entire process is repeated at each time step.

The initial connection weights of the network and the learning rate coefficient are chosen sufficiently small so that $U_n \cong 0$; during the initial training period, the network has little influence over the control actions and all control actions for disturbance rejection are performed by the DMC controller. As learning proceeds, the network tries to configure itself to minimize ΔU_d. As ΔU_d becomes small, the feedback action by the DMC controller is gradually switched to the feedforward action by the network. Learning and control are simultaneously performed by the network during the training period. When the error between the desired and actual plant responses becomes sufficiently small, the training has been accomplished, and only a relatively small ΔU_d is necessary to compensate for the random uncertainties. The trained neural controller has the characteristics of feedforward-feedback control: the neural network makes rapid and initial corrections, while the DMC controller makes long-term and precise compensations.

Application to Distillation Column Control

The well-known model by Wood and Berry (1973) was chosen for testing various properties of the neural controller. This linear model has several typical features of chemical processes, such as strong interactions and significant time delays. The model is given by:

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \\ \dfrac{6.6e^{-7s}}{10.9s+1} & \dfrac{-19.8e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} + \begin{bmatrix} \dfrac{3.8e^{-8s}}{14.9s+1} \\ \dfrac{4.9e^{-3s}}{13.2s+1} \end{bmatrix} D \quad (14)$$

The controlled variables Y_1 and Y_2 are the overhead and bottom methanol compositions, respectively. The manipulated variables U_1 and U_2 are the reflux and steam rates, respectively. The disturbance D is the feed rate. The steady-state values of the overhead and bottom methanol compositions are 96.25 mol % and 0.5 mol % , respectively.

The sampling time interval was 3 min. The tuning parameters of the DMC controller selected were $P=40$, $N=5$, $\Lambda = I$, and $\Gamma = I$. Figure 3 shows the BPP and specific configuration of the network employed in this work. A three-layered network was used, because in most applications the association of input patterns with desired outputs could be achieved with a three-layered architecture. The input layer contains four neurons and receives the input signals comprising $D(k-n)$, where $n = 0,1,2,3$. The hidden layer has ten neurons. The output layer has two neurons and produces controller signals as its outputs. No attempt was made to optimize the parameters of the network and the DMC controller. The initial weights of the network were randomized small enough that during the initial training period the network had little influence over the manipulated variable U. The hyperbolic tangent function was used as the activation function. The learning rate coefficient $\eta = 0.4$ was chosen.

Neural controller training

Simple training patterns lasting for 900 min, including ten consecutive random step changes in the feed flow, were repeated until the desired performance was achieved. Disturbance patterns for one training cycle are shown in Figure 4a. To measure the performance, the integral square error (ISE) for one training cycle was periodically checked at each training cycle. Figure 5 shows changes in the ISE vs. the number of training cycles. The ISE's of the bottom and top compositions decrease significantly during only a small number of training cycles, as shown in Figure 5. Gradual improvement in the control law implemented by the network is evident from the result.

Figure 6 compares the ΔU_d responses of the neural controller, before and after learning, to the training disturbance patterns. Note that the initial untrained neural controller is the same as the DMC controller as mentioned earlier. Once the network adapts to or learns the disturbance-desired control action relationship, ΔU_d due to the unmodeled effect mostly vanishes as shown in Figure 6. Figure 7 compares the control

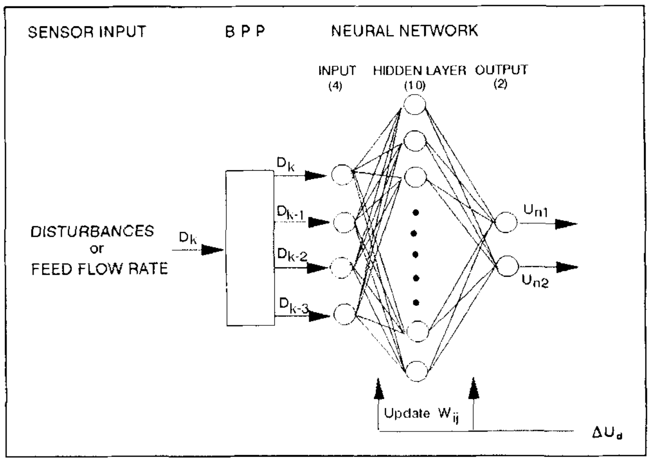

Figure 3. BPP and three-layered neural network used in the simulation.

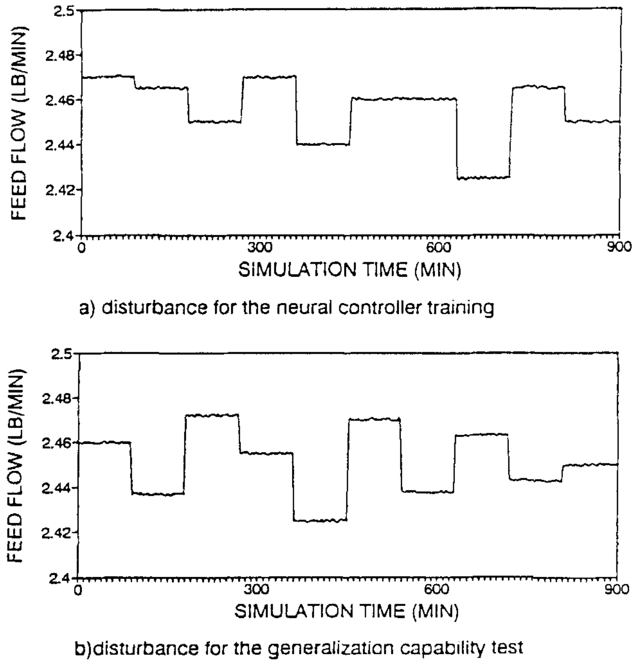

Figure 4. Disturbance patterns for the training and generalization capability test.

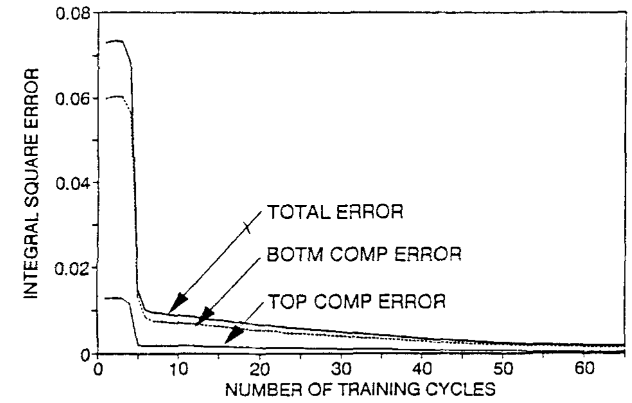

Figure 5. Changes of the ISE in training the neural network.

actions by the neural network and the DMC controller in the trained neural controller in response to the training disturbance patterns. In Figure 7, Y axis represents deviation from the steady-state value. Since the trained network takes the control action to eliminate the impact of the measured disturbances on the process outputs in a feedforward manner, most control action for disturbance rejection is performed by the network and only a relatively small control action is achieved by the DMC controller, as shown in Figure 7.

Generalization capability by interpolation

The primary advantage of the neural network is its generalization capability by interpolation, through which, it can respond correctly to the disturbance patterns for which it has not specifically been trained. Thus, it is not necessary to include every possible case for training. Figure 8a shows the output responses of the DMC controller alone, and Figure 8b shows the trained neural controller to the disturbance patterns in Figure 4b, which differ significantly from the training patterns in Figure 4a. The result shows that the neural controller consistently performs well even for the inexperienced disturbance patterns.

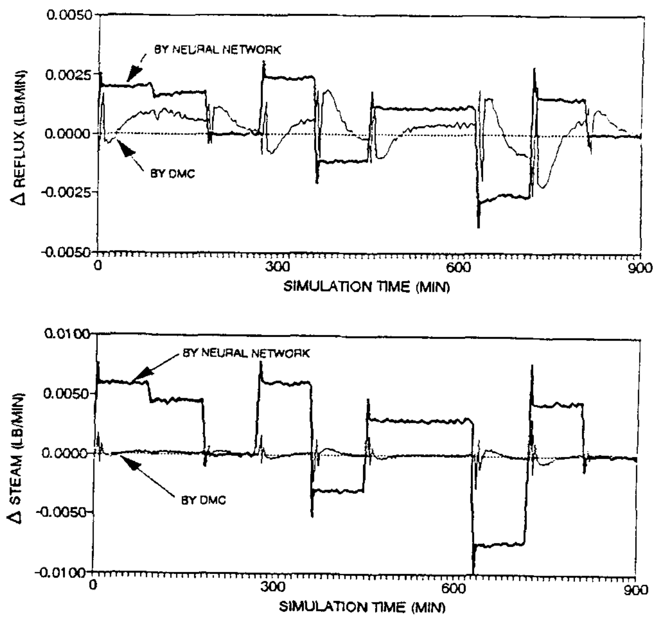

Figure 7. Control actions U_c vs. U_n in the trained neural controller in response to the training disturbance patterns.

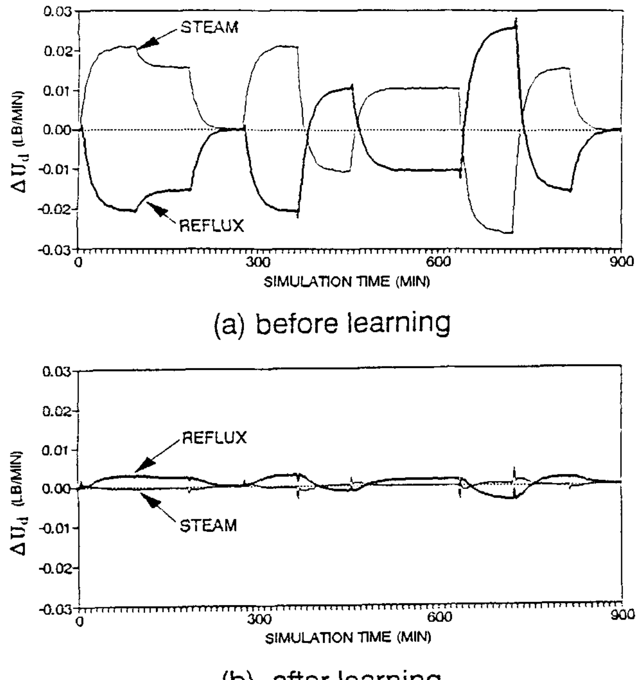

Figure 6. Control actions ΔU_d of the neural controller to the training disturbance patterns: before vs. after learning.

Fault tolerance

In general, the performance of the feedforward controller based on the transfer function model is very sensitive to the controller parameters. On the other hand, the neural controller is robust in the faults of the connection weights because the way the neural network stores information is highly distributed. To investigate the fault-tolerant characteristics of the neural controller, one half of the connection weights of the trained neural network were randomly set to zero. Figure 8c presents the output responses of the damaged neural controller to the disturbances shown in Figure 4b. Despite the significant faults, the damaged controller still maintains reasonably robust control performance.

Effect of process/model mismatch

In most cases, it is inevitable to have modeling errors as mentioned earlier. Thus, robustness against the modeling errors is indispensable to the model-based controller design. For the equal base comparison of the neural controller and the feedforward DMC controller, we considered the case where the errors exist only in the process model (that is, relationship between the controlled variables and the manipulated variables), but no errors in the disturbance model (the relationship between the controlled variables and the disturbances). We also assumed that real process dynamics were as Eq. 15, while the same dynamic matrix A as in the original model was used:

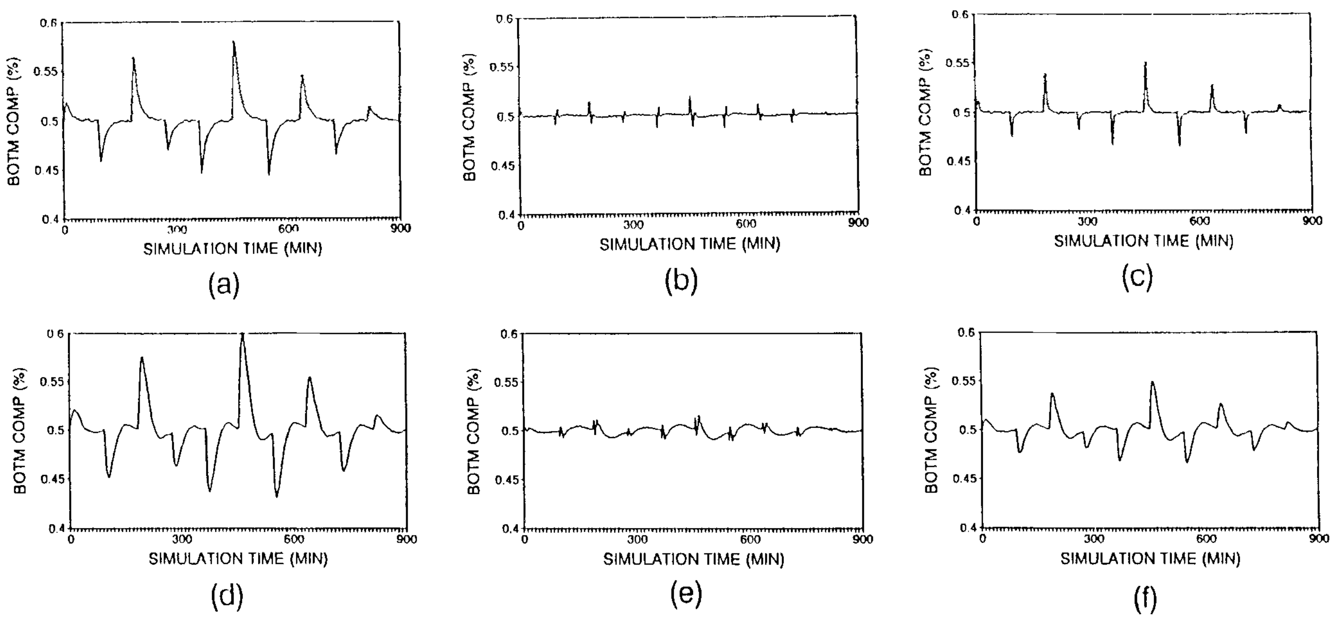

Figure 8. Output responses to unexperienced disturbance patterns: (a) DMC controller alone; (b) trained neural controller; (c) 50% damaged neural controller; (d) DMC controller alone (process/model mismatch); (e) trained neural controller (process/model mismatch); (f) feedforward DMC controller (process/model mismatch).

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} \frac{9.0e^{-2s}}{26.7s+1} & \frac{-14.9e^{-5s}}{41.0s+1} \\ \frac{4.6e^{-5s}}{20.9s+1} & \frac{-14.4e^{-4s}}{24.4s+1} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} + \begin{bmatrix} \frac{3.8e^{-8s}}{14.9s+1} \\ \frac{4.9e^{-3s}}{13.2s+1} \end{bmatrix} D \quad (15)$$

Gains have an error of 30%, time constants have an error of 60–100%, and the deadtime has an error of 30–100% with respect to those used in the dynamic matrix A.

Figures 8d, 8e and 8f show the bottom composition responses of the DMC controller alone, the neural controller, and the feedforward DMC controller, respectively, to the disturbances in Figure 4b in the face of the process/model mismatch. Note that the feedforward DMC controller was designed based on the exact disturbance model. The neural controller was trained in the face of the process/model mismatch.

As expected, the process/model mismatch results in the performance deterioration of the DMC controller (compare Figures 8d and 8a). This is the case in the neural controller, as shown in Figure 8e. In the neural controller case, however, the amount of the performance deterioration is not large at all in spite of the large process/model mismatch. Further, the performance of the neural controller is superior to that of the feedforward DMC controller (compare Figures 8f and 8e). Modeling errors distort the direction of the feedback gain matrix K in DMC and consequently decrease the learning efficiency of the neural controller in some degree. Nevertheless, the neural controller largely compensates for the process/model mismatch by minimizing ΔU_d through training. In contrast, the performance deterioration of the feedforward DMC controller is quite large as shown in Figure 8f. Although the feedforward DMC controller generally gives almost perfect disturbance rejection (not shown here) when the model is perfect, it is very sensitive to modeling errors. In contrast, the proposed neural controller is robust in the process/model mismatch in DMC.

Learning by random input patterns

In the original back-propagation algorithm (Rumelhart and McClelland, 1986), predetermined input patterns are repeated in the network until desired performance is obtained. In many real situations, however, it may not be easy to present predetermined disturbance patterns repeatedly. Therefore, it is desirable for the neural controller to learn by observing random disturbance patterns that can happen during the usual operation.

To investigate this type of learning capability, step changes in the setpoints and the disturbance were introduced randomly and simultaneously to the control system. Figure 9 compares ISE changes of the neural controller and the DMC controller. In this simulation, one training cycle was set at 320 min. The result shows that the neural controller gradually improves its performance even for the random training patterns. Although the simulation shows the promising result, it should be noted that in this type of learning mode, there is a potential problem due to unlearning: if the process is excited by the disturbances of one operating regime for long time, the network may forget what it has learned about the other operating regime or disturbance patterns, which has not been shown recently.

Application to Nonlinear Reactor Control

The nonlinearities of many chemical processes limit the application of the linear-model-based feedforward controller. To illustrate the effectiveness of the neural feedforward controller on the nonlinear system, the CSTR model by Williams and Otto (1960) was chosen. The CSTR system supports the following multiple reaction:

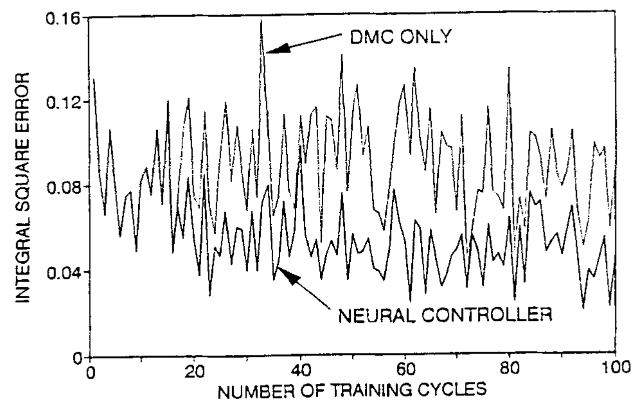

Figure 9. Changes of the ISE by the random disturbance patterns.

$$A+B\xrightarrow{k_1}C \tag{16}$$

$$C+B\xrightarrow{k_2}P+E \tag{17}$$

$$P+C\xrightarrow{k_3}G \tag{18}$$

Reactants A and B enter as pure components in separate streams with flow rates F_a and F_b, respectively. The equations describing the kinetic behavior of the above reactions and the dynamic mass balances for the CSTR represent a coupled set of nonlinear algebraic and differential equations as follows (see Williams and Otto, 1960, for details):

$$dx_a/dt=F_a/\rho V-k_1x_ax_b-x_aF_r/\rho V \tag{19}$$

$$dx_b/dt=F_b/\rho V-k_1x_ax_b-k_2x_bx_c-x_bF_r/\rho V \tag{20}$$

$$dx_c/dt=2k_1x_ax_b-2k_2x_bx_c-k_3x_cx_p-x_cF_r/\rho V \tag{21}$$

$$dx_e/dt=2k_2x_bx_c-x_eF_r/\rho V \tag{22}$$

$$dx_g/dt=1.5k_3x_cx_p-x_gF_r/\rho V \tag{23}$$

$$dx_p/dt=k_2x_bx_c-0.5k_3x_cx_p-x_pF_r/\rho V \tag{24}$$

where ρ is an average density, V denotes a reactor volume, and the reaction coefficients, k_1, k_2, and k_3, are:

$$k_1=5.9755(10^9)e^{-12{,}000}/T_r \tag{25}$$

$$k_2=2.5962(10^{12})e^{-15{,}000}/T_r \tag{26}$$

$$k_3=9.6283(10^{15})e^{-20{,}000}/T_r \tag{27}$$

Values of the parameters are the same as in Williams and Otto (1960). For the purposes of simulation, the feed rate F_a and the reactor temperature T_r are considered as manipulated variables, while F_b is considered as a disturbance variable, and the mass fractions of product C and G (x_c and x_g) are considered as controlled variables. The feasible steady-state values of the major variables were calculated solving the nonlinear algebraic equations of the system. The values of the major variables and parameters are listed in Table 1.

Table 1. Steady-State Values of Williams/Otto Reactor Model Variables

$T_r=621$ R	$F_a=3.4$ kg/s	$F_b=9.76$ kg/s,	
$x_a=0.1603$	$x_b=0.5684$	$x_c=0.0353$	$x_e=0.1501$
$x_g=0.0163$	$x_p=0.0696$	$\rho=800.9$ kg/m^3	$V=2.63$ m^3

The sampling time interval was 1 min. The tuning parameters of DMC were set to $P=20$, $N=5$, $\Lambda=0.001I$, and $\Gamma=I$. Step response coefficients for the dynamic matrix were obtained by perturbing the model by a unit step. The result of open-loop response analysis showed that this system had moderately nonlinear characteristics. The same network as in the previous example was employed. Training patterns lasting for 500 min, which included 10 consecutive random step changes in F_b in the range of $\pm$5kg/s, were repeated.

The disturbance rejection performance of the trained neural controller was compared with those of the DMC and feedforward DMC controllers under the same condition. The result is shown in Figure 10. The neural feedforward controller performs better than the feedforward DMC controller. The result shows that the disturbances take the system away from the nominal linear model, adversely affecting the performance of the linear feedforward DMC controller. In the neural controller case, the nonlinear architecture of the neural network allows the controller to capture the nonlinear mapping, and the adverse effect, due to the linearized process model, is compensated minimizing ΔU_d through training.

Conclusions

A new control technique using the neural network in feedforward control for unknown disturbances in the MPC scheme has been presented. This neural controller is relatively simple

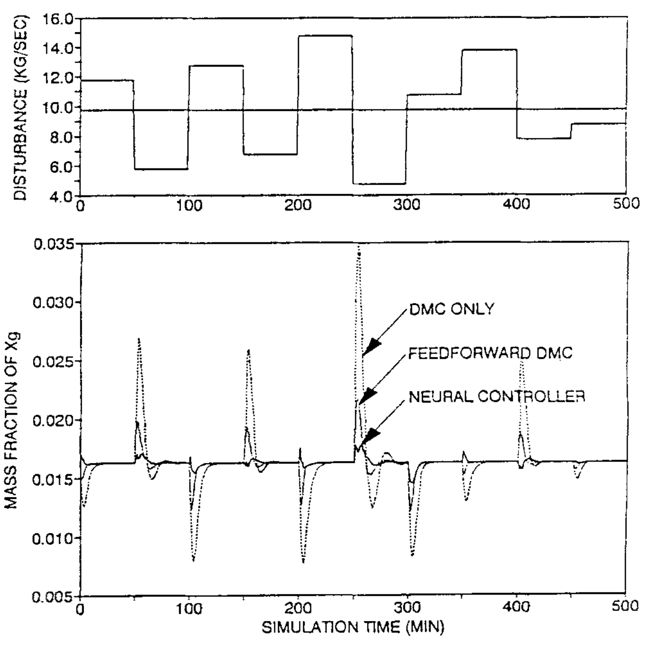

Figure 10. Output responses of the DMC controller, feedforward DMC controller, and trained neural controller.

in terms of concept and computation. The neural controller learns about the relationship between the disturbance patterns and the desired control actions by minimizing the MPC controller output due to the unmodeled effect through training. Once the training is completed, it does not require any iterative steps for solving nonlinear problems to produce control actions.

A well-known distillation column system and a reactor system were used to demonstrate the proposed neural controller. From the simulation, the following results can be drawn:

- Its generalization capability by interpolation allows it to maintain the performance level even for unexperienced disturbance patterns.
- It is robust for the faults in the connection weights.
- It can learn with the unrepeated random disturbance patterns during the usual operation.
- It is robust in the process/model mismatch in DMC in contrast to the feedforward DMC controller.
- It copes well with the nonlinear system.

Many areas require further research: stability and convergence of the neural controller must be analytically established; the local minimum problem, the learning time, and the optimal design of network architecture also have to be studied. Nevertheless, the proposed neural controller appears to have the potential to deal with unknown complex disturbances.

Acknowledgment

Financial support from the Yukong Limited, the Korea Science Foundation, and the Korea Ministry of Science and Technology is gratefully acknowledged.

Notation

A	=	dynamic matrix of the system
d	=	unmodeled factors
D	=	disturbance vector
e	=	predicted error vector
e_m, e_d	=	predicted error vectors due to modeled and unmodeled effect, respectively
$F(\cdot)$	=	nonlinear activation function
F_i	=	flow rate of i component, kg/s
K	=	feedback gain matrix given by Eq. 11
L	=	output layer number
N	=	control horizon
N^l	=	number of neurons in the lth layer
O^l_{pj}	=	output of the jth neuron in the lth layer for pattern p
P	=	prediction horizon
R	=	set-point vector
s	=	Laplace transformation variable
T_{pj}	=	target signal or output of the jth neuron in the output layer for the pattern p
T_r	=	reactor temperature, R
U	=	manipulated variable vector
U_c, U_n	=	control signal vectors from the DMC controller and the neural network, respectively
$\Delta U_m, \Delta U_d$	=	vectors for control input changes in DMC at different time intervals caused by modeled effect and unmodeled effect, respectively
ΔU_c	=	vector for control input change in DMC at different time intervals
V	=	reactor volume, m^3
W^l_{ij}	=	connection weight between the ith neuron in the $(l-1)$th layer and the jth neuron in the lth layer
x_i	=	mass fraction of i component
Y	=	controlled variable vector
$\hat{Y}^*$	=	output vector for contribution of past control inputs

Greek letters

ρ	=	average density, kg/m^3
η	=	learning rate coefficient
Γ	=	positive weighting matrix for selective weighting of controlled variables
Λ	=	positive weighting matrix for move suppression of manipulated variables

Literature Cited

Bhat, N., and T. J. McAvoy, "Use of Neural Nets for Dynamic Modeling and Control of Chemical Process Systems," *Comp. Chem. Eng.*, **14** (4/5), 573 (1990).

Brengel, D. D., and W. D. Seider, "Multistep Nonlinear Predictive Controller," *Ind. Eng. Chem. Res.*, **28**, 1812 (1989).

Cutler, C. R., and B. L. Ramaker, "Dynamic Matrix Control—a Computer Control Algorithm," AIChE Meeting, (Apr. 1979); *Joint Automatic Control Conf. Proc.*, San Francisco (1980).

Garcia, C. E., and M. Morari, "Internal Model Control: 2. Design Procedure for Multivariable Systems," *Ind. Eng. Chem. Proc. Des. Dev.*, **24**, 472 (1985).

Hernandez, E., and Y. Arkun, "Neural Network Modeling and an Extended DMC Algorithm to Control Nonlinear Systems," *Proc. Amer. Control Conf.*, 2454 (1990).

Kawato, M., Y. Uno, M. Isobe, and R. Suzuki, "Hierarchical Neural Network Model for Voluntary Movement with Application to Robotics," *IEEE Control Systems Mag.*, **8**, 8 (1988).

Lee, M., and S. Park, "Process Control Using a Neural Network," *Comp. Chem. Eng.*, submitted (1991).

Li, W. C., and L. T. Biegler, "Newton-Type Control Strategies for Constrained Nonlinear Processes," *Chem. Eng. Res. Dev.*, **67**, 563 (1989).

Patwardhan, A A., J. B. Rawlings, and T. F. Edgar, "Nonlinear Model Predictive Control," *Chem. Eng. Comm.*, **87**, 123 (1990).

Peterson, T., E. Hernandez, Y. Arkun, and F. J. Schork, "Nonlinear Predictive Control of a Semi Batch Polymerization Reaction by an Extended DMC," *Proc. Amer. Control Conf.*, 1534 (1989).

Psaltis, D., A. Sideris, and A. A. Yamamura, "A Multilayered Neural Network Controller," *IEEE Control Systems Mag.*, **8**, 17 (1988).

Psichogios, D. C., and L. H. Ungar, "Nonlinear Internal Model Control Using Neural Networks," *AIChE Meeting*, Chicago (1990).

Richalet, J., A. Rault, J. L. Testud, and J. Papon, "Model Predictive Heuristic Control: Application to Industrial Processes," *Automatica*, **14**, 414 (1978).

Roat, S., and C. F. Moore, "Applications of Neural Networks and Statistical Process Control to Model Predictive Control Schemes for the Chemical Process Industry," *AIChE Meeting*, Chicago (1990).

Rouhani, R., and R. K. Mehra, "Model Algorithmic Control (MAC): Basic Theoretical Properties," *Automatica*, **18**(4), 401 (1982).

Rumelhart, D. E., and G. E. Hinton, and R. J. Williams, "Learning Internal Representation by Error Propagation," *Distributed Parallel Processing*, Vol. 2, MIT Press, Cambridge (1986).

Setoyama, T., M. Kawato, and R. Suzuki, "Manipulator Control by Inverse-Dynamics Model Learned in Multilayer Neural Network," *Japan IEICE Tech. Report MBE87-135*, 249 (1988).

Williams, T. J., and R. E. Otto, "A Generalized Chemical Processing Model for the Investigation of Computer Control," *AIEE Trans.*, **79**, 458 (1960).

Wood, R. K., and M. W. Berry, "Terminal Composition Control of a Binary Distillation Column," *Chem. Eng. Sci.*, **28**, 1707 (1973).

Ydsite, B. E., "Forecasting and Control Using Adaptive Connectionist Networks," *Comp. Chem. Eng.*, **14**(4/5), 583 (1990).

Manuscript received Feb. 21, 1991, and revision received Oct. 28, 1991.