A NEURAL NETWORK STRATEGY FOR DISTURBANCE PATTERN CLASSIFICATION AND ADAPTIVE MULTIVARIABLE CONTROL

L. MEGAN† and D. J. COOPER‡

Chemical Engineering Department, University of Connecticut, U-222, Storrs, CT 06269-3222, U.S.A.

(Received 28 June 1993; final revision received 7 January 1994; received for publication 8 April 1994)

Abstract—This paper presents a neural network approach to adaptive control through pattern recognition techniques, extending previously published results on single-input/single-output systems to two-input/two-output systems. Two vector-quantizing neural networks are used to analyze both the input and output patterns resulting from a perturbation to the process. The results of these analyses are then used to update the model gain of the first-order plus dead time model that describes each input/output pair. This work focuses primarily on making model adaptations following load disturbances as opposed to set point changes, as load disturbances present by far the greatest adaptation challenge to chemical process applications. The results are compared to a more traditional modeling technique, batchwise model regression, with respect to both accuracy and computational load. The adaptive strategy is demonstrated using a variety of disturbances on two challenging multivariable process simulations.

INTRODUCTION

As most chemical processes are characterized by nonlinear and/or nonstationary behavior, those control strategies which incorporate an adaptable process model to describe the recent behavior of the system are among the most promising. However, in spite of the potential benefits, including improved safety and increased profitability, industry has been slow to accept model based adaptive control. A primary reason is the unwillingness of plant engineers and operators to abandon techniques proven to be successful. A properly tuned PID controller may not optimize performance, but it will usually maintain stable control over a wide range of operating conditions. Also, many plant practitioners are uncomfortable with the mathematical complexity inherent to the popular model adaptation methods such as recursive least squares (RLS).

For an adaptive controller to gain widespread industrial acceptance it must be stable and robust, simple to understand and implement, and applicable to a wide variety of processes. One promising adaptation strategy which has the potential to meet these requirements is based on pattern recognition (Bristol, 1977). The authors' initial pattern recognition work focused on a batchwise analysis of the process data following either a set-point change or disturbance to the system. In that work, features of the process transients such as overshoot and damping are used to indicate whether the data is dynamically rich in information, thereby being useful for model updating via batch regression (Cooper et al., 1990). Further extensions focused on using artificial neural networks (ANN) to rapidly analyze the error response to indicate the degree of model adaptation (Cooper et al., 1992a), thereby avoiding time consuming model regression techniques.

These methods have been proven successful on a wide range of single-input/single-output (SISO) simulations and pilot scale/mini systems and apply to a number of model based controller algorithms, including a model based PI controller and the Generalized Predictive Control framework of Clarke et al. (1987). The work presented here investigates the applicability of such techniques to two-input/two-output (2 × 2) processes. Adaptive control strategies for 2 × 2 systems have yet to receive the attention in the literature that SISO systems have received, and much of the work that has been done may have difficulty in finding acceptance in the chemical industry due to the mathematical complexity of the techniques.

This paper focuses on applying pattern recognition techniques to disturbance dynamics. From an adaptation standpoint, the most challenging perturbations to chemical systems are in the form of load disturbances. However, the most popular adaptive control strategies rely on dynamics caused by set point changes to update the models, where the model updating is typically done either through
batch regression or RLS (Seborg et al., 1986). Because load disturbances can potentially change process character, it would be valuable if any information contained in the disturbance induced dynamics could be utilized to update the controller model. However, the previously mentioned modeling techniques can produce suspect results in modeling such dynamics as the transient patterns may be a reflection of the disturbance dynamics and not a true representation of the process input/output relationship.

Previous work with pattern based disturbance analysis (Cooper et al., 1992b) has shown that a simultaneous analysis of the manipulated input and controller error patterns can lead to effective model gain adaptations in SISO systems. That work also demonstrated that ANNs, backpropagation networks (BPN) and vector quantizing networks (VQN), are effective tools for pattern analysis.

This paper extends these ideas to $2 \times 2$ nonlinear systems which are open loop stable, do not have multiple steady states or large variations in time delay and which are minimum phase. Extending these concepts to such systems is challenging since the degree of process interaction determines the adaptation decision associated with a particular pattern. What may be considered a good controller error pattern for a highly interacting system may be considered poor control in a process with less severe interaction. Whereas in the SISO work reasonably precise model parameter adaptations are made based on a given controller error pattern, the extension of pattern recognition methods to $2 \times 2$ systems is limited to a more qualitative level.

**Pattern recognition based adaptive control**

Generally speaking, in SISO systems the controller gain is inversely proportional to the process gain. Therefore, in model based strategies, if the estimated process model gain is too low, then the controller gain is too high. As with SISO systems, a highly underdamped response to a set-point change or disturbance within a model based $2 \times 2$ strategy is still used to infer that one or more of the model gains is too low, and conversely a highly overdamped response infers one or more is too high. If a reliable method is developed for decomposing the problem to find which model gains are mistuned, a qualitative evaluation of the model mismatch is possible. The model parameters can then be heuristically adjusted in a manner to improve performance. Such heuristic methods could be applied to adaptation following set-point changes. However, more accurate techniques such as batch regression or recursive modeling can be used to model set-point response data since the true dynamics of the system are reflected in the input/output history. The application of pattern recognition techniques to $2 \times 2$ systems is therefore best suited to disturbance dynamics.

For the model based adaptive control strategy discussed here, the four input/output relationships of a $2 \times 2$ system are described by a first-order plus dead time model (FOPDT). As an initial exploration of these ideas, this work focuses solely on model gain adaptation. Hence there are four process model gains to adapt. To allow individual adaptation of each of the four models, the model adaptation strategy uses both manipulated input and controlled output data, where the output data is expressed in terms of the controller error or deviation of the output from set-point.

The adaptation strategy proceeds as follows. A batchwise analysis of the patterns in the recent history of the manipulated input for each input/output pair following a disturbance is used to determine whether or not adaptation of the corresponding model gain is necessary. The decision is based on whether or not the disturbance has shifted the manipulated input form its previous steady state value, indicating a change in operating regime and a potential change in character due to process nonlinearity.

If a given input pattern indicates a potential change in process character, a subsequent analysis of each error pattern is used to identify the degree of mismatch between the process model gain and its actual value for each of the two input/output models corresponding to that input variable. Controller aggressiveness is generally inversely proportional to process gain, so that if the process model gain is too low, the controller action will be too aggressive. The degree of aggressiveness of each controller error pattern is categorized into one of a series of classes which allow for varying degrees of model gain adaptation. The given process model gain is then adjusted to move it closer to its true value.

**Controller implementation**

The manipulated input and controller error pattern analysis is performed via two separate ART2-A neural networks (Carpenter et al., 1991). These are vector quantizing networks which contain a series of exemplar patterns in each "node" and classify a given input pattern as the most similar network exemplar pattern. Upon classification, the model mismatch associated with that node is sent to a decision maker which combines the input analyses from the first network with the error analyses from the second network to determine appropriate model gain adaptations.
The ideas developed here are general in nature and should be applicable to any model based controller. This work demonstrates the methods using the least squares formulation of dynamic matrix control (Prett and Garcia, 1988; Cutler and Ramaker, 1979). Open-loop step response models are used in which data for each input/output pair are fit with a FOPDT model, and the FOPDT differential equation is then integrated to get the model parameters used in DMC. Following model gain adaptation in the Decision Maker, the step response models used in DMC are regenerated using the new FOPDT model expression.

The adaptive strategy is demonstrated on two process simulations. The first is the transfer function model of Wood and Berry (1973). The adaptive strategy is demonstrated by initially mistuning the model gains from the true process values, thereby giving the adaptive strategy the opportunity to adjust the model gains closer to their expected values. The second simulation is a rigorous distillation column, modeled after that of McCune and Gallier (1973). This process contains strong interaction among the variables and is highly nonlinear, thus demonstrating the adaptive strategy on a more challenging chemical process control problem.

It should be stressed that the objective of this work is on the development of a general adaptive strategy and not one specifically suited for distillation. Thus while the presented adaptive strategy could be improved on a process-to-process basis, no attempt to do so is made as such specifications would limit the general applicability of the techniques. The fact that the two process demonstrations are distillation problems is merely a result of the readily available documentation for 2×2 distillation systems.

For comparison to the ANN method, results for both regulatory and servo control are compared to those obtained from model updating via batchwise regression techniques. In this case, the dynamic data immediately following a set point change or disturbance to the system are used to regress the standard single-output/two-input model that describes each output variable, where again a FOPDT model is used to describe each input/output relationship. The results from the regression techniques are compared to the ANN results with respect to both accuracy and computational load.

**BACKGROUND**

*Dynamic matrix control*

Dynamic matrix control has been well documented in the literature so this discussion focuses only on the aspects directly related to this work. The least squares formulation of DMC is used here, where a series of unconstrained future manipulated input moves are calculated to minimize the following objective function at sample $t$:

$$
J = Q_1 \sum_{i=1}^{P} [y_{e}(t+i) - y^*(t+i)]^2 + Q_2 \sum_{i=1}^{C} [\Delta u(t + i - 1)]^2,
$$

where $y^*(t+i)$ and $y_{e}(t+i)$ are the vector of predicted outputs and future set points, respectively, and are of dimension $2P$, where $P$ is the prediction horizon. The first $P$ elements of $y^*$ are predicted outputs of the first output variable $y_1$, while the second $P$ elements are the predictions of the second output $y_2$. $\Delta u(t + i - 1)$ is the vector of future manipulated variable moves, and is of dimension $2C$, where $C$ is the control horizon. Similarly to the output vector, the first $C$ elements of $\Delta u$ correspond to $u_1$ moves while the second $C$ elements correspond to $u_2$ moves.

$Q_1$ and $Q_2$ are diagonal weighting function matrices whose dimensions are $[2P, 2P]$ and $[2C, 2C]$ respectively. In this work, the diagonal elements of $Q_1$ are equal to one, so that both controlled variables are weighted equally. The diagonal elements of $Q_2$ are known as input suppression factors and are defined as functions of the diagonal elements of the model gain matrix, where the input/output pairings are chosen in accordance with the relative gain array (Bristol, 1966). Thus the diagonal elements of $Q_2$ are defined:

$$
Q_2 = \Gamma \times K_{pu}^2
$$

where $K_{pu}$ is equal to $K_{pi}$ for the first $C$ elements and $K_{p2}$ for the second $C$ elements. $\Gamma$ is chosen to give user specified desired performance in the initial operating regime. This formulation thus defines the penalties on manipulated input moves as a function of the process model while keeping the two terms of equation (1) consistent in units and proportionate in size.

The least squares solution of DMC minimizes equation (1) over $C$ future control actions of both manipulated variables to give:

$$
\Delta u = (A^T A + Q_2)^{-1} A^T [y_{e}(t+i) - y^*(t+i)],
$$

$l = 1, 2P$.  

(3)

The step response model matrix $A$ contains each of the four step response models for each input/output pair. The first control move of each variable, $\Delta u(1)$ and $\Delta u(C+1)$, is implemented and the procedure repeated at each sampling instant. Note that upon
model adaptation, both the $A$ matrix and $Q_2$ are reset to reflect the updated model parameters.

In this work, the elements of the model matrix $A$ are initially developed by first regressing a FOPDT model on the open loop step response of each input/output pair. The elements of the $A$ matrix are then generated by differentiating the FOPDT equation:

$$ \tau_p \frac{dy(t)}{dt} + y(t) = K_{p_u} u(t - t_0), \quad (4) $$

where $K_{p_u}$ is the model gain for the given input/output pair, $\tau_p$ is the model time constant and $t_0$ is the model dead time. Online, a new solution for equation (4) is found each time $K_{p_u}$ is updated.

To aid in keeping the adaptive strategy process independent, the controller parameters are defined as functions of the minimum and maximum time constant of each system, as opposed to being set to constant values. This is done as follows:

$$ \Delta t = A \tau_p^{\min}, $$

$$ P = (B \tau_p^{\max}) / \Delta t, $$

$$ C = CP, \quad (5) $$

where $\Delta t$ is the sample time and $\tau_p^{\min}$ and $\tau_p^{\max}$ are the minimum and maximum of the four model time constants, respectively. Throughout this work, $A = 0.2$, $B = 5.0$ and $C = 0.25$.

ART2-A neural networks

Vector-quantizing neural networks have been shown to be effective tools for the pattern classification task at hand (Cooper et al., 1992a). VQNs are composed of a series of nodes, each of which contains a specific exemplar pattern. These exemplar patterns are then used to encode and classify incoming patterns online. This work uses the ART2-A vector-quantizing network, which is a simplification of the ART2 network of Carpenter and Grossberg (1987).

The ART2-A network is well-suited for situations where a large number of potential patterns exist, and the goal is to cluster similar patterns to reduce the number of pattern classes to a workable number. In the case of process systems, for example, there exists a limitless number of potential controller error patterns, and the goal of the network is to cluster similar patterns into a moderately small library of exemplar patterns that still has satisfactory classification abilities.

The ART2-A groups similar groups into exemplar patterns and stores these patterns in its nodes. All the patterns in this work are data histories in either the manipulated input or controller error which contain the first 50 data samples following the onset of a disturbance to the system, where the disturbance is recognized by the controlled output deviating from the set-point with respect to an appropriate noise band.

The training of the network, where the initially empty nodes are subsequently filled with exemplar patterns, proceeds in the following manner. The user initially defines a number of nodes, which represents the maximum number of classifications. All nodes are initially uncommitted, which is to say they represent no patterns.

The training patterns $I$ are first normalized:

$$ \eta^i = \frac{I^i}{||I^i||}, \quad (6) $$

where $|| \cdot ||$ signifies the norm (Euclidean distance) and the operator $\eta$ performs Euclidean normalization. In order to minimize the effect of noise, thereby improving the performance of the algorithm, the normalized input pattern is then filtered:

$$ (\mathcal{F}_\eta)(I^i) = \begin{cases} 
\eta^i, & \text{if } (\eta^i) > \Theta, \\
0, & \text{otherwise}, 
\end{cases} \quad (7) $$

where $i$ represents each element of the input vector and $\Theta$ is a threshold value, defined as:

$$ 0 \leq \Theta \leq \frac{1}{M^{0.5}} \quad (8) $$

$M$ is the length of the input vector, and is equal to 50 throughout this work. The filtered input pattern is then renormalized which results in the enhancement of larger values and the attenuation of smaller values. The final network input thus takes the form:

$$ I = \eta \mathcal{F}_\eta I^i. \quad (9) $$

The next step is to calculate the matching scores, which represent how closely each input pattern corresponds to the exemplar patterns. The matching score is calculated as:

$$ T_j = \begin{cases} 
\alpha \sum_i (I_i) & \text{if } j \text{ is an uncommitted node,} \\
I_z_j & \text{if } j \text{ is a committed node}, 
\end{cases} \quad (10) $$

where

$$ \alpha \leq \frac{1}{M^{0.5}}. \quad (11) $$

Each vector $\tau_j^*$ is the exemplar pattern for each node $j$. Committed nodes are those which contain a pattern representation.

The "winning" node is:

$$ T_j = \max_i (T_i). \quad (12) $$
However, although node \( J \) is the best match, it may not be a very close match. Thus, before the input pattern is committed to the winning node, it must pass a vigilance test. Node \( J \) is selected as the winning node if \( Y_J \geq \rho \), where \( \rho \) is a positive constant between zero and one. If \( T_J < \rho \), then the input pattern is committed to the next uncommitted node. If all the nodes are committed, then the input pattern is discarded and the algorithm moves on to the next training pattern.

Learning in the committed nodes is performed as long as \( J \) is committed or there are available uncommitted nodes. The exemplar pattern for node \( J \), \( z_J^{(e)} \), is updated as:

\[
\begin{align*}
I & \quad \text{if } J \text{ is an uncommitted node}, \\
\tau_t^{(e)} & = \eta(\beta T J + (1 - \beta) z_J^{(e)(0)}) \\
& \quad \text{if } J \text{ is a committed node},
\end{align*}
\]

(13)

where

\[
\Psi_I = \begin{cases} 
I, & \text{if } z_I > \Theta, \\
0, & \text{otherwise}.
\end{cases}
\]

(14)

This insures that no new information is learned where there previously was none. The learning parameter \( \beta \) is set between 0 and 1 and determines to what degree each training pattern contributes to the exemplar pattern of the associated winning node. In this work, new training patterns are continuously generated so the network does not see repeating presentations of the same training set, as would happen in BPN training. Thus, convergence for the ART2-A network is complete once the network exemplar patterns stabilize within a given tolerance and either no new nodes are being committed or the maximum number of nodes has been filled.

Once trained and put online, the incoming patterns are normalized by equations (6–9) and the matching scores are calculated by:

\[
T_J = I \tau_J.
\]

(15)

The winning node is the one which satisfies equation (12). If \( T_J \) is greater than the vigilance parameter \( \rho \) the classification that corresponds to that winning node is used in the model adaptation strategy. Otherwise, the pattern is ignored and no changes are made to the model based on a lack of information to make a reliable classification.

For this work, the parameter values are set as follows. \( \Theta \) is set equal to 0.0 as the pattern classification requirements of the network are not precise enough to be affected by small amounts of noise. This work uses very general classification of both input and error patterns. If more precise pattern classification was required then larger values of \( \Theta \) would be needed to provide additional noise filtering. For similar reasons, \( \rho \) equals 0.75, which by ART2-A standards is a relatively low value for the vigilance factor. Higher values of \( \rho \) would result in more patterns being rejected on-line. However, in this work the network exemplars only provide a general pattern classification and do not have to be an exact match to the incoming patterns.

\( \beta \) is set to 0.1 which implies that new training patterns have only a small effect on the established network exemplar. The value is similar to values the authors have found to be effective in implementing first-order filters, which are similar mathematically to equation (13). The authors have found, however, that varying \( \beta \) has little effect for the pattern classification task at hand. \( \alpha \) equals \( 1/(M)^{0.5} \), which is the standard specification of Carpenter. Finally, the length of all the input patterns is \( M = 50 \), which from equation (5) is equal to \( 10 \tau_r^{(m)} \). Higher values of \( M \) would allow the network to see a larger portion of the dynamic pattern but at the expense of more computation and a longer time before pattern classification and model adaptation.

MODEL ADAPTATION

The biggest obstacles to be overcome by any multivariable control algorithm, including pattern recognition methods, are the variable interactions. Without the interaction between all or some of the process variables, the control system can be reduced to independent SISO loops. In the authors’ work with pattern recognition based adaptation in SISO systems, the underlying principle is a communality of controller error patterns from process to process based on the multiplicative mismatch between the process and model parameters. This allows an adaptation strategy based on patterns developed from an idealized process to be applicable to a wide range of actual processes. The result is a quantitative assessment of process–model parameter mismatch which is then used to make precise adaptations to the model parameters (Cooper et al., 1992a).

As mentioned in the introduction, the difficulty in extending this idea to \( 2 \times 2 \) systems is that there is no longer a single set of idealized patterns, but rather a changing set of patterns as a function of the degree of interaction. What may be considered a good controller error pattern for a highly interacting system may be considered poor control in a process with less severe interaction. The result is that the extension of pattern recognition methods to \( 2 \times 2 \) systems is limited to a more qualitative rather than quantitative level. A highly underdamped response to a set-point change or disturbance within a model-
based 2 × 2 strategy is still used to infer that one or more of the model gains is too low, though it may not be immediately clear which model parameters are incorrect. A similar idea holds for a highly overdamped response. Thus if a reliable method can be developed for decomposing the problem to find which model gains are mistuned, then at least a qualitative assessment of the model mismatch can be made. This can then be used to heuristically adjust the model parameters such that the updated parameters offer a better representation of the true process. Since disturbance dynamics may mask the true input/output relationship, a model developed via regression or least squares techniques may give erroneous results. Therefore, any model adaptation is limited to heuristic methods, making pattern recognition a viable approach.

**Neural network pattern analysis**

The adaptation strategy consists of two ART2-A neural networks and a Decision Maker which determines model gain adaptations based on the network classifications. The system is pictured in Fig. 1, where one network is used to analyze and classify the input patterns, while the second is for the error patterns. The overall process model being adapted consists of the two equations that describe the output variables, which in the Laplace domain are of the form:

\[
y_i(s) = \frac{K_{p1} e^{-Tv_i}}{\tau_p s + 1} u_i(s) + \frac{K_{p2} e^{-Tv_i}}{\tau_p s + 1} u_c(s); \quad i = 1, 2 \quad (16)
\]

Thus there are four variables, the four process model gains, that can be potentially adapted following a perturbation to the system.

The ANN based adaptation strategy is triggered by the occurrence of a disturbance to the system, which is defined as a significant deviation of either controlled output from the noise band in the absence of a recent set-point change. Once the algorithm determines a disturbance is occurring, it collects the next 50 samples of manipulated input and controller error data and uses these patterns for analysis, where the sample rate is determined by equation (5).

The first step in the adaptation process is the analysis of the manipulated input patterns. The input patterns are used to determine whether or not a disturbance to the system has potentially changed the process operating regime and thus the character of the process. For example, a step disturbance will shift the manipulated input away from its mean value that existed prior to the onset of the disturbance. Thus the process has changed operating regime, potentially changing the process character due to nonlinearity in the process.

Conversely, a pulse or oscillating disturbance will not permanently change the character of the process, so although the error patterns resulting from such a disturbance may look underdamped, no model adaptation is necessary. This oscillatory component will be reflected in the manipulated input patterns, which should not significantly deviate from their local operating regime as the process character is not permanently changed. This information can be used to infer that reliable model adaptation cannot be made based on the given error patterns.

Thus the goal of the input classification network is to classify each manipulated input response to the disturbance into one of two categories: (1) the input patterns have a step component showing significant deviation from the steady state that existed prior to
the onset of the disturbance, which reflects a change in process character and the need for analysis of the controller error patterns; or (2) the input patterns are remaining local to the mean value which existed prior to the disturbance and thus no model adaptation is necessary.

If the pattern analysis for a given input pattern signifies a potential change in character, the second network is activated to analyze the controller error responses. Recall that the controller error is defined as:

$$e_i(t) = y_{sp}(t) - y_i(t).$$  \hfill (17)

at sample $t$, where $i = 1, 2$ in this work. The purpose of the error network is to classify the degree of oscillation in the error patterns and determine how much a given model gain should be adapted based on its associated error pattern. For a $2 \times 2$ system, $K_{p11}$ and $K_{p22}$ are associated with the error pattern of $y_1$, while $K_{p12}$ and $K_{p21}$ are associated with $y_2$. Thus $K_{p11}$ and $K_{p12}$ adaptations are based on the $e_1$ pattern, while $K_{p21}$ and $K_{p22}$ adaptations are based on the $e_2$ pattern. This work takes a qualitative approach by assigning the error pattern exemplars in the network to one of five classifications, based on the degree of damping. The appropriate model gain adjustment $\delta_{eq}$ is then based on the given class. The classifications and their associated adjustment factors are given in Table 1. The output of the error network for a given error pattern is one of the five classes, which is then translated to the appropriate gain adjustment. An important feature of this approach is that by focusing on mismatch ratios as opposed to actual values of the gain, the strategy remains independent of the given process.

The decision maker

While processes with significant interaction will certainly have an effect on the error responses, a combinatorial analysis of the input and error patterns can lead to a reasonable classification for each of the four model gains. The adaptation strategy as implemented in the decision maker works in the following manner:

Step 1 Analyze first input pattern $u_i$, for shift in steady state, signifying potential change in process character.

Table 1. Error network pattern classes and corresponding model gain adjustments

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>$\delta_{eq}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Very underdamped</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>Underdamped</td>
<td>1.50</td>
</tr>
<tr>
<td>3</td>
<td>Acceptable</td>
<td>1.00</td>
</tr>
<tr>
<td>4</td>
<td>Overdamped</td>
<td>0.67</td>
</tr>
<tr>
<td>5</td>
<td>Very overdamped</td>
<td>0.50</td>
</tr>
</tbody>
</table>

Step 2a If shift in input steady state, error network classifies $e_i$ and adapts $K_{p11}$ based on $\delta_{11}$, since $K_{p11}$ is the model gain which relates $e_i(y_i)$ to $u_i$. The error network then classifies $e_i$ and adapts $K_{p21}$ based on $\delta_{21}$. The adaptation is done according to the formula:

$$K_{p21}(t) = K_{p21}(t-1) \times \delta_{eq}$$  \hfill (18)

Step 2b If input network determines $u_i$ has not shifted from steady state, $K_{p11}$ and $K_{p21}$ remain unchanged and no analysis of the error patterns is necessary.

Step 3 Analyze second input pattern $u_i$ for shift in input steady state. Then repeat Step 2, where, if necessary, $K_{p12}$ is adapted based on $e_1$ and $K_{p22}$ is adapted based on $e_2$.

Step 4 Using new model parameters, update DMC model parameter matrix $A$ as described in the DMC section.

Network training

The development of training patterns for both networks is a continuous process of randomly generating a training pattern and then feeding it to the network for classification. In developing the training patterns it is important to remember that the goal of this work is an adaptation strategy that is process independent. The only limitation is that the process input/output relationships can be reasonably described with a FOPDT model. As a result, there are an endless number of possible combinations of the four process transfer functions and the training set must take this into account.

With this idea in mind, the training set is developed in the following manner. The first step is to randomly select parameters for four process models, the parameters being $K_p$, $\tau_p$, and $\theta_d$. These models are then simulated as second-order plus dead time processes, which is the lowest order process that can be representative of higher order processes. Note that the networks are trained only once, using an idealized process, and do not have to be retrained on a process-to-process basis.

To give as accurate a representation of the potential patterns as possible, all the training patterns are developed in closed loop with the appropriate controller. Thus the next step in the development of the training patterns, following the selection of process models, is to generate the DMC model matrix $A$. This is done as described in the DMC section, where the FOPDT differential equation of each transfer function is solved for a step change in input. The DMC parameters are then defined as discussed in the DMC section.
For the adaptive strategy to be effective, a wide variety of patterns are needed to represent varying degrees of process/model gain mismatch. Thus for the selected set of process models $G_p$, the process simulation gains are then randomly mistuned from the model values and a unique disturbance is randomly generated and applied to the process simulation.

In this work, this is done ten times for each $G_p$. The disturbance transients are generated by randomly selecting various degrees of ramping, delay and oscillation over a 50 sample window, then forcing this disturbance on the process simulation. The resulting input and error patterns are then fed to the respective networks for training. The entire procedure is then repeated with a new $G_p$. After a number of iterations, which include a wide variety of model combinations and mistunings, the final result is a wide range of patterns which reflect varying characteristics of process/model mismatch for a variety of disturbances.

As new input and error patterns are generated, each respective network clusters similar patterns in its nodes. Training is complete for a given network once it ceases to create new exemplar patterns for a significant number of pattern presentations. Each net took approx 10,000 pattern presentations to train, which takes about 30 min on a 486/33 PC using Microsoft Fortran Version 5.1 as the compiler.

The resulting manipulated input classification network contains 40 unique patterns, each of which is then classified based on whether the pattern signifies a shift in the input's steady state value. Figure 2a shows a subset of the input network exemplar patterns. The top row of patterns are examples of those classified as those changing operating regime, indicating a potential change in process character. These patterns tell the Decision Maker to make a subsequent analysis of the error patterns. The bottom two rows show patterns classified as not having the potential for changing process character, so no check of the error patterns is necessary. For a pattern to be classified as shifting steady state it must clearly show a deviation from its previous steady state. For example, although the middle pattern in the second row of Fig. 2a seems to be deviating from its initial steady state, it is not definitely clear that this is the case and so it is classified as a pattern with no shift in steady state. This approach is chosen to insure that the strategy only makes decisions with complete confidence.

The training method results in a error classification network that contains 105 patterns, each of which is then classified into one of the five categories given in Table 1. A subset of the error network exemplars and their corresponding classifications is shown in Fig. 2b, with the most highly underdamped patterns in the top row and the most overdamped patterns in the bottom row. Due to the lack of dynamics in the overdamped patterns, there are fewer unique Class 4 and 5 exemplar nodes in comparison to the other three classes. For the bottom row of patterns in Fig. 2b, the first two patterns are Class 4 patterns while the third one is a Class 5 pattern.

**Batchwise regression**

As discussed in the introduction, the ANN strategy will be compared to a more traditional modeling technique, batchwise regression. The batchwise regression routine works in the following manner. Following a set point change or disturbance to the system, the next 50 samples of both input and both output variables are collected. The two models
given by equation (16) are then regressed on the data. Each model regression is a standard multidimensional minimization problem where the objective function to be minimized is:

$$J = \sum_{i=1}^{m} [y_p(t) - y'(t)]^2,$$

(19)

for $i = 1, 2$, where $y_p(t)$ is the predicted output based on the model parameters. The minimization technique is a typical two step multidimensional minimization procedure. Given an initial parameter vector $x^0$ and an initial search direction $s^0$, the objective function is minimized along the initial search direction, after which a new search direction is chosen and the procedure repeated until convergence. This work uses the BFGS technique to calculate the new search direction (Edgar and Himmelblau, 1988). This is a secant method which uses first-order information to obtain a new approximation of the Hessian matrix at every iteration, which is then used to calculate the new search direction by:

$$s^k = -(H^k)^{-1} \nabla f(x^k),$$

(20)

where $\nabla f(x^k)$ is the value of the gradient and $k$ represents the present iteration. Upon selection of a new search direction, a one dimensional golden section is used to find the step length $\lambda$ which gives the new estimate of the parameter vector that minimizes the objective function along the given search direction according to:

$$x^{k+1} = x^k + \lambda s^k.$$  

(21)

This procedure is done separately for each of the two equations described in equation (16). The minimization routine for a given model is terminated when both the change in parameter values and the change in the objective function fall below a specified tolerance at a given iteration. The routine is set up such that the parameter vector for a given model fit can be between two and six parameters, where a two parameter fit typically involves finding new values for the two process gains, $K_{P1}$ and $K_{P2}$, while a six parameter fit finds new values for all six parameters in a given model equation.

**SIMULATION DEMONSTRATIONS**

**Wood and Berry transfer function model**

The first demonstration process is the well-known Wood and Berry (1973) transfer function representation of a $2 \times 2$ distillation column. The simulation consists of the six linear FOPDT transfer functions given in Table 2. The control objective is dual control of the distillate composition $x_D$ and bottoms composition $x_B$. The manipulated variables are the reflux rate $R$ and steam rate $Q$, with changes in feed

**CLASS**

1

2

3

4–5

Fig. 2b. Subset of error network exemplar patterns.
rate acting as an unmeasured disturbance. The process simulation and control parameters are given in Table 3. The DMC parameters are calculated using the guidelines in the DMC section.

While this process is linear, it is a useful demonstration as it allows the model gains to be mistuned from the actual values in the process, thereby allowing the adaptive algorithm to adjust the model gains to improve performance. Since the true process values are known, it gives an opportunity to compare the values determined by the adaptation strategy to the true values.

The first element of the relative gain array (RGA) for this process, $\lambda_1$, is equal to 2.01, indicating significant interaction between the variables. To further challenge the adaptive strategy, the system is corrupted with a small amount of random error to simulate measurement noise. The standard deviation of the error, is set equal to 0.05. The network exemplar patterns are noise-free, and thus this demonstrates the networks’ abilities to classify patterns corrupted with noise.

The first two demonstrations (Figs 3 and 4) compare the performance between the ANN method and batchwise regression for model adaptation following set point changes. Although the ANN method is designed more towards disturbance adaptation, Figs 3 and 4 offer a good comparison in performance between the two techniques. In both cases, the four process model gains are initially mistuned to 50% of the four process gains given in Table 2 such that the initial model gain vector, $[K_{p11}, K_{p12}, K_{p21}, K_{p22}]$, is $[6.40, -9.45, 3.30, -9.70]$. Two set-point changes are made to the distillate composition, at sample 10 and sample 100. As can be seen in both figures, the process/model mismatch causes poor control in both $x_0$ and $x_b$ following the first set point change in $x_0$.

The ANN method, shown in Fig. 3, works in the following manner. Following the set-point change at sample 10, the analysis of the first input pattern $u_1$,

Table 3. Wood and Berry simulation parameters

<table>
<thead>
<tr>
<th>Steady state parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0 = 96.0%$</td>
</tr>
<tr>
<td>$x_b = 0.5%$</td>
</tr>
<tr>
<td>Reflux = 1.95 lb/min</td>
</tr>
<tr>
<td>Steam = 1.71 lb/min</td>
</tr>
<tr>
<td>Feed = 2.45 lb/min</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMC parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t = 2.2$ min</td>
</tr>
<tr>
<td>$P = 48$</td>
</tr>
<tr>
<td>$C = 12$</td>
</tr>
<tr>
<td>$\Gamma = 1.0$</td>
</tr>
</tbody>
</table>

Fig. 4. Batch regression based adaptive control of: (a) $x_0$; and (b) $x_b$ in Wood and Berry simulation with process/model mismatch.
signifies a shift in steady state. The error network classification of the error patterns signifies that both error patterns are Class 2 patterns as given in Table 1. The two model gains associated with $u_{i}$, $K_{p1}$, and $K_{p2}$, are then increased by a factor of 1.5, as described in Step 2a of the Decision Maker section. Similarly, $u_{i}$ is also classified as having a shift in steady state, and therefore $K_{p1}$ and $K_{p2}$ are also adjusted by 1.5 and the controller model updated appropriately. The new process model gain vector is thus [9.60, -14.37, 4.95, -14.55], compared to the true values given in Table 2 of [12.80, -18.90, 6.60, -19.40]. Although the model is not exact, it is a better representation of the process, resulting in improved performance for both variables at the next set-point change.

Figure 4 shows the results for the same situation using batchwise regression. In this demonstration, the model time constants and dead times are fixed since it is known in advance that only the gains are mistuned. Thus each regression consists of fitting two parameters, the model gains, to the data. Following the model fit at sample 60, the regression routine converges on a new $K_{p}$ vector of [13.21, -18.46, 7.27, -19.50], which is very close to the true values of [12.80, -18.90, 6.60, -19.40]. This is reflected in the response to the following set-point change in $x_{f}$ at sample 100, as shown in Fig. 4. A comparison of Fig. 4 to Fig. 3 shows that the regression routine gives better performance in making adaptations to set-point changes. Although the ANN performance is not as good as regression in this case, it offers comparable performance without requiring nearly the computational expense. On a 486/33 PC using the Microsoft Fortran compiler, the ANN identification method works within a few seconds while the regression routine is at least an order of magnitude slower in computing time.

While the ANN method is applicable to set point dynamics, its real strength is when it is applied to disturbance dynamics, as demonstrated in Fig. 5. As with Fig. 3, the model gains are initially mistuned to 50% of their true values. The two set point changes in $x_{p}$, at sample 10 and sample 100, show highly underdamped responses in both variables as a result of the mistuning. In this case, no adaptation is made following set-point changes so that the performance on disturbance dynamics can be isolated. The disturbance, a 25% change in feed rate at sample 200, also results in a highly underdamped response. While there is an oscillatory component to the input patterns, the input pattern network classifies both $u_{i}$ and $u_{o}$ as having a dominant step characteristic, thus triggering the error pattern network. The error pattern corresponding to $x_{p}$, $e_{1}$, is a Class 1 pattern and thus $K_{p1}$ and $K_{p2}$ are doubled from their present values. The error pattern corresponding to $x_{o}$, $e_{2}$, is classified as a Class 2 pattern, and thus $K_{p2}$ and $K_{p2}$ are increased by 1.5 times their present values. Thus while $K_{p1}$ and $K_{p2}$ are not completely adapted to the desired values, they are moved closer to the actual process gains. The improvement in performance resulting from this adaptation is demonstrated by the last two set point changes, at samples 300 and 400.

To better illustrate how the ANN method works, Fig. 6a shows the input network analysis for $u_{i}$ while Fig. 6b shows the error network analysis of $e_{2}$ for the disturbance in Fig. 5. The solid line in Fig. 6a shows the actual Euclidean normalized input pattern of $u_{i}$, while the dashed line shows the winning input network exemplar pattern, that is, the one with the highest matching score. This exemplar pattern gives a matching score of 0.979 and is classified as signifying a shift in steady state. Figure 6b shows the result of the error network analysis for $e_{2}$. The solid line is the normalized error pattern while the dashed line shows the winning error network pattern, which has a matching score of 0.948. As previously mentioned, this particular network exemplar pattern is a Class 2 pattern leading to the increase in $K_{p2}$ and $K_{p2}$ by 1.5.

For comparison, Fig. 7 shows the performance of the regression routine in trying to model the disturbance. As previously mentioned, it is difficult to
model disturbance induced dynamics and this is reflected in Fig. 7. The poor model fits give very poor behavior for the remainder of the trajectory following the model fit at sample 250. In such a situation, although the ANN approach can only make qualitative designations of the data patterns, it still allows useful model adaptation to take place.

Rigorous distillation model

The second demonstration simulation is a tray-by-tray distillation column model. While the Wood and Berry demonstration is only a system of linear transfer functions, the distillation column simulation reflects the highly nonlinear characteristics and strong interactive quality of a real process. The system, is modeled after that of McCune and Gallier (1973). For each tray, along with the reboiler and condenser, the model includes differential equations to describe the overall and component mass balances, along with an algebraic expression for the energy balance. Stage efficiencies are represented with a Murphree tray efficiency, while a linear hydraulic relationship is used to represent the liquid flow from each tray as a function of the mass holdup. Constant pressure is assumed in the column so that tabulated equilibrium data can be used. The system is benzene–toluene at 1 atm. Other assumptions in the model include negligible vapor phase holdup, perfect mixing on each plate, saturated feed and negligible heat loss from the column. The parameters of the model are given in Table 4. For more details on the development of the model, see the paper of McCune and Gallier.

The control variables are the benzene composition in the distillate stream $x_d$ and in the bottoms stream $x_b$. The manipulated inputs are the reflux rate and the heat input to the reboiler. There are also PI controllers to control the two level loops: the reflux drum via the distillate rate, and the bottoms level via the bottoms flowrate. However, the dynamics of these two loops are negligible and thus do

![Graph](image)

Fig. 6. (a) Normalized input pattern for steam rate; and (b) normalized error pattern for $x_b$ in response to step disturbance in Fig. 5.

![Graph](image)

Fig. 7. Batch regression based adaptive control of: (a) $x_d$; and (b) $x_b$ in Wood and Berry simulation with process/model mismatch in response to step disturbance.

<table>
<thead>
<tr>
<th>Table 4. Rigorous distillation simulation parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Column parameters:</strong></td>
</tr>
<tr>
<td>Trays = 16</td>
</tr>
<tr>
<td>Feed tray = 8</td>
</tr>
<tr>
<td>Tray efficiency = 0.60</td>
</tr>
<tr>
<td>Pressure = 1 atm</td>
</tr>
<tr>
<td><strong>Steady state parameters:</strong></td>
</tr>
<tr>
<td>$x_d = 94.5%$</td>
</tr>
<tr>
<td>$x_b = 2.6%$</td>
</tr>
<tr>
<td>$x_b = 50.0%$</td>
</tr>
<tr>
<td>Feed = 7.0 kgmol/min</td>
</tr>
<tr>
<td>Reflux = 10.5 kgmol/min</td>
</tr>
<tr>
<td>Heat input = 441,000 kJ/min</td>
</tr>
<tr>
<td>Reflux ratio = 3.0</td>
</tr>
<tr>
<td><strong>DMC parameters:</strong></td>
</tr>
<tr>
<td>$\Delta t = 10.85$ min</td>
</tr>
<tr>
<td>$P = 32$</td>
</tr>
<tr>
<td>$C = 8$</td>
</tr>
<tr>
<td>$\tau = 0.1$</td>
</tr>
</tbody>
</table>
not have a significant impact on the performance of the composition controller. The DMC tuning parameters for the composition controller are also given in Table 4. As with the previous demonstration, the output is corrupted with random error to simulate measurement noise, where in this case $\sigma = 0.20$.

The FOPDT model matrix is developed by least squares regression on a series of small open loop steps in the start up operating regime, the results of which are given in Table 5. As can be seen, the transfer functions display widely varying characteristics. Another important feature is the first element of the RGA, $\lambda_{11}$, is equal to 12.2, indicating a strongly interactive process.

The first demonstration with the rigorous distillation simulation is shown in Fig. 8. The first two set point changes in $x_D$, at samples 10 and 100, show good control of both $x_D$ and $x_B$ as the model is an accurate representation of the process. A disturbance to the feed rate is then injected into the system at sample 150 and continues for 50 samples.

As shown in Fig. 9, the disturbance is a ramp with a component of oscillation, the end result of which is a 20% change in the feed rate. The highly interactive and nonlinear nature of the process can be seen in the irregularity of the response of both $x_D$ and $x_B$ to the step disturbance, as seen in Fig. 8. This figure shows the fixed model case, where no adaptation is made at any point in the trajectory. As can be seen, not only does the disturbance cause a large deviation in both variables from set point, but changes the character of the process. This is reflected in the ensuing set point change, at sample 300, which gives a highly underdamped response in both variables.

Figure 10 shows how an ANN adaptation strategy performs for the same trajectory, where the only adaptation is made on the disturbance. The two
input patterns resulting from the disturbance both signify a shift in input steady state and thus the potential exists to change all four process model gains. The first error pattern is classified as a Class 1 pattern in Table 1 and therefore the two corresponding model gains, $K_p\text{11}$ and $K_p\text{12}$, are increased by a factor of 2.0 as given by equation (18). The $e_1$ pattern is classified as a Class 2 pattern. However, due to the irregularity of the pattern, the winning error network exemplar pattern has a matching score of 0.566, which is lower than the required vigilance factor of 0.75. Thus, although the pattern is classified as aggressive, the Decision Maker decides that the pattern is too irregular for reliable classification and thus no adaptation to the corresponding model parameters is made. Figure 11 shows the poor pattern match in this case, where again the solid line is the actual error pattern and the dotted line is the winning exemplar pattern from the error network. This demonstrates another strength of the ANN strategy in that the Decision Maker will make no adaptation unless it is confident that the ANN classification is accurate and reliable.

Although not pictured here, as expected it is difficult to fit a model to such dynamics and thus the model fitting routine is of little use. Even trying to fit just the model gains gives a very inaccurate model and performance similar to that shown in Fig. 7. If the ANN adaptation strategy was not present, any adaptive mechanism would have to wait until the next set-point change in the system before making any model adaptations.

The next demonstration with the distillation column shows how the adaptive strategy handles an oscillatory disturbance to the feed rate, which could be caused for example by a poorly tuned controller upstream of the column. Figure 12 shows the results of this demonstration for both $x_D$ and $x_B$. Again, the first two set point changes in $x_D$ demonstrate good control for both controlled variables. The oscillatory disturbance in the feed rate begins at sample 150 and continues for 100 samples. In this demonstration, the input network recognizes the dominant oscillatory character of the input patterns and thus decides that the patterns do not show a shift in operating regime. As a result, the Decision Maker bypasses the error network as it determines that model adaptation is not necessary. The response of both controlled variables following the next two set point changes in $x_D$, at samples 300 and 400, confirm this decision.

It has been discussed in the literature that a problem with some present pattern recognition controllers is a difficulty in reliable classification of oscillatory disturbances (Cao and McAvoi, 1990). The pattern recognition algorithm notices the high overshoot and damping and decreases the controller gain $K_c$ when no decrease is necessary, resulting in a poorly tuned controller.

To demonstrate this problem, Fig. 13 shows what would happen if the input network were bypassed and model adaptation was made solely based on the error patterns for the same disturbance trajectory. The oscillatory characteristics of error patterns for both the $e_1$ and $e_2$ patterns cause them to be classified as Class 1 patterns, and since the input network classifications are ignored, the result is an increase in all four model gains. The result is a degradation of performance as compared to Fig. 12. This demonstrates the advantage of a simultaneous analysis of both the input and error patterns in making reliable model adaptations.

![Fig. 11. Normalized error pattern for $x_D$ in response to disturbance in feed rate in Fig. 10.](image1)

![Fig. 12. Neural network based adaptive control of: (a) $x_D$, and (b) $x_B$ in distillation column simulation in response to oscillatory disturbance.](image2)
CONCLUSIONS

This work demonstrates the applicability of pattern recognition techniques to model adaptation in model based $2 \times 2$ controllers. The highly interactive nature of $2 \times 2$ systems makes qualitative pattern analysis much more difficult than is the case for SISO systems. As a result, the focus is on qualitative pattern analysis and simple heuristics for reliable model adaptation. As disturbance induced dynamics can mask the true process character, making model regression on input/output data difficult, disturbance rejection is the ideal situation for the use of pattern based techniques. The approach presented here can be seen as a complimentary technique to other modeling approaches that are more effective in modeling dynamics resulting from set-point changes. Also, by focusing on disturbance classification, the algorithm has more opportunities for model adaptation due to the greater frequency of disturbances in a typical chemical system.

The proposed approach to model adaptation in $2 \times 2$ systems is one which combines reasonably simple mathematical tools with intuitive reasoning that reflects the thought process of the plant engineer or process operator. The use of vector quantizing neural networks adds a simple to understand yet very reliable method of pattern classification. By focusing on general trends in the process data and multiplicative model parameter adaptation, the strategy is applicable to a wide variety of processes. It is shown to be effective on processes with strong interaction among the variables and is general enough to be applicable to a varying disturbance dynamics.

Acknowledgement—Acknowledgement is gratefully made to the National Science Foundation through Grant CTS-9008596, to Connecticut Innovations Inc. through an Elias Howe grant, and to the University of Connecticut Precision Manufacturing Center.

NOMENCLATURE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>DMC model parameter matrix</td>
</tr>
<tr>
<td>$C$</td>
<td>DMC control horizon</td>
</tr>
<tr>
<td>$c_i(t)$</td>
<td>Controller error for controlled output $i$ at sample $t$</td>
</tr>
<tr>
<td>$\mathcal{F}$</td>
<td>Filtering operator used in ART2-A network</td>
</tr>
<tr>
<td>$\mathcal{H}$</td>
<td>Hessian matrix</td>
</tr>
<tr>
<td>$G_i$</td>
<td>Model transfer function matrix</td>
</tr>
<tr>
<td>$\mathbf{f}$</td>
<td>Filtered and normalized network input pattern vector</td>
</tr>
<tr>
<td>$\mathbf{F}$</td>
<td>Network input pattern vector</td>
</tr>
<tr>
<td>$K_{ij}$</td>
<td>Steady state model gain for output/input pair $ij$</td>
</tr>
<tr>
<td>$M$</td>
<td>Dimension of network input pattern vector</td>
</tr>
<tr>
<td>$P$</td>
<td>DMC prediction horizon</td>
</tr>
<tr>
<td>$s_k$</td>
<td>Batch regression search vector at iteration $k$</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>Sample time</td>
</tr>
<tr>
<td>$t_M$</td>
<td>Time expressed as integer number of samples</td>
</tr>
<tr>
<td>$t_{md}$</td>
<td>Model dead time for output/input pair $ij$</td>
</tr>
<tr>
<td>$T_j^n$</td>
<td>Network exemplar tro network input pattern matching score for node $j$</td>
</tr>
<tr>
<td>$Q_i$</td>
<td>DMC weighting matrix for controlled outputs</td>
</tr>
<tr>
<td>$Q$</td>
<td>DMC weighting matrix for manipulated inputs</td>
</tr>
<tr>
<td>$\Delta u_i$</td>
<td>Vector of manipulated input moves</td>
</tr>
<tr>
<td>$u_i(t)$</td>
<td>Manipulated input $i$ at sample $t$</td>
</tr>
<tr>
<td>$x_b$</td>
<td>Bottoms composition</td>
</tr>
<tr>
<td>$x_d$</td>
<td>Distillate composition</td>
</tr>
<tr>
<td>$x_i^k$</td>
<td>Batch regression parameter vector at iteration $k$</td>
</tr>
<tr>
<td>$y_i(t)$</td>
<td>Controlled output variable $i$ at sample $t$</td>
</tr>
<tr>
<td>$y_j(t)$</td>
<td>Model estimate of true process output $i$ at sample $t$</td>
</tr>
<tr>
<td>$y_{ij}(t)$</td>
<td>Set point of controlled output $i$ at sample $t$</td>
</tr>
<tr>
<td>$x_j^n$</td>
<td>Network exemplar pattern at node $j$</td>
</tr>
</tbody>
</table>

Greek symbols:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Network initialization constant</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Network learning parameter</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>DMC tuning parameter</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Standard deviation of measurement noise</td>
</tr>
<tr>
<td>$\delta_v$</td>
<td>Model gain adjustment for model $ij$</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Normalization operator used in ART2-A network</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>Network filter threshold value</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Network vigilance parameter</td>
</tr>
<tr>
<td>$\tau_m$</td>
<td>Model time constant for output/input pair $ij$</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>Adjusted network input pattern for training of committed nodes</td>
</tr>
</tbody>
</table>

REFERENCES


