Monitoring Model Predictive Control Systems Using Pattern Classification and Neural Networks

Fred Loquasto III
loquasto@engineering.ucsb.edu

Dale E. Seborg*
seborg@engineering.ucsb.edu

Department of Chemical Engineering

University of California, Santa Barbara, CA 93106

May 28, 2003

ABSTRACT

A novel, pattern classification approach is proposed for monitoring the performance of Model Predictive Control (MPC) systems. Current MPC operation is compared to a simulated database of closed-loop MPC system behavior, containing various combinations of disturbances and plant changes. Neural network based pattern classifiers are used to classify the MPC performance as normal or abnormal, and to determine whether an unusual disturbance or significant plant change has occurred. If a plant change is detected, other classifiers are used to diagnose the specific sub-model(s) that are no longer accurate. The proposed methodology is successfully demonstrated in a detailed case study for the Wood-Berry distillation column model.

KEYWORDS

Controller monitoring and diagnosis; Model Predictive Control; Neural networks; Pattern classification

*Corresponding author. Tel.: (805) 893-3352 Fax: (805) 893-4731

1 Introduction

Model predictive control systems are widely used in the petrochemical and associated industries for two main reasons: (1) the model-based controller can perform well for multiple input-multiple output (MIMO) processes with complex dynamics, and (2) the controller accounts for inequality constraints on both input and output variables.[1,2] The ability of MPC to handle time-varying constraints allows processes to be operated at the current economic optimum conditions.[1,3] Furthermore, MPC can be used for relatively large control problems; it is not uncommon to have the number of outputs, inputs, and measured disturbances each be ten or more. There have been over 5500 industrial applications of MPC, mainly in refineries and petrochemical plants, according to a recent survey.[1]

Excluding major computer failures, there are three main reasons why MPC performance can degrade:

a. Unusually large or severe disturbances

b. A significant change in the static or dynamic plant characteristics

c. An equipment or sensor problem.

Plant characteristics can vary due to different feedstocks or feed flow rates, changes in operating conditions or product grades, process variations such as heat exchanger fouling and catalyst deactivation, weather conditions, etc. If such a change occurs, the process model used in the control calculations may no longer describe the plant behavior accurately, which can lead to poor MPC performance. Similarly, severe disturbances can cause MPC performance to degrade. The MPC control calculations are typically implemented as setpoints for lower-level control loops. Thus, if a lower-level control loop changes due to controller re-tuning, for example, the process model may no longer be accurate and MPC performance could degrade.

Considering the widespread application of MPC systems, it is somewhat surprising that the task of monitoring their performance remains a largely unsolved problem. In this paper, the term *MPC monitoring* means to perform an assessment of the current MPC system performance to determine whether or not it is operating normally, and if not, to determine if there are unusual disturbances present and/or the process behavior is no longer accurately described by the controller model. The goal of this research is to develop a systematic and general MPC monitoring strategy that can cope with practical realities such as inequality constraints. Consequently, the objective is to develop a monitoring strategy that can provide answers to the following questions:

1. Is the MPC performance normal or abnormal?

2. If it is abnormal, is the source of the problem one or more of the faults cited in items a - c?

Figure 1 illustrates these monitoring concepts in a simple tree that emphasizes the general approach proposed in this paper — detecting abnormal MPC system behavior and diagnosing its general cause. Although "Sensor Fault" and "Disturbance" are shown as distinct diagnoses, sensor fault detection is beyond the scope of this paper, but has been considered elsewhere.[4,5] Thus, the approach proposed in this paper is to group sensor faults and unusual disturbances together. Because MPC controllers are inherently nonlinear due to the inequality constraints, and usually of high dimension, a data-driven, pattern classification approach is proposed. The current MPC system performance is classified by comparison with a simulated database that contains a wide variety of closed-loop conditions. Feedforward, multi-layer perceptron (MLP) neural networks (NNs) are used to perform the classifications. In principle, the proposed monitoring strategy is applicable to both linear and nonlinear MPC systems. However, the scope of this paper is restricted to linear MPC.

This paper is organized as follows. In Section 2 an overview of MPC is presented, followed by a review of current multivariate controller monitoring and assessment methodologies in Section 3. Background material on neural networks and the proposed monitoring methodology is presented in Section 4. This section also describes the generation of the simulated database, the specific neural network configurations, and performance measures used for pattern classification. The proposed MPC monitoring strategy is evaluated in Section 5 for a simulated case study, the Wood-Berry distillation column model. Finally, the summary and conclusions are presented in Section 6.

2 Model Predictive Control

Although each proprietary, commercial MPC system has its own method for calculating control moves and handling constraints, a generic dynamic control move calculation is based on a constrained optimization. Future predictions of the process output, $\widehat{y}$, from a linear dynamic model are used to determine the optimal control move sequence, $\Delta u^*(k)$. The linear model can be formulated in many ways; commercial algorithms often use step-response or ARX models, and state-space formulations are now becoming available.[1,2] To perform the on-line control calculation, the cost function J in Eq. (1) is minimized by calculating the next

M optimal control moves, where M is called the *control horizon*, over a *prediction horizon* from P_0 to P. That is, the MPC problem is solved for the optimal control move sequence, $\Delta u^*(k) = \arg\min_{\Delta u} J$, by minimizing

$$J = \sum_{i=P_0}^{P} \|\widehat{y}(k+i|k) - r(k+i|k)\|^2_{\mathbf{Q}(i)} + \sum_{i=0}^{M-1} \|\Delta u(k+i|k)\|^2_{\mathbf{R}(i)} \tag{1}$$

subject to inequality constraints:

$$u^- \leq u(k+j) \leq u^+, \quad j = 0, 1, \ldots, M-1 \tag{2}$$

$$\Delta u^- \leq \Delta u(k+j) \leq \Delta u^+, \quad j = 0, 1, \ldots, M-1 \tag{3}$$

$$y^- \leq \widehat{y}(k+j) \leq y^+, \quad j = P_0, P_0+1, \ldots, P \tag{4}$$

where the process output is $y \in \mathbb{R}^l$, the manipulated input is $u \in \mathbb{R}^m$, and Δu represents the next M control moves, starting at current time k. The output prediction, $\widehat{y}(k+i|k)$, is made at time k for i steps ahead. The setpoint is $r \in \mathbb{R}^l$. $\mathbf{Q}(i)$ and $\mathbf{R}(i)$ are weighting matrices for the predicted control error and input moves. The diagonal elements of $\mathbf{R}(i)$ are sometimes referred to as *move suppression factors*. The low and high limits for x are represented by x^- and x^+. Also, $\|x\|^2_{\mathbf{S}} \triangleq x^T \mathbf{S} x$. The control move calculation in Eqs. 1–4 can be formulated as a quadratic programming (QP) problem.[2,1]

The optimization is performed at each time step, but only the *first* input move in the optimal sequence, $\Delta u^*(k)$, is implemented. The entire optimization is then repeated at the next time step, resulting in a *receding horizon* approach.[2] If the optimization problem is unconstrained, an analytical solution can be obtained in which the control moves are based on error feedback with a constant controller gain matrix.[2]

In many commercial MPC systems, feedback is introduced by shifting the output predictions by a bias equal to the prediction error at the previous time step. This approach is equivalent to assuming a step disturbance at the output that is constant throughout the prediction horizon. This technique, referred to as the *DMC disturbance model*,[2,6] can give sluggish disturbance rejection in certain situations.[7]

The actual control move calculation in commercial MPC systems is typically a proprietary algorithm that is not an actual QP technique. However, the QP formulation is popular in the MPC research literature and is representative of the problems that commercial algorithms solve. In addition, most commercial MPC systems include a steady-state optimization that resides in an intermediary position between the plant-wide

optimization and the control move calculation of Eqs. 1–4. This steady-state optimizer is often a linear program (LP) that uses the steady-state gains from the process model, and the constraints, to find feasible "targets" (setpoints) for y, u, or both.[2,1] The analysis of this target calculation layer for monitoring purposes is beyond the scope of this paper, but is discussed by Kozub.[8]

3 Previous work

This section presents a brief overview of the literature related to MPC monitoring. General controller performance assessment techniques are summarized, and methods specifically developed for MPC monitoring and assessment are reviewed.

3.1 SISO controller performance assessment techniques

Monitoring and assessing control system performance has been widely considered for linear, single-input single-output (SISO) systems. The most common techniques are the stochastic *minimum variance* methods[9,10] in which the current control performance is compared to a type of ideal performance, minimum variance control. These SISO techniques are relatively easy to use because they require only normal, closed-loop operating data and knowledge of the process time delay. The method has also been extended to feedforward systems and alternative benchmarks have been proposed.[8−15]

However, a SISO analysis can provide misleading results for MIMO processes due to process interactions. For example, Huang and Shah[16] consider an industrial paper machine header box example in which the SISO minimum variance metric indicates good performance, but when MIMO minimum variance metrics are evaluated (see Section 3.2), the performance is actually poor. Furthermore, it has been argued that the minimum variance benchmark may not even be desirable.[10]

3.2 MIMO controller performance assessment techniques

Monitoring and assessing performance of a MIMO system is much more involved than for SISO systems. Minimum-variance performance assessment techniques have been extended to MIMO systems,[16,17] but the time-delay structure, *i.e.* the *interactor matrix*, must be known or estimated.[16,18] This requirement alone makes MIMO assessment more difficult than for the SISO case. In essence, MIMO minimum variance assessment techniques compare the current performance to the best possible performance that would be

achieved if a linear, MIMO feedback controller specifically designed for minimum variance was used. Thus, it does not compare performance relative to the best possible multi-loop/decentralized control system performance.[19] Extensions and alternatives to the MIMO MV assessment methodology have been proposed, including methods that simplify the determination of the interactor matrix,[18] and techniques for using system identification to evaluate controller performance.[9]

3.3 Monitoring techniques for MPC

All of the MIMO control monitoring techniques mentioned in the previous section have a major shortcoming for MPC applications — they are not directly applicable to constrained control systems. Furthermore, inequality constraints that vary with time are a distinguishing feature of practical MPC applications.[1] It is important to note that most assessment techniques specifically designed to account for these constraints require a process model.[9]

Next, we consider monitoring strategies that have been proposed specifically for MPC. Tyler and Morari[20] present a technique using qualitative propositional logic with quantitative fault models to diagnose faults or indicate model error in an MPC system. Kesevan and Lee[21] present a model-based methodology involving statistical tests for detecting and diagnosing faults in MPC systems. Wan and Huang[22] propose a measure of the robust performance variation in a closed-loop system caused by model-plant mismatch (MPM). The analysis is based on the maximum singular value of the generalized closed-loop error transfer function. A data-driven approach for updating the controller model to reduce plant-model mismatch is presented by Samyudia *et al.*[23] An expert-system approach is described by Leung and Romagnoli[24] that indicates the presence of the following operating conditions: invalid model, presence of a severe oscillation, rejection of deterministic disturbances, or stochastic control operation (which is considered to be normal). Zhang and Henson[25] present a method for assessing constrained MPC performance. They develop a stochastic measure I_{MPC} that compares expected MPC performance to actual performance by performing a constrained MPC simulation in parallel with the actual MPC operation. Kozub[8] states that for monitoring a constrained MPC system consisting of both a steady-state optimizer and a dynamic controller, the LP layer is crucial to the analysis.

Some authors choose to apply the unconstrained minimum variance control benchmark to MPC systems, arguing that although it is not designed for constrained systems, it nevertheless remains a tool that can be

used to assess MPC system performance. For example, Kadali et al.,[26] Vishnubhotla et al.,[27] and Miller and Huang[28] apply minimum variance assessment to MPC systems. Haarsma and Nikolaou[29] apply MIMO MV assessment to an MPC-controlled, snack food process. They present an approach for the case of constrained inputs that re-filters the interactor matrix to obtain one for the reduced dimension system of non-constrained inputs. Hugo[30] argues that control engineers are mainly interested in how well each individual output is held to its setpoint/LP target, so that SISO MV techniques are appropriate for multivariate systems, although the results are biased on account of the process interactions.

Huang and coworkers discuss model validation as one of several tools that can be combined to perform MPC assessment.[31,32,26] They present a general methodology for model validation in MPC systems based on a two-model divergence algorithm that can detect process changes regardless of disturbance changes.[33] This technique requires a significant amount of input excitation, but the authors propose that the required excitation signal can be injected into the process automatically by the optimizer. Huang[34] also presents model-validation techniques for SISO and MIMO systems, based on the *local approach* discussed in Basseville and Nikiforov.[35]

Patwardhan and Shah[36] review fundamental limitations on controller performance due to hard constraints, using an Internal Model Control (IMC) framework. They also present results on how MPC performance is affected by model uncertainty and process nonlinearity. A linear quadratic regulator (LQG) benchmark for MPC performance has been proposed by Shah and coworkers.[37,36,16] They argue that the LQG benchmark is appropriate for comparison with MPC because a LQG controller can be designed to weight both output error and input moves in analogy with MPC. They have also proposed two benchmarks based on the objective function in the MPC optimization, a design case benchmark, and a historical benchmark.[37,36,38] These historical and design benchmarks are applied in a novel framework by Schäfer and Çinar[39] to perform MPC system fault detection and diagnosis.

Ko and Edgar[40] suggest comparing the output variances achieved by unconstrained and constrained minimum variance controllers (in simulation) to the actual output variance of the MPC system. The unconstrained minimum output variance is calculated by simulating a finite-time, moving horizon minimum variance controller,[40] or by other MIMO MV methods.[41,16] The numerical value of the constrained minimum output variance is found by simulating a constrained, optimization-based controller designed for minimum output variance, using optimal disturbance prediction and a disturbance model identified from the process

data.

3.4 Commercial products

MPC system suppliers offer a few MPC assessment tools. *AspenWatch* is a performance monitoring tool for *DMCplus* that offers "statistical performance measures" relative to an "optimum standard", a "design standard", and "previous operating experience".[42] Honeywell's *Performance Monitor* for its *RMPCT* model predictive control systems calculates and summarizes several simple statistics of controller operation.[43]

4 Proposed methodology

The proposed methodology for monitoring MPC systems is based on pattern classification using neural networks. It consists of the following steps:

1. Select an appropriate set of current operating data to be analyzed (the test data).
2. Create a database consisting of windows of simulated, closed-loop system responses.
3. Train the NNs for pattern classification.
4. Classify the test data.

The individual steps will now be discussed in greater detail. In Step 1 a set of test data is selected by the user for MPC performance evaluation. In Step 2 the simulated database is created after the test data have been selected. Thus, the proposed methodology is implemented as an "on-demand" approach. A database of simulated, closed-loop, MPC responses is created for a wide variety of disturbances and simulated process changes (to introduce *model-plant mismatch*, or *MPM*) but the *same* setpoint trajectories, measured disturbances, and constraint limits that are present in the test data are also used in the simulations. The generation of the simulated database will be discussed in more detail in Section 4.1.

In order to develop effective pattern classifiers, two independent databases are created: one for training and one for validation. The training database is used to train several pattern classification NNs to distinguish between normal operation, unusual disturbances, and significant plant changes. The inputs to the NNs are scalar *features* extracted from the data. These features will be discussed in more detail in Section 4.2.

Once trained, all networks are evaluated on the validation database, and the network with the best validation performance is used for subsequent evaluations of the MPC performance.

The motivation for using the actual setpoint, measured disturbance, and constraint information in both the training and validation databases is as follows. In our experience, the pattern classification approach has a much higher degree of accuracy if the closed-loop database is simulated for conditions that are as close as possible to the conditions present in the test data.

In designing the simulated database and performing the subsequent classifications of MPC performance, four classes of operation are proposed:

1. Normal. No abnormal disturbance or abnormal model-plant mismatch is present (labeled N).

2. Disturbance. An abnormal (sustained or large magnitude) disturbance occurs (labeled D).

3. Plant Change. The plant behavior differs significantly from that described by the process model (labeled P).

4. Disturbance and Plant Change. Both an abnormal disturbance and a significant model-plant mismatch are present (labeled $P + D$).

By definition, these four classes are mutually exclusive. Thus, for any set of test data, the process operation can only belong to one of these four classes. We have evaluated several alternative classification strategies:

Method 1. Three individual classifiers. In this method, three independent classifiers are used to answer the following yes/no questions:

$i.$ Is the operation normal or abnormal? (the *AB* classifier)

$ii.$ Is an abnormal disturbance present? (the *DIST* classifier)

$iii.$ Is there significant model-plant mismatch? (the *MPM* classifier)

Method 2. A single four-class classifier. A single NN classifier indicates membership in one of the four operating classes discussed above.

Method 3. An exclusion strategy. The exclusion strategy is based on the results of two NNs, the *AB* and *DIST* classifiers. It provides a novel detection scheme for plant changes.

Method 4. A combination approach. In order to provide an alternative technique for detecting plant changes, the *AB* classifier can be used in combination with the *MPM* classifier. A "Plant change"/"No plant change" classification is made only if the two classifiers agree.

When abnormal MPC system performance has been detected, it is desirable to determine whether the root cause is model-plant mismatch. In particular, we would like to be able to detect plant changes while having a low false alarm rate. It is very important to minimize the false alarm rate for the following reason: if the monitoring strategy incorrectly concludes that the plant model is inaccurate, expensive and time-consuming plant tests to re-identify could be performed unnecessarily.

For the simulation results of Section 5.2, the *AB* and *DIST* networks are very accurate, while the *MPM* network is less accurate. These considerations motivated the development of Method 3, the *exclusion strategy*, which is illustrated in Figure 2. In this novel approach, only the *AB* and *DIST* networks are used to detect plant changes; the *MPM* network is not used. The current operating period is classified as a plant change if the *AB* network indicates "Abnormal" and the *DIST* network indicates "No Disturbance". The advantage of this exclusion strategy is more accurate classification of plant change conditions, *i.e.* a lower false alarm rate. A minor disadvantage is that operating periods that exhibit a simultaneous disturbance and plant change may be classified as either "Disturbance" or "Plant Change", rather than "Plant Change + Disturbance". Furthermore, if the plant change persists, and the abnormal disturbance eventually dies out, a "plant change only" situation would then exist that can be readily detected using the exclusion strategy. Thus, the ability of Method 3 to detect a plant change during an operating period with a simultaneous disturbance is considered to be less important than minimizing the false alarm rate for MPM detection.

The *combination approach* of Method 4 is an alternative classification method for detecting plant changes. As shown in Figure 3, Method 4 classifies the current test data as "plant change" (P) *only* if the *AB* and *MPM* networks agree. Thus, the only allowable classifications are "Normal" and "Plant Change". The advantage of this approach is that it will detect plant changes even when a significant disturbance is present, but it tends to have a higher false alarm rate because of the lower performance of the *MPM* network.

Performance metrics. In order to evaluate the performance of the proposed MPC monitoring techniques, it is informative to consider a test database with known characteristics. The following *efficiency* and error measures are proposed:

$\eta_i \triangleq$ *efficiency* $= \frac{N_{1,i}}{N_{T,i}} \times 100\%$, the percentage of datasets that are correctly classified (5)

where $i = \{$Overall, $N, D, P, P+D\}$. $N_{1,i}$ is the number of type i correctly classified of the total number of type i in the database, $N_{T,i}$. 'Overall' refers to the performance for all four classes of operation.

$E_1 \triangleq$ the percentage of 'Type I' errors (the null hypothesis H_0 is rejected when actually true).

$E_2 \triangleq$ the percentage of 'Type II' errors (the null hypothesis H_0 is not rejected when it is false).

For the individual classification networks, the null hypothesis is specified to be the "normal" condition. For example, the null hypothesis for the *MPM* classifier is:

H_0: No significant model-plant mismatch exists.

For the four-output network, each output has its own null hypothesis. Thus, for the ith output, $i = \{N, D, P, P+D\}$, the null hypothesis is:

H_{0i}: The current dataset belongs to class i.

To illustrate the Type I and Type II error calculations for the individual network classifiers, define $e = t - o$, where t represents the target and o represents the network output. Both t and o are binary variables where 0 denotes the "normal" condition, and 1 represents the "abnormal" condition. Thus, a Type I error occurs when $e = -1$, and a Type II error when $e = +1$. The E_1 and E_2 performance measures are based on all of the datasets.

The Type I and Type II errors for the four-output network are formulated differently. Define e, t, and o as four-element vectors, where each element is a binary variable for which 0 denotes the normal condition. Suppose that a test dataset is from class i. A Type I error for output i occurs when either $e(i) = +1$, or when $e(i) = 0$ and $o(j) = 1$ for any $j \neq i$. That is, a Type I error occurs when output i is incorrect, or when another output ($j \neq i$) incorrectly classifies the test data from class i as being from class j. Thus, the E_1 measure does not double count errors with respect to the different classes. A Type II error for output j occurs when $e(j) = -1$, when output j mislabels test data that belongs to class i as class j. Thus, the E_2 measure for each output is not affected by the predicted classification of the other outputs. Double counting can occur.

Additional performance metrics are defined for the exclusion strategy. Two *accuracy* measures are:

$p(P) \triangleq$ the percentage of datasets classified as P that actually belong to class P.

$p(P, P+D) \triangleq$ the percentage of datasets classified as P that actually belong to either P or $P+D$.

Two false alarm rates are defined that are appropriate for the exclusion strategy:

$$FAR(P) \triangleq 100\% - p(P) \tag{6}$$

$$FAR(P, P+D) \triangleq 100\% - p(P, P+D) \tag{7}$$

False alarm rate $FAR(P)$ considers a 'P' classification correct only if the test data actually belong to class P. For the second measure, $FAR(P, P+D)$, if the exclusion strategy labels a $P+D$ condition as P, this classification is considered correct and thus does not contribute to the false alarm rate.

4.1 Design of the simulated database

The process model used in the MPC calculations obviously plays a key role in the proposed MPC monitoring methodology. This model is used to simulate the closed-loop behavior of the MPC system for a wide variety of disturbances and process changes. The details of the database generation will now be discussed.

The MPC monitoring strategy is based on classifying a window of closed-loop data from an operating period of interest, the *test data*. The window length is selected to be the longest settling time for the process outputs in order to allow all transients to settle out for reasonable controller tuning. The simulated data windows used for training the classifiers should have the same setpoint trajectories, measured disturbances, and constraint limits as the selected test data.

In general, the disturbance models for unmeasured disturbances are not known. Thus for MPC monitoring, it is important to simulate a wide variety of disturbances with random magnitudes and random start times. Two general types of unknown disturbances are introduced into the simulated data, *output disturbances* and *input disturbances*. For the case study of Section 5, the output disturbances include steps, ramps, sinusoids, and stochastic sequences (white noise filtered by various transfer functions). They were generated independently for each output, with the type and magnitude of the disturbance chosen randomly. It is advantageous to note that step and ramp output disturbances are equivalent to sensor biases and sensor drifts. Thus, the proposed methodology can be extended to include detection and diagnosis of faulty sensors. Input disturbances are introduced as proxies for unknown disturbances that tend to be filtered by the process

itself. Steps, ramps, sinusoids, and stochastic sequences were introduced independently at each input as the basis for input disturbances.

In this paper, the choice of disturbance type (input or output disturbance for each variable), its magnitude, the time of occurrence in the data window, and other parameters (*e.g.* ramp rate, oscillation frequency) are chosen randomly from uniform distributions. This approach helps ensure that a very wide range of possible disturbance effects are included in the simulated database. The basic idea is to choose generic disturbance types that have at least some of the characteristics of the actual disturbances.

For MPC monitoring, a critical issue is the ability to detect when the model-plant mismatch has become unusually large. In the proposed methodology, it is assumed that a physical model of the process is not available and the monitoring is based on the empirical, linear model used in the MPC calculations. Plant changes are included in the simulated database by perturbing the nominal model and using it as the simulated plant. The MPC calculations still use the nominal model. If the degree of model inaccuracy can be characterized, training data with this amount of model variability could be considered as "normal" data, for purposes of training the NN classifiers.

The way in which the model is perturbed depends on the structure of the model. For transfer function models, the gains, time delays, and time constants are varied. For large MPC problems, even if the plant model consists of first-order plus time delay (FOPTD) transfer functions, it is not feasible to evaluate all possible combinations of model parameter changes. For example, consider a 2×2 process model that consists of FOPTD sub-models. If only three values of each model parameter are considered, the total number of combinations for the 12 parameters is $\sum_{r=1}^{3\cdot 12} \binom{3\cdot 12}{r} \sim O(10^{10})$. Thus, a more tractable method than simple enumeration is necessary.

A proposed selection procedure using random parameter selection and random magnitude perturbations for the case study in Section 5 is summarized in the Appendix. By requiring the MPM detection and diagnosis NNs to be able to detect random, arbitrary model parameter changes, the simulated problem may actually be more difficult than an actual application. For example, time delays and time constants often change with flow rates, so that these parameter changes are correlated rather than independent. It is reasonable to expect correlated parameter changes in actual applications, which is encouraging for a monitoring strategy that performs well in detecting arbitrary and random plant changes.

To complete the database design, the total amount of training data, and the relative amounts of N, D, P,

and $P + D$ datasets must be specified. The total number of datasets in the training database can be selected using heuristic arguments from the neural network literature. For example, a general heuristic states that the total number of training sets, N_t, should be related to the number of free parameters (weights) of the neural network, w_{tot}, and the desired misclassification rate, ϵ, by the expression:[44]

$$N_t \sim O\left(\frac{w_{tot}}{\epsilon}\right) \tag{8}$$

This expression agrees with intuition for curve fitting, which suggests that $N_t > 2w_{tot}$ to prevent overfitting.

After choosing N_t, the allocation of the training data for the four independent operating classes must be specified. The authors have evaluated several alternative allocations.[45] For simplicity, each of the four independent operating classes was allotted $N_t/4$ datasets in the case study of Section 5.

4.2 Design and training of NN classifiers

Multi-layer perceptron. A multi-layer perceptron (MLP) is a type of feedforward artificial neural network that consists of an input and output layer, with one or more "hidden" layers between the input and output layers. The hidden and output layers contain neurons that perform a mathematical operation on a vector input to generate a scalar output.[44,46] A generic MLP NN is illustrated in Figure 4 where there are P inputs, H hidden neurons, and R outputs. The features (inputs) are indicated by 'f';'g' represents the hidden layer neuron output values, 'o' represents the network outputs, and 'w' and 'v' represent network weights. The bias inputs have been omitted in Figure 4. For the proposed pattern classification approach, MLP NNs are used with a threshold function (*i.e.* "rounding") that is applied to the outputs to give a binary (0 or 1) value to indicate a "No" or "Yes" decision. It is possible to use the actual output value (without rounding) as a measure of confidence of the classification, but this paper does not consider this option.

Pattern classification. As discussed in Section 4, several classification methods are proposed. Either a single four-class classifier, or a set of three individual classifiers can be used to classify MPC system performance. The four-class classifier consists of a single neural network with four outputs, one output for each of the independent operating classes described in Section 4. That is, the four outputs correspond to (N, D, P, $P + D$). For example, if the second output has a value of 0.8, it is rounded to 1 and the test data being analyzed are classified as belonging to the "Disturbance" class. Because the four operating classes are mutually exclusive, a window of test data can belong to *only* one class. Thus, if the four-output network has

more than one output with a value of 1, or if all four outputs are 0's, this classification is inconsistent and should be discarded. Additionally, the relative output values (not the rounded values) provide an estimate of the degree of confidence we have in the most likely classification, with respect to the other potential classifications. This approach using independent outputs is often formalized in the neural network-based fault detection literature, for example, with Bayesian classifiers, because it can give *a posteriori* probability estimates of the occurrence of each independent class.[47,48]

For Method 1 three independent classifiers, *AB*, *DIST*, and *MPM*, are implemented as three, single-output networks. For each network, the output is rounded to provide a "Yes" (1) or a "No" (0) answer to one of the three classification questions presented in Section 4.

To diagnose the specific models that are inaccurate after a "plant change" situation has been detected, a similar approach can be used. For a system with l inputs and m outputs, the process model consists of $n = l \times m$ individual, SISO sub-models. For smaller MPC systems, n single-output networks can be used to diagnose which sub-models are no longer accurate. In this case, a '0' indicates that a model is accurate and a '1' indicates that it is inaccurate.

NN inputs. The inputs to the neural networks are features that have been extracted from the window of test data being considered. These features should be metrics that effectively characterize the closed-loop operation. The following variables are used as the basis for the features: the manipulated inputs and controlled outputs, the differenced inputs, the control errors (CE), and the one-step prediction errors (residuals),

$$\epsilon_i(k+1) = y_i(k+1) - \widehat{y}_i(k+1|k) \quad (i = 1, 2, \ldots, l) \tag{9}$$

where $\widehat{y}_i(k+1|k)$ is the one-step ahead prediction made at time k. The features are grouped into the categories shown in Table 1 and are discussed below. For the case study, only sub-sets of these features are selected to be NN inputs.

Basic statistics. This category of features consists of basic statistics such as sample means, standard deviations, and total variations[49] (TV), where $TV(x) \triangleq \sum_{i=1}^{\infty} |x_{i+1} - x_i|$.

Correlation coefficients. In addition to basic statistics, it is reasonable to include features that characterize the dynamic behavior of the process. For this reason, autocorrelation (ACF) and partial autocorrelation (PACF) coefficients[50] are included in the feature list. ACF and PACF coefficients also have a useful interpre-

tation. For an autoregressive (AR) process, the ACF decays exponentially (it may also oscillate depending on the AR parameters), while the PACF becomes zero for lags greater than or equal to the order of the AR process. For a moving average (MA) process, the ACF is identically zero for lags greater than or equal to the MA order, while the PACF decreases exponentially. For an ARMA process, mixed behavior occurs.[50] For most of the simulations in the case study, the coefficients quickly decayed to within the confidence bounds with increasing lag. Consequently, only coefficients for the first few lags are included. Kozub and Garcia[51] have also used residuals, differenced inputs, and autocorrelation measures for SISO controller performance assessment.

Another measure of dynamic behavior is the variogram, V_{ij}^z, where $V_{ij}^z \triangleq var(z_{t+i} - z_t)/var(z_{t+j} - z_t)$. It is the ratio of the variances of i-step differences in the data to j-step differences. For example, V_{i1}^z is equivalent to the autocorrelation. The variogram has the benefit that it can be used to analyze non-stationary processes.[52]

Classical performance metrics. Traditional metrics for setpoint changes, such as rise time, settling time, and overshoot can also be used as features. The rise time is defined as the time from the setpoint change until the output first reaches its setpoint; the settling time is the time required for the output to enter and remain within a $\pm 10\%$ band about the setpoint.

To summarize, the potential feature set includes the following items: sample means and standard deviations, and total variations; traditional performance measures such as rise time (t_r), settling time (t_{set}), integral of absolute error (IAE), and overshoot (OS); and statistics such as sample autocorrelation ($\hat{\rho}_k$) and partial autocorrelation ($\hat{\phi}_{kk}$) coefficients and variogram measures V_{ij}^z.

It is important to scale NN inputs to the same range so that relatively smaller inputs do not become overshadowed by larger inputs. In this paper, the features that are used as inputs are first 'standardized' to zero mean and unit variance. Then, the inputs are scaled between ε and $1 - \varepsilon$, where $\varepsilon = 0.15$. The scaling is based on Eq. 10, where $\tilde{x}_i$ is the scaled feature i, and $\hat{x}_i$ is the standardized feature, giving a minimum scaled feature value of 0.15 and a maximum of 0.85.

$$\tilde{x}_i = (1 - 2\varepsilon) \ln \left[\frac{\hat{x}_i - \hat{x}_{i,min}}{\hat{x}_{i,max} - \hat{x}_{i,min}} + 1 \right] / \ln[2] + \varepsilon \tag{10}$$

This specific form of log scaling helps spread out small values that are disproportionately close compared to large values.[46]

Selection of NN inputs. There are many ways that NN inputs can be chosen from features. A straightforward approach is to first collect features into reasonable groups that would effectively describe the behavior of a window of test data. Then, the classification ability of each input group would be evaluated based on its performance for the validation database. Consequently, many different groups of the features considered in the previous section were evaluated as NN input variables. In the case study (Section 5.2), many of the input groups resulted in similar performance. An alternative approach would be to develop an influence metric for the inputs, train either linear perceptrons or MLPs on all input features, then select the k most influential inputs.[46] However, the disadvantage for the MLP case would be the significant increase in training time due to many more inputs and hidden neurons (see below).

Neural network design. In this paper, the NNs employ tan-sigmoidal activation functions in the hidden layer and log-sigmoidal activation functions in the output layer. Log-sigmoidal functions could also be used in the hidden layer, but preliminary evaluations resulted in a slight increase in performance using the tan-sigmoidal functions. In order to specify the number of neurons in the hidden layer, H, three heuristic design methods were considered:[46]

1. $H = K$
2. $H = \sqrt{PR}$
3. $H = 1.7095 \log_2(2K)$

where P is the number of network inputs, R is the number of network outputs, and K is the number of distinct target patterns that the network must learn. One network was designed for each of the three values of H.

It is well known that neural networks trained by backpropagation can become trapped in local minima.[44] Thus, for every combination of input group and network design method, five individual classification networks were trained using the same training database. Each of these networks had randomized initial weights. The five resulting networks were compared using the training database, and the best NN out of the five was chosen for further evaluation, for each input group-network design combination. This training strategy helped reduce the likelihood of selecting a poorly performing network that was trapped in a local minimum. These "best networks", one for each input group-NN design combination, were then evaluated

on the validation database. The best performing NN on the validation database (*i.e.*, a specific input group and NN design) was then chosen to be the classification network for subsequent applications.

5 Case Study: Wood-Berry distillation model

The Wood-Berry model is a well-known 2×2 transfer function model of a pilot-plant distillation column for a methanol-water mixture.[53] The output variables are the distillate and bottoms compositions, x_D and x_B [$wt\ \%$ methanol]. They are controlled by manipulating the reflux and steam flow rates, R and S [lb/min]. The feed flow rate, F, is an unmeasured disturbance variable. The column model[53,6] is shown in Eq. 11.

$$\begin{bmatrix} x_D(s) \\ x_B(s) \end{bmatrix} = \begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \\ \dfrac{6.6e^{-7s}}{10.9+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} R(s) \\ S(s) \end{bmatrix} + \begin{bmatrix} \dfrac{3.8e^{-8s}}{14.9s+1} \\ \dfrac{4.9e^{-3s}}{13.2s+1} \end{bmatrix} F(s) \tag{11}$$

This model is a classical example that has been used in many previous papers concerned with process control, monitoring, and identification.

5.1 Generation of the simulated databases

In this section, a detailed description of the simulated case study is presented.

MPC tuning parameters. The MPC controller was tuned by trial-and-error to give a reasonably fast setpoint response. The prediction horizon in Eq. 1 was $P = 30$, with $P_0 = 0$, and the control horizon was $M = 5$. Weighting matrices $\mathbf{R}$ and $\mathbf{Q}$ were chosen to be identity matrices. The control execution period was $\Delta t = 1$ min. The standard "DMC" constant output disturbance approach was used to update the model predictions.

For the purpose of this case study, unconstrained MPC was employed in order to reduce the time required to create the simulated databases for numerous case studies (many more than the ones discussed here). However, it is important to note that the proposed methodology is also directly applicable to constrained MPC.

Training and validation databases. For the purpose of verifying the reproducibility of the results, the methodology was evaluated by repeating the monitoring procedure with five training databases and separate

validation and test databases. Each dataset in a database was based on the same setpoint change, $x_D^{sp} = -1$ at $t = 0$, which was chosen because x_D is the faster responding of the two outputs. Based on process settling times, all simulations used data windows that were 100 minutes long. The individual datasets could also include disturbances and/or plant changes, as described below. Plant changes were created based on the procedure presented in the Appendix, in which the choices of the model(s) and the associated model parameter(s) to be perturbed were made randomly. The perturbation magnitudes for the model parameters were chosen randomly from the range $\pm$[12.5 – 37.5] % of the nominal values.

The training and validation data were created using the output and input disturbance approach discussed in Section 4.1. The type of output and input disturbances were chosen randomly for each output from the following set of disturbance types: step, ramp, sinusoidal, and stochastic. (Two stochastic transfer functions were used for training, a third one for validation.) The step disturbances occurred at random starting times between $t = 25$ min and $t = 75$ min, while the ramps began between 25 and 50 min. The ramp duration was randomly specified to be between 25 and 50 min. The amplitudes of the output disturbances were chosen randomly in the range $\pm[0.25 - 3.25]$ wt%. The magnitudes of the input disturbances were adjusted so that the open-loop changes in the outputs were approximately the same as for the output disturbances.

For stochastic output and input disturbances, the magnitude of the white noise sequence used to generate the stochastic disturbance and the magnitude of the resulting stochastic disturbance (the output from the stochastic disturbance transfer function) were scaled to provide approximately the same effects on the outputs as the deterministic disturbances. Two discrete transfer functions were used to create stochastic disturbances in the training data,

$$G_{d1}(z) = \frac{1}{z - 0.95} \tag{12}$$

and

$$G_{d2}(z) = \frac{2z - 0.4}{z - 0.8} \tag{13}$$

A third transfer function was used for the validation data:

$$G_{d3}(z) = \frac{1}{(z - 0.9)(z - 0.5)} \tag{14}$$

For the sinusoidal disturbances, the frequency was a random number uniformly distributed in the interval $[0.03 - 1.03]$. Gaussian measurement noise with an approximate standard deviation of 0.05 wt% was added to each output, and Gaussian process noise was added to the two process inputs (R, S) with approximate

standard deviations of 0.006 and 0.003, respectively. These numerical values gave approximately the same standard deviation at the outputs in open-loop as the measurement noise did. These database design parameters and the number of datasets for each type of operating condition are summarized in Tables 2 and 3. The total amount of training data, $N_t = 2000$, was chosen to exceed the guideline in Eq. (8), with $\epsilon = 0.1$ and $w_{tot} = 131$ network weights and biases, giving $N_t = 1310$. The number of validation datasets, 1000, was chosen arbitrarily to be less than the number of training datasets.

Test database. For the simulated case study, the test database takes the place of actual process data. In order to make the evaluation of the test database more interesting, the unmeasured disturbances were chosen to be feed flow rate disturbances generated using the actual disturbance model in Eq. 11. This model had been ignored in creating the training and validation data. For the feed disturbances, steps, ramps, sinusoids, and stochastic sequences were used. The ranges of disturbance starting times were the same as for the training and validation data, as were the sinusoid frequencies. The disturbance magnitude was in the range $\pm[12.5 - 37.5]\%$ of the nominal feed flow rate (2.45 lb/min). These parameter values were again chosen randomly. For the stochastic sequences, a fourth disturbance transfer function,

$$G_{d4}(z) = \frac{0.9z - 0.00894}{z - 0.9802} \tag{15}$$

was used. Plant changes were introduced randomly in the same manner as for the training and validation databases. Tables 2 and 3 summarize this information, along with the relative amounts of each operating condition.

NN input groups. The NN inputs were selected from a total of 100 features that were based on the categories in Table 1. The inputs were scaled using the log-scaling technique discussed in Section 4.2. Table 4 describes the fourteen different groups of inputs that were used to train the neural networks. The input groups were chosen to investigate how different types of features perform relative to one another, for example, how static features, such as means and standard deviations, compare to dynamic features, such as correlation coefficients. The best performing input groups were selected based on performance for the validation database.

All simulations were performed in MATLAB using the MPC Toolbox.[6] In the case study, the MATLAB Signal Processing Toolbox[54] function, `xcorr`, was used to calculate the ACF coefficient estimates. The

coefficients from this function differ somewhat from the standard sample autocorrelation function[50] because mean values are not removed. Consequently, these estimates tend to decay more slowly, but at small lags, are similar to the standard estimates. The neural networks were created using the MATLAB Neural Network Toolbox.[55] The networks were trained for 100 epochs using batch training and the Levenberg-Marquardt algorithm.[55] The results presented are averages for five different networks trained using five independent training databases.

5.2 Results

Detailed results for the proposed MPC monitoring strategy are presented in this section.

Input variable group. The classification results (η values) for each NN input group and the test database are presented in Table 5. The performance of many of the input groups is about the same, suggesting that the choice of which features to use as inputs is not as critical a decision, as one might anticipate *a priori*. The *AB* and *DIST* classification networks perform very well. The *MPM* network performs reasonably well, considering that detecting plant changes is a very difficult problem. The small standard deviations indicate that the results are consistent over the five independently trained networks.

Individual classification networks. Next, we consider average results for the five trials of independently trained individual classification networks. For each trial, the best performing input group-network design combination for the validation database was chosen for evaluation on the test database. Thus, each trial could have a different input group-network design combination. The results for single-output networks are summarized in Table 6, where the best performing input groups and network design methods are identified. The overall results are consistent with the results in the previous section: the *AB* network is 98.4% correct, the *DIST* network 95.1%, and the *MPM* network 76.2%. Also, the values of the standard deviation of the efficiency, $\sigma\left(\eta_{ov}\right)$, are quite low. In general, the single-output networks provide consistent performance for each individual class ($N, D, P,$ and $P + D$). However, the *MPM* network classifies disturbance data poorly with only 43.1% correct. Although the best input group and design method vary, there is often a predominant input group and network design, and the performance is relatively consistent for these combinations.

Combined networks. Test data results are presented in Table 7 for the best combined network, based on validation set performance. The results were obtained in the following way. All possible network design and input group combinations for both the *AB* and *MPM* networks were compared based on their validation database accuracy at correctly detecting model-plant mismatches. The best combination was then evaluated on the test database. As an alternative, one could also use the individual networks with the best validation performance, but for this case study, the validation and test data results were better using the best combined networks. In Table 7, h_i represents the percent correct for each class i in the pool of classified results; 'ov' represents the overall percent correct for the pool. Also, $\mathbf{C_i}$ is the percentage of class i in the pool, except for 'ov', which is the percentage of the total data that were classified. By using the combined approach, the overall percent of correct cases for a "Plant change/No plant change" decision improved from 76.2% for the individual *MPM* network to 82.2%. However, in order to achieve this modest increase in performance, 27.8% of the test data were not classified. Type I errors also increased compared to the individual *MPM* network, 16.1% vs. 14.8%. The main cause of the classification errors for the combined network approach is that it classifies a large amount of the D test data instead of labeling them as "Not Classified". This result is incorrect because this classifier has only two classifications, either "No plant change" or "Plant change". No entry for η_D is included in Table 7 because it would not be applicable for this type of classifier.

Exclusion strategy. The exclusion strategy of Section 4 was evaluated using the individual *AB* and *DIST* networks that had the best validation performance for each trial. These results are shown in Table 8. Almost all of the N and D datasets are classified correctly, and 86.7% of the P cases are correctly labeled as a plant change. The majority of the misclassified cases are classified incorrectly as a disturbance (D). The efficiency is $\eta_P = 86.7\%$, while the accuracy is relatively high, $p(P) = 87.4\%$. Thus, the false alarm rate for a plant change-only condition has now been reduced to $FAR(P) = 12.6\%$ from $E_1 = 14.8\%$ for the *MPM* classifier in Table 6. Also, $FAR(P, P + D)$ is very low, 6.4%.

The results in Table 8 indicate that non-classified datasets are not an issue, unlike the situation in Table 7 for the combined network approach. Also, the $P + D$ datasets, while not classified as plant changes, are almost always classified as disturbances, which is, in a sense, correct. This situation is much more desirable than having them misclassified as "Normal". As for the combined network approach, one could design the exclusion classifier by choosing the combination of input group and network design method that maximized

the performance for the validation database. However, based on comparing validation set performance, the best approach (based on the average of efficiency and accuracy) for the case study is to use the best individual networks.

Four-output network. An advantage of the four-output network is its ability to discard invalid classifications. That is, if more than one output has a value of one, the classification is meaningless because the test data can belong to only one of the four classes. This approach is called *Smart Classification*, as opposed to *Standard Classification* where invalid results are not discarded. However, the test database results for the case study in Table 9 indicate that very few of the classifications were actually invalid. Regardless of whether the invalid classifications are taken into account, the performance is about the same. Overall, the four-output network achieves about 73% correct, with the D datasets causing the most difficulty, because only about 52% were correctly classified. A disadvantage of the four-output network is the much slower training speed. Tables 10 and 11 present a detailed error analysis and a comparison of the actual and indicated classifications, respectively.

Model-error diagnosis. After a model-plant mismatch has been detected, it is very desirable to diagnose the specific sub-models (if any) that are most affected. This task is very difficult, but important. For the Wood-Berry column case study, four single-output networks were used to diagnose plant changes in the four corresponding SISO models. These networks were developed using the plant change only (P) data from the same training and validation databases used for the detection networks previously described. Using actual P data for training is a simpler and faster approach than using classified P data from the exclusion strategy classifiers. The resulting networks were then evaluated on two sets of test data, (1) the actual P data in the test database, and (2) the indicated P data from the exclusion strategy classifiers.

The results of the model error diagnosis are presented in Table 12. For the actual P data, the diagnosis achieves an average of over 80% correct for three out of the four sub-models. Note that the plant changes in the test database include simultaneous changes in more than one sub-model, as described in Section 5.1. Also, the results are fairly consistent across the five trials, as indicated by the standard deviations. The model error diagnosis is also quite good for the indicated P data in Table 12, although somewhat less accurate than for the actual P data, as would be expected.

The authors have also investigated the effect of reducing the amount of training data for the various NN

classifiers.[45] The classifier performance was still very satisfactory even when the size of the training database was reduced from 2000 to 500 datasets. For example, η_{ov} values for the *AB* classifier only decreased from 98.4% to 97.8%, while the false alarm rate $FAR(P, P + D)$ in Table 8 increased from 6.4% to 14.4%.

6 Summary

A novel methodology has been proposed for classifying the performance of model predictive control systems, based on pattern classification and neural networks. The methodology is based on the premise that MPC system operation can be classified as one of four classes: Normal, Disturbance, Plant Change, or Plant Change + Disturbance. (Sensor faults are considered to be equivalent to output disturbances.) A simulated database is used to train neural network classifiers that determine whether or not an unusual disturbance, a significant plant change, or both are present. After a plant change is detected, other classifiers are used to diagnose which sub-models are inaccurate. The simulated database consists of windows of closed-loop MPC data that contain a wide variety of random disturbances and perturbed plant models that represent the four operating classes. Features are extracted from the simulated data and used as inputs to the neural networks. The proposed monitoring methodology has provided accurate, reproducible results for a detailed, simulated case study using the Wood-Berry distillation column model. The methodology could distinguish between normal and abnormal MPC performance for 98% of the independent test cases, an average value for five replicate trials. The proposed methodology detected abnormal disturbances correctly for 95% of the test cases. Model-plant mismatch was detected correctly for 76% of the test cases. Also, a low false alarm rate of less than 7% was achieved for detecting plant changes by using a novel exclusion strategy classifier. Once a plant change is detected, the proposed methodology can be used to diagnose the particular sub-models that are no longer accurate. In the case study, the accuracy of each sub-model was determined correctly for about 80% of the test cases.

Acknowledgements

Financial support provided by ChevronTexaco Research and Technology Company, Richmond, CA, and technical support from Ron Sorensen, Dave Anderson, and Terrell Touchstone are gratefully acknowledged. The authors would also like to thank anonymous reviewers for their encouraging comments and helpful

suggestions. Lastly, the first author, a Lehigh University Chemical Engineering alumnus (1998), would like to thank Professor Bill Luyben for serving the Lehigh community so well for so many years. ("In = Out" will most definitely never be forgotten!)

An earlier version of this paper was presented at the 2001 AIChE Annual Meeting in Reno, NV (November, 2001).

Literature Cited

(1) Qin, S. J.; Badgwell, T. A. A Survey of Industrial Model Predictive Control Technology. *Control Eng. Practice*. **2003**, *11*, 733.

(2) Maciejowski, J. M. *Predictive Control with Constraints*; Prentice Hall: Harlow, England, 2002.

(3) Sorensen, R. C.; Cutler, C. R. LP integrates economics into dynamic matrix control. *Hydrocarbon Proc.* **1998**, *Sept.*, 57.

(4) Qin, S. J.; Li, W. Detection and Identification of Faulty Sensors in Dynamic Processes. *AIChE J.* **2001**, *47*, 1581.

(5) Huang, Y.; Gertler, J.; McAvoy, T. J. Sensor and actuator fault isolation by structured partial PCA with nonlinear extensions. *J. Process Control*. **2000**, *10*, 459.

(6) Morari, M.; Ricker, N. L. *Model Predictive Control Toolbox User's Guide*; The MathWorks, Inc.: Natick, MA, 1995.

(7) Lundström, P.; Lee, J. H.; Morari, M.; Skogestad, S. Limitations of Dynamic Matrix Control. *Comp. Chem. Eng.* **1995**, *19*, 409.

(8) Kozub, D. J. Controller performance monitoring and diagnosis. An industrial perspective. In *Proc. 15th IFAC Triennial World Congress*, Barcelona, Spain, 2002.

(9) Harris, T. J.; Seppala, C. T. Recent developments in controller performance monitoring and assessment techniques. In *Proc. Chem. Process Control-VI, AIChE Symposium Series*. 2002, *98*, 208.

(10) Qin, S. J. Control performance monitoring — a review and assessment. *Comp. Chem. Eng.* **1998**, *23*, 173.

(11) Bezergianni, S.; Georgakis, C. Controller performance assessment based on minimum and open-loop output variance. *Control Eng. Practice*. **2000**, *8*, 791.

(12) Huang, B. Minimum variance control and performance assessment of time-variant processes. In *Proc. IFAC ADCHEM Sympos.*, Pisa, Italy, 2000, 177.

(13) Ko, B.-S.; Edgar, T. F. Assessment of Achievable PI Control Performance for Linear Processes with Dead Time. In *Proc. Amer. Control Conf.*, Philadelphia, PA, 1998, 1548.

(14) Shinskey, F. G. How good are our controllers in absolute performance and robustness? *Meas. Control*. **1990**, *23*, 114.

(15) Swanda, A. P.; Seborg, D. E. Controller performance assessment based on setpoint response data. In *Proc. Am. Control Conf.*, San Diego, CA, 1999, 3863.

(16) Huang, B.; Shah, S. L. *Performance Assessment of Control Loops: Theory and Applications*. Springer: London, 1999.

(17) Harris, T. J.; Seppala, C. T.; Desborough, L. D. A review of performance monitoring and assessment techniques for univariate and multivariate control systems. *J. Process Control*. **1999**, *9*, 1.

(18) Ko, B.-S.; Edgar, T. F. Performance assessment of multivariable feedback control systems. *Automatica*. **2001**, *37*, 899.

(19) Sourlas, D. D.; Manousiouthakis, V. Best achievable decentralized performance. *IEEE Trans. Automatic Control*. **1995**, *40*, 1858.

(20) Tyler, M. L.; Morari, M. Propositional logic in control and monitoring problems. *Automatica*. **1999**, *35*, 565.

(21) Kesavan, P.; Lee, J. H. Diagnostic Tools for Multivariable Model-Based Control Systems. *Ind. Eng. Chem. Res.* **1997**, *36*, 2725.

(22) Wan, S.; Huang, B. Robust performance assessment of feedback control systems. *Automatica*. **2002**, *38*, 33.

(23) Samyudia, Y.; Kint, E.; de Jong, P. Data-driven Approach to Process/Model Mismatch Compensation for Model-based Controllers. AIChE Ann. Mtg., Reno, NV, November 2001; Paper 275a.

(24) Leung, D.; Romagnoli, J. Real-time MPC supervisory system. *Comp. Chem. Eng.* **2000**, *24*, 285.

(25) Zhang, Y.; Henson, M. A. A Performance Measure for Constrained Model Predictive Controllers. In *Proc. European Control Conf.*, Karlsruhe, Germany, 1999, CM-12.

(26) Kadali, R.; Huang, B.; Tamayo, E. C. A Case Study on Performance Analysis and Trouble Shooting of an Industrial Model Predictive Control System. In *Proc. Amer. Control Conf.*, San Diego, CA, 1999, 642.

(27) Vishnubhotla, A.; Shah, S. L.; Huang, B. Feedback and feedforward performance analysis of the Shell industrial closed loop data set. In *Proc. IFAC ADCHEM Sympos.*, Banff, AB, Canada, 1997.

(28) Miller, R. M.; Huang, B. Perspectives on multivariate feedforward/feedback controller performance measures for process diagnosis. In *Proc. IFAC ADCHEM Sympos.*, Banff, AB, Canada, 1997.

(29) Haarsma, G.; Nikolaou, M. Multivariate controller performance monitoring: lessons from a snack food process. *submitted to J. Process Control*. **2002**.

(30) Hugo, A. Process controller performance assessment: several recently developed techniques make the task easier. *Hydrocarbon Processing*. **2000**, *April*, 85.

(31) Huang, B.; Kadali, R.; Zhao, X.; Tamayo, E. C.; Hanafi, A. An investigation into the poor performance of a model predictive control system on an industrial CGO coker. *Control Eng. Practice*. **2000**, *8*, 619.

(32) Kadali, R.; Huang, B. Trouble shooting and performance audit for a multivariable control system: an industrial case study. In *Proc. Engr. Solns. for Next Millennium, 1999 IEEE Can. Conf.*, Edmonton, AB, Canada, 1999, 1570.

(33) Huang, B.; Tamayo, E. C. Model valiation for industrial model predictive control systems. *Chem. Eng. Sci.* **2000**, *55*, 2315.

(34) Huang, B. Multivariable model validation in the presence of time-variant disturbance dynamics. *Chem. Eng. Sci.* **2000**, *55*, 4583.

(35) Basseville, M.; Nikiforov, I. V. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall: Englewood Cliffs, NJ, 1993.

(36) Patwardhan, R. S.; Shah, S. L. Issues in performance diagnostics of model-based controllers. *J. Process Control*. **2002**, *12*, 413.

(37) Patwardhan, R. S. Studies in Synthesis and Analysis of Model Predictive Controllers. Ph.D. Dissertation, University of Alberta, Edmonton, Canada, 1999.

(38) Shah, S. L.; Patwardhan, R.; Huang, B. Multivariate Controller Performance Analysis: Methods, Applications, and Challenges. In *Proc. Chem. Proc. Control-VI, AIChE Symposium Series*. 2001, *98*, 190.

(39) Schäfer, J.; Çinar, A. Multivariable MPC performance assessment, monitoring, and diagnosis. In *Proc. 15th IFAC World Congress*, Barcelona, Spain, 2002.

(40) Ko, B.-S.; Edgar, T. F. Performance Assessment of Constrained Model Predictive Control Systems. *AIChE J.* **2001**, *47*, 1363.

(41) Harris, T. J.; Boudreau, F.; MacGregor, J. F. Performance assessment of multivariable feedback controllers. *Automatica*. **1996**, *32*, 1505.

(42) Aspen Technology Inc. *AspenWatch*^{TM} brochure. 2000.

(43) Honeywell, Performance Monitor Profit Suite Toolkit manual. 1999.

(44) Haykin, S. *Neural networks : a comprehensive foundation, 2nd ed.*; Prentice-Hall: Upper Saddle River, NJ, 1999.

(45) Loquasto, F., III; Seborg, D. E. Monitoring Model Predictive Control Systems Using Pattern Classification and Neural Networks. Technical Report PC-108a/2002; University of California: Santa Barbara, CA, 2002.

(46) Looney, C. G. *Pattern Recognition Using Neural Networks Theory and Algorithms for Engineers and Scientists*; Oxford University Press: New York, 1997.

(47) Rengaswamy, R.; Venkatasubramanian, V. A fast training neural network and its updation for incipient fault detection and diagnosis. *Comp. Chem. Eng.* **2000**, *24*, 431.

(48) Leonard, J. A.; Kramer, M. A. Diagnosing dynamic faults using modular neural nets. *IEEE Expert*. **1993**, *8*, 44.

(49) Skogestad, S. Simple analytic rules for model reduction and PID controller tuning. *J. Process Control*. **2003**, *13*, 291.

(50) Box, G. E. P.; Jenkins, G. M.; Reinsel, G. C. *Time Series Analysis Forecasting and Control*, 3rd *ed*; Prentice-Hall: Englewood Cliffs, NJ, 1994.

(51) Kozub, D. J.; Garcia, C. E. Monitoring and diagnosis of automated controllers in the chemical process industries. AIChE Annual Meeting, St. Louis, MO, November 1993.

(52) Box, G. E. P.; Luceño, A. *Statistical Control by Monitoring and Feedback Adjustment*; Wiley: NY, 1997.

(53) Wood, R. K.; Berry, M. W. Terminal composition control of a binary distillation column. *Chem. Eng. Sci.* **1973**, *28*, 1707.

(54) The MathWorks, Inc. *Signal Processing Toolbox, Ver. 5.1*. Natick, MA, 2001.

(55) Demuth, H.; Beale, M. *Neural Network Toolbox User's Guide*. The MathWorks, Inc.: Natick, MA, 1998.

Appendix Simulation of plant changes

The following approach was used to ensure that a wide variety of plant changes were present in the simulated database. Specify:

1. The ratio n_r/n_{tot}, where n_r is the number of simulated datasets in which r out of the n models are perturbed, and n_{tot} is the total number of plant change datasets. For the case study, n_r/n_{tot} is the same for all values of r; thus for $n = 4$ models, $n_r/n_{tot} = 0.25$.

2. The manner in which $r \in \{1, 2, \ldots, n\}$, the number of models to be perturbed, is chosen. For example, in the case study, r was chosen randomly.

3. How the *number* of perturbed model parameters $g_i' \in \{1, \ldots, g_i\}$, where g_i is the total number of parameters in model i, is distributed. In the case study, $g_i = 3$, and g_i' was chosen randomly to be one, two, or three.

4. How the choice of *which* g_i' parameters are perturbed is made. In the case study, the specific parameters to be perturbed were chosen randomly.

5. A range of perturbation magnitudes for each model parameter.

After these decisions have been made, the distribution of plant changes in the closed-loop database is specified.

List of Figures

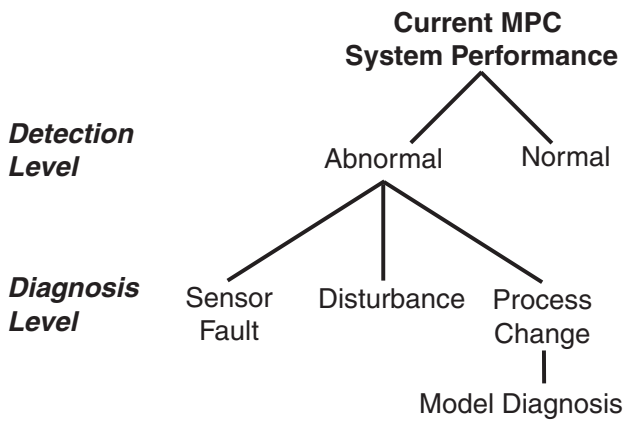

Figure 1: Overall strategy for monitoring MPC systems.

		Disturbance NN	
		No Disturbance	Disturbance
Normal / Abnormal NN	Normal	**CLASSIFY:** ***Normal***	**NOT CLASSIFIED**
	Abnormal	**CLASSIFY:** ***Plant Change***	**CLASSIFY:** ***Disturbance***

Figure 2: Illustration of the exclusion strategy for improved plant change detection.

		Plant Change NN	
		No Plant Change	Plant Change
Normal / Abnormal NN	Normal	**CLASSIFY:** ***No Plant Change***	**NOT CLASSIFIED**
	Abnormal	**NOT CLASSIFIED**	**CLASSIFY:** ***Plant Change***

Figure 3: Illustration of the combined network approach for improved plant change detection.

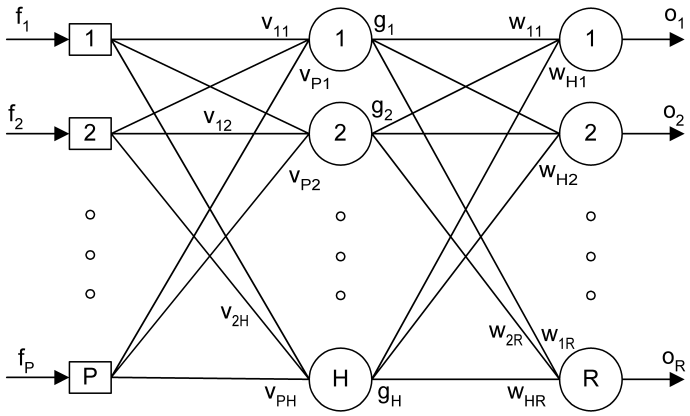

Figure 4: A generic, feedforward, multi-layer perceptron.

List of Tables

Table 1: Features considered as potential neural network inputs.

Category	*Features*
Basic Statistics	Mean, Std. Dev., Total Variation
Correlation	ACF, PACF, Variogram
Performance Metrics	Rise Time, Settling Time, Overshoot, IAE
Variables	u, y, ϵ, CE, Δu

Table 2: Distribution of the operating conditions in the simulated databases.

Database	N	D	P	$P+D$	Total
Training	500	500	500	500	2000
Validation	250	250	250	250	1000
Testing	200	200	200	200	800

Table 3: Disturbances and plant changes in the simulated databases.

	Output & Input disturbances		Feed disturbance		# Models Changed:
Database	Types†	Range ($wt\%$)	Types	Range ($\%$)	1 / 2 / 3 / 4 models
Training	S,R,O,$St_{1,2}$	$\pm[0.25 - 3.25]$	—	—	125 / 125 / 125 / 125
Validation	S,R,O,St_3	$\pm[0.25 - 3.25]$	—	—	62 / 63 / 63 / 62
Testing	—	—	S,R,O,St_4	$\pm[12.5 - 37.5]$	50 / 50 / 50 / 50

† Disturbance types: S=step, R=ramp, O=sinusoidal oscillation, St_i=stochastic with transfer function i.

Table 4: Description of input feature groups used for the 14 NN input groups.

Group	No.	Mean	Std. Dev.	ACF†	PACF‡	Vario.	Ctl. Perf.	TV
1	8	I,O	I,O					
2	12	I,O,ΔI	I,O,ΔI					
3	8						O	
4	12				ΔI,ϵ,CE			
5	12			ΔI,ϵ,CE				
6	24			ΔI,ϵ,CE	ΔI,ϵ,CE			
7	20					I,O,ΔI,ϵ,CE		
8	24	I,O,ϵ	I,O,ϵ		ΔI,ϵ,CE			
9	16	I,O,ϵ	I,O,ϵ		ϵ			
10	18	I	I,ϵ	ϵ	ϵ			
11	20	I,ϵ	I,ϵ	ϵ	ϵ			
12	12	I,O,ϵ	I,O,ϵ					
13	12	I,O,ϵ						I,O,ϵ
14	16	I,O,ϵ			ϵ			I,O,ϵ

† Lags of 1 and 3 were used for Groups 5 & 6; lags of 1, 2, and 3 were used for Groups 10 & 11.

‡ Lags of 1 and 3 were used except for Groups 10 & 11 which used lags of 1, 2, and 3.

Symbols: I=input, O=output, ΔI=differenced input, ϵ=one-step ahead residuals, CE=control error

Table 5: Test database performance results (η_{ov} in %) for each input feature group. The best results are shown in boldface.

Input	#	*AB*		*DIST*		*MPM*	
Group	Inpts.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
1	8	**98.5**	0.2	95.1	2.8	71.4	1.0
2	12	98.5	0.2	93.9	2.5	76.2	1.9
3	8	93.4	0.5	**95.9**	0.3	69.2	2.2
4	12	94.2	0.3	93.4	1.2	73.1	1.6
5	12	94.0	0.4	93.9	1.0	73.2	2.7
6	24	96.1	0.4	92.8	0.9	**76.3**	0.9
7	20	86.9	0.9	81.4	0.6	72.3	1.3
8	24	97.8	0.7	95.2	0.3	74.7	1.5
9	16	98.4	0.4	93.8	1.1	74.9	2.6
10	18	97.7	1.0	94.5	1.5	74.1	3.2
11	20	97.8	0.8	93.7	1.4	75.6	1.9
12	12	98.4	0.3	93.7	2.8	**76.3**	1.5
13	12	97.3	0.6	93.6	0.8	76.1	2.3
14	16	98.0	0.4	94.0	1.7	74.1	1.7

Table 6: Test database performance results (%) for individual classification networks*.

Network	η_{ov}	$\sigma(\eta_{ov})$	E_1	E_2	η_N	η_D	η_P	η_{P+D}	**Input Group**	**NN Design**
AB	98.4	0.4	0.2	1.4	99.1	100.0	94.5	100.0	1-1-9-9-9	2-2-3-2-3
DIST	95.1	0.9	2.0	2.9	100.0	94.3	92.1	94.0	12-1-2-12-2	3-2-3-3-3
MPM	76.2	1.5	14.8	9.0	97.8	43.1	83.3	80.8	11-10-11-11-11	2-2-2-3-3

* The null hypothesis H_0 for E_1 and E_2 is defined in Section 4.

Table 7: Test database performance results (%) for combined networks.

h_{ov}	$\sigma(h_{ov})$	E_1	E_2	η_N	η_P	η_{P+D}	$\mathbf{C}_{ov}$	$\mathbf{C}_N$	$\mathbf{C}_D$	$\mathbf{C}_P$	$\mathbf{C}_{P+D}$
82.2	3.6	16.1	1.5	93.7	68.2	74.4	72.2	33.0	16.2	25.1	25.7

Table 8: Comparison of Actual and Indicated classifications using the exclusion strategy.

	Classified as (%)			
Actual Dataset	*N*	*D*	*P*	*Not Classified*
N	**99.1**	0	0.9	0
D	0	**94.3**	5.7	0
P	5.4	7.8	**86.7**	0.1
P+D	0	94.0	6.0	0
η_P = 86.7%			$p(P)$ = 87.4%	
$FAR(P)$ = 12.6%			$FAR(P, P+D)$ = 6.4%	

Table 9: Test database performance (%) of four-output network: Standard and Smart classification.

		Actual Operating Condition					Cases Not Classified			
		η_{ov}	η_N	η_D	η_P	η_{P+D}	N	D	P	$P+D$
Standard	Mean	72.8	96.8	52.0	72.8	69.7	0	0	0	0
Classification	Std. Dev.	1.9	1.1	7.2	6.8	6.0	0	0	0	0
Smart	Mean	73.3	97.1	52.4	73.6	70.0	0.6	1.4	2.0	1.0
Classification	Std. Dev.	1.9	1.1	7.2	7.3	5.9	0.9	1.1	1.4	1.0

Table 10: Test database error analysis of four-output network: Type I and II Errors (% of total)*.

Null hypothesis:		N		D		P		$P+D$	
		E_1	E_2	E_1	E_2	E_1	E_2	E_1	E_2
Standard	Mean	0.8	2.2	12.0	6.8	6.8	2.0	7.6	13.0
Classification	Std. Dev.	0.3	0.7	1.8	1.3	1.7	0.9	1.5	2.0
Smart	Mean	0.7	2.1	11.9	6.7	6.6	1.7	7.5	12.8
Classification	Std. Dev.	0.3	0.7	1.8	1.3	1.9	0.7	1.5	1.9

* The null hypotheses for E_1 and E_2 are defined in Section 4.

Table 11: Comparison of indicated and actual operating condition of test database for four-output network.

		Classified as (%)							
		N		D		P		$P+D$	
	True Class	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Standard	N	97.1	1.1	0.0	0.0	2.0	1.2	0.0	0.0
Classification	D	0.0	0.0	52.3	7.0	2.0	1.3	43.9	5.8
	P	8.9	2.7	4.8	1.4	73.6	7.5	8.1	3.7
	P+D	0.0	0.0	22.4	5.5	4.0	1.8	70.2	5.8
Smart	N	97.1	1.1	0.0	0.0	2.0	1.2	0.0	0.0
Classification	D	0.3	0.5	52.1	7.1	2.0	1.3	43.8	6.2
	P	9.0	2.7	5.0	1.4	73.2	7.4	8.4	3.5
	P+D	0.0	0.0	22.6	5.4	5.2	2.1	68.5	6.8

Table 12: Model error diagnosis based on classifying actual and indicated P datasets.

	Actual P datasets					Indicated P datasets	
Network /	% Correct		% in	Input	Design	% Correct	
sub-model	Mean	Std. Dev.	Database	Group	Method	Mean	Std. Dev.
G_{11}	81.1	3.3	60.5	8-5-5-2-8	2-2-2-3-3	77.4	4.0
G_{12}	77.6	4.1	67.5	10-1-11-7-8	1-3-3-3-1	72.5	4.0
G_{21}	81.2	0.8	62.0	12-10-8-12-9	1-3-3-3-3	75.9	1.4
G_{22}	83.7	1.0	60.0	11-8-8-6-8	1-3-1-3-3	79.2	1.9