A PATTERN-BASED APPROACH TO EXCITATION DIAGNOSTICS FOR ADAPTIVE PROCESS CONTROL

RALPH F. HINDE, JR. and DOUGLAS J. COOPER
Chemical Engineering Department, University of Connecticut, Storrs, CT 06269-3222, U.S.A.

(First received 8 December 1992; accepted in revised form 31 August 1993)

Abstract—To maintain desired controller performance in the presence of process nonlinearity and nonstationarity, linear model-based control strategies become dependent upon the regular updating of a process model. This paper explores the use of a passive adaptive algorithm which updates the process model in a closed loop by taking advantage of naturally occurring dynamic events rather than by injecting perturbations into the system to create dynamic events. Such closed-loop identification is possible, but it requires that these events contain process information that is not masked by measurement noise or unmeasured disturbances. Presented here is a pattern-based excitation diagnostic tool (EDT) that determines when sufficient process excitation exists for model updating. The EDT consists of vector quantizing neural networks (VQNs) similar to the ART2-A and a decision maker that is a simple set of rules. The VQNs are trained to recognize local dynamic behavior in the recent histories of each process variable. The decision maker uses the outputs from these VQNs to diagnose when sufficient dynamics exist for model updating. Details of the EDT are presented along with several challenging demonstrations on both simulated and real single-input single-output processes.

INTRODUCTION

Because most chemical processes are nonlinear and nonstationary, linear algorithms such as the PID controller tend to degrade in performance over time. One approach to maintain stable control in light of such process nonidealities is to de-tune the controller. This results in sluggish controller response and ultimately leads to reduced profitability.

To maintain desired controller performance in the presence of process nonlinearity and nonstationarity, linear model-based adaptive control algorithms regularly update a model of the process. The updated model parameters are then used to redesign the controller. Popular model updating schemes include recursive least squares (RLS) (Ljung and Söderström, 1983) and the batchwise regression of process data (e.g. Cooper et al., 1990).

RLS is convenient as it enables determination of process model parameters based solely on their previous values and new sampled data. However, as time progresses, RLS gradually loses its ability to track changing process character and must be periodically resensitized to incoming data. Popular methods to achieve this include variations on variable forgetting factors (Fortescue et al., 1981) and covariance resetting (Goodwin et al., 1983). However, if RLS is resensitized while the process is not experiencing sufficient input/output excitation, the resulting model parameters will not be descriptive of the true dynamic character of the process. Similarly, a process model regressed from a batch of data collected during a period of insufficient excitation will not be accurate. In either case, using a poor process model in controller design will degrade controller performance.

Success in any model-updating scheme requires that the sampled data possess dynamic input/output information that is not masked by measurement noise or the dynamic of unmeasured disturbances (Gustavsson et al., 1977). Such dynamically rich data may be intentionally produced by perturbing the process, but it is typically not desirable to disrupt production in such a manner. To avoid such disruptions, a passive adaptive algorithm is presented here. Passive model-based adaptive algorithms update process models by taking advantage of naturally occurring dynamic events rather than by injecting perturbations into the system to create dynamic events. A passive algorithm thus requires a method of determining when sufficiently rich dynamic events occur. The method will, in essence, serve as an on/off switch for the model-updating algorithm.

Simple excitation diagnostic methods have previously appeared in the literature. One approach declares excitation when the controller error leaves a pre-specified noise band (Chia et al., 1987; Kraus and Myron, 1984), defined as a multiple of the standard deviation of the process output measurement at steady state. By this method, however, the diagnosis of excitation does not consider that the controller may be sluggish and not producing sufficient dynamic information for the regression of a process model.

Of even larger consequence is when the standard deviation of the measurement noise changes with time. As the standard deviation of the noise increases, the previously specified noise band becomes too small. The dynamics of the measurement noise then drive the adaptation with potentially disastrous results. As the standard deviation of the noise decreases, the pre-specified noise band becomes too large. Dynamic events are then ignored and opportunities for adaptation are lost.

Another popular method for determining the existence of process excitation is to look for the variance of the error in the process model prediction to exceed a dead band defined in terms of the variance of the
process output measurement noise and the bounds of known disturbances (Åström and Wittenmark, 1989; Middleton et al., 1988). This requires significant \textit{a priori} process knowledge in that all potential disturbances and their bounds must be identified. Furthermore, if the model prediction during a process transient succeeds in reasonably predicting the true process output then the dead band will not be exceeded and valuable dynamic information will be ignored.

Clearly, a stable and robust method of excitation diagnostics that requires minimal \textit{a priori} process knowledge is desirable. Toward this goal, a unique method of excitation diagnostics is detailed here. In this method, patterns displayed in the process input and output variables are analyzed using vector quantizing neural networks (VQNs) similar to the ART2-A of Carpenter et al. (1991). As shown in Fig. 1, two VQNs work in tandem to evaluate short process variable history patterns in the manipulated input and measured output variables. As described later, these history patterns are compared with a "library" of exemplar patterns which display generic dynamic trends. When a process variable history pattern is classified by a VQN as being similar to one of the exemplar patterns in the library, that VQN signals to a decision maker that its respective process variables is experiencing locally dynamic behavior. The decision maker combines information from both VQNs to determine when sufficient dynamics exist to activate a modeling algorithm.

Through both simulated and laboratory process demonstrations this excitation diagnostic tool (EDT) is shown to be stable, simple to use, and reasonably robust in a variety of applications. The simulated process demonstrations consist of set point tracking and disturbance rejection for systems that display various nonideal characters including significant measurement noise and process nonlinearity, nonstationarity and model order mismatch. The laboratory process demonstration consists of set point tracking for a stirred tank in which the temperature of two mixing streams is controlled by manipulating the flow rate of the cold stream. To demonstrate that the EDT is general to any number of model-based control algorithms, both an IMC tuned PI controller with dead time compensation and the dynamic matrix control (DMC) algorithm are employed.

The scope of this work is passive model-based adaptive control of single-input/single-output (SISO), nonintegrating, minimum phase, open-loop stable processes. Although specific model-based controllers employ a specific model form in this work, the method is applicable to a number of model-based control algorithms and process model forms. The EDT only requires that the model contain a characteristic or dominant process time constant to be used in determining data window lengths and minimum durations of dynamics for diagnosing excitation. Also, the process must not be persistently excited as this will cause the data batch used in the model regression to grow excessively large.

**THE CONTROL ALGORITHMS**

**The process model**

The process model employed in controller design must adequately describe the process or class of processes that the operator wishes to control. Whereas any number of models that are popular in control theory may be used in this work, the first order plus dead time (FOPDT) model is employed here for dem-
onstration purposes. An FOPDT model reasonably describes the steady-state gain, \( K_p \), the dominant time constant, \( \tau_p \), and the dead time, \( \tau_d \), of most open-loop stable processes. Although an FOPDT model cannot accurately track the oscillatory behavior of higher-order processes, or the inverse response character of nonminimum phase processes, these three parameters provide sufficient predictive capabilities to obtain stable and reasonably robust control in a great many applications. The FOPDT model parameters are also intuitive and lend themselves to operator understanding and trust.

In difference form the FOPDT model is expressed as
\[
y'(t|t - 1) = y'(t - 1|t - 2) + a_1 \Delta y'(t - 1|t - 2) + b_1 \Delta u(t - k)
\]
\[
a_1 = \exp(-\Delta t/\tau_p) \quad b_1 = K_p(1 - a_1)
\]
\[
k = 1 + \int(\tau_d/\Delta t) + 1
\]

where \( y'(t) \) is the prediction of the measured output variable, and \( u(t) \) is the manipulated input variable.

Adaptive model-based controllers

Once a model form is chosen, the parameters of that model may be used to design any one of a number of model-based controllers. In this work, a model-based PI controller with predictor for dead time compensation and the DMC algorithm are employed for demonstration purposes. These algorithms, well documented in the literature, are summarized here.

The velocity form of the PI algorithm (Seborg et al., 1989; Smith and Corripio, 1985) computes the incremental control action at sample number \( t \), \( \Delta u(t) \), as
\[
\Delta u(t) = K_c[e'(t) - e'(t - 1) + e'(t)\Delta t/\tau_i]
\]
where \( K_c \) is the controller gain and \( \tau_i \) is the integral or reset time. To compensate for dead time, the controller error, \( e'(t) \), is predicted using
\[
e'(t) = y_{sp}(t) - y'(t + k - 1|t) - y(t) + y'(t|t - k + 1).
\]

This compensates for dead time using a future prediction of the process output and corrects for model error using the present measurement and the prediction from one dead time ago.

In this work, the PI controller is tuned using the internal model control (IMC) architecture (Rivera et al., 1986). Since \( \tau_d \) has been compensated for, only \( K_p \) and \( \tau_p \) are employed with a first-order filter to arrive at the IMC tuning relations:
\[
K_c = \frac{\tau_p}{\tau_i K_p}, \quad \tau_i = \tau_p
\]

where \( \tau_i \), the one IMC tuning parameter, is the desired closed-loop time constant. This PI controller with predictor is made adaptive by updating the process model parameters \( K_p \), \( \tau_p \), and \( \tau_d \).

The form of the DMC (Prett and Garcia, 1988; Cutler and Ramaker, 1979) algorithm employed here is the least-squares, unconstrained, single-input/single-output (SISO) implementation. At each sample, future control actions are determined that minimize the objective function:
\[
J = \sum_{j=1}^{P} \left[ y_{sp}(t + j) - y'(t + j) \right]^2 + Q \sum_{j=1}^{C} \left[ \Delta u(t + j - 1) \right]^2
\]

where \( y'(t + j) \) and \( y_{sp}(t + j) \) are the expected future process outputs and \( y_{sp}(t + j) \) is the expected future set points. This work employs the typical assumption that future set points will equal the present set point value. The variable \( P \), the length of the prediction horizon, is set equal to an estimated response time and the variable \( C \), the length of the control horizon, is set equal to one dominant process time constant, or
\[
P = (5\tau_p + \tau_d)/\Delta t \quad C = \tau_p/\Delta t.
\]

The input suppression factor, \( Q \), is set equal to \( 10K_p^2 \). This serves to cast the two terms of eq. (6) into consistent units and work toward keeping them proportionate in size. The parameter \( \Gamma \) is a tuning parameter chosen by the operator to provide desired performance.

A least-squares minimization of eq. (6) over the control horizon, \( C \), provides the solution of the future control actions:
\[
\Delta u = (B^TB + Q)^{-1}B^T[y_{sp} - y'], \quad j = 1, P
\]

where \( Q \) is a diagonal matrix of input suppression factors and \( B \) is the dynamic matrix of dimension \( P \times C \). The dynamic matrix is constructed using an open-loop step response of the FOPDT model to a unit step input. The first element of \( \Delta u \) is implemented at time \( t \) and a new vector of future control actions is calculated at the next sample. Similar to the PI controller, this DMC algorithm is made adaptive by updating the process model. The new model is then used in development of a new open-loop step response, dynamic matrix \( B \), input suppression factor \( Q \), and horizons \( P \) and \( C \).

To implement either of these model-based controllers, the parameters of the chosen model, \( K_p \), \( \tau_p \), and \( \tau_d \) in this work, must be specified at start up. For all demonstrations, these parameters are based on the regression of data collected from a single open-loop test step made in the start up operating regime. The sample time, \( \Delta t \), is set equal to 0.04 times the initial estimate of \( \tau_p \). By sampling 25 times per dominant process time constant, patterns containing only measurement noise will be less likely to display apparent dynamic trends than if the more traditional sample rate of 10 times per time constant were employed.

The IMC and DMC tuning parameters, \( \tau_i \) and \( \Gamma \), respectively, are then set to values that produce desired controller performance defined in this work as a short rise time with 10 to 15% overshoot. For the
processes employed in this work, such desired performance is obtained using a $r$, of 0.5$r$, and a $\Gamma$, of 10. Note that different values may be required or desired for other implementations. Also note that smaller values of $r$, and $\Gamma$ meant to increase controller effort and induce greater amounts of process excitation will make the controller more sensitive to measurement noise and decrease the signal-to-noise ratio in the process input. The operator must exercise caution when significantly decreasing these parameters as doing so may mask process excitation.

**THE EXCITATION DIAGNOSTIC TOOL**

Process excitation diagnostics must determine when sampled data possesses input/output dynamics that contain process information that is not masked by measurement noise or unmeasured disturbances. Toward this goal, VQNs, which are proven tools for pattern recognition, are used to form the basis of the EDT.

*Vector quantizing networks*

Classical vector quantizing algorithms perform data compression where high-dimensional data vectors are encoded to low-dimensional data vectors to reduce storage and transmittance requirements. Upon receiving or recalling the compressed data, a "library" of codes uses that data to recreate the original vector. The goal is to minimize any distortion or error between the original and recreated data vectors (Gray, 1984).

The type of neural network employed here is called a VQN because it implements a vector quantizing algorithm. Incoming patterns are encoded into scalar pattern classification numbers, effectively compressing the pattern. Examples of VQNs are Kohonen's network (1982) and the adaptive resonance theory (ART) networks of Carpenter et al. (1991) and Carpenter and Grossberg (1987).

The VQN, although similar in some respects to the popular backpropagation network (BPN) (Rumelhart and McClelland, 1986), has a different architecture and therefore possesses different abilities than the BPN. Both networks are comprised of processing units called nodes. These nodes take a weighted sum of their inputs and pass this sum through either a linear or nonlinear function.

The connectivity of the nodes determines the architecture of the network. Unlike the BPN, the VQN architecture has only one layer of nodes where each node receives the entire incoming pattern and produces a network output. The architecture defines the functionality of the network. Whereas the nodes of a BPN make a collective effort to learn a functional relationship, each node of a VQN makes an individual effort to learn to classify a discrete pattern class. The functionality of the network determines how it must be trained. During training, a BPN must be supervised by providing a desired output for each input pattern. VQN training is unsupervised as it involves clustering similar input patterns together using no *a priori* knowledge of the separate pattern classes.

Both the VQN and BPN are equally capable pattern recognition tools for certain applications (Cooper et al., 1992). This issue that really separates the two networks for this strategy lies in the fact that a BPN, once online, has no inherent method for evaluating its pattern recognition performance. A VQN not only yields a pattern classification, but also produces a matching score which may be used as a measure of performance or reliability of the classification.

*The VQN architecture*

As previously mentioned, VQNs are comprised of many nodes where each node receives the entire incoming pattern, takes a weighted sum of its inputs and produces an output of the network. The weights by which the inputs to each node are multiplied form the exemplar pattern for that node. The output that the node produces is the matching score between the incoming pattern and that exemplar pattern. This is presented in vector form.

Incoming patterns in this work consists of the recent histories of the process variables. An incoming history pattern may be expressed as a vector whose elements are sampled data recast as perturbations from the initial element:

\[
i^T = [i(t - M + 1)i(t - M + 2) \ldots i(t)].
\]

Each element is an input to every node in the network and $M$ is the total number of elements in the history pattern. The exemplar pattern for each node $j$ is also expressed in vector form as

\[
z_j^T = [z_j(1)z_j(2) \ldots z_j(m)].
\]

To maintain a consistent scale for the matching scores, all vectors are normalized to a unit length. Normalization of given vector $x$ is written as $X = \eta x$, where

\[
\eta x = \frac{x}{\|x\|}
\]

and $\|x\|$ denotes the norm or Euclidean distance of vector $x$. Hence, the operator $\eta$ performs Euclidean normalization.

Once all vectors are normalized, the matching scores, $T_j$, may be calculated. These scores represent a measure of how similar each exemplar pattern is to the history pattern. Again, each node accomplishes this by taking a weighted sum of its inputs. This is identical to the inner product of the incoming and exemplar pattern vectors:

\[
T_j = \eta^T z_j.
\]

Due to the Euclidean normalization of the vectors, a perfect match has a matching score of 1.0. Likewise, for two patterns that are mirror images of each other the score is $-1.0$. 

The node that wins the classification is determined using the choice function:

\[ T_j = \max_j |T_j| \]  

(13)

where the absolute value of \( T_j \) allows successful classification of mirror image patterns effectively doubling the number of patterns recognized by the network. The node with the largest matching score may not win the classification, though, if the magnitude of \( T_j \) is small indicating a weak match. This is determined through a vigilance test that compares the value of \( T_j \) to a vigilance parameter \( \rho \), where \( 0 \leq \rho \leq 1 \). If \( T_j > \rho \) then there is a strong match and classification is successful. If \( T_j < \rho \) then there is a weak match and no classification is declared.

For online classification, network operation for a single pattern presentation ends after the vigilance test. The network is then prepared to receive the next incoming pattern to be classified. During training, however, network operation for a pattern presentation continues as long as the vigilance test is passed or there are available empty nodes. If \( T_j \) fails the vigilance test, then the incoming history pattern is unlike any existing exemplar patterns and a new exemplar pattern is formed by setting the weights of a new node equal to the elements of \( I \). If \( T_j \) passes the vigilance test then that node's exemplar pattern, \( Z_j \), is updated as

\[ Z_j = \eta[\beta I + (1 - \beta)Z_j] \]  

(14)

where \( \beta \) is the learning parameter set as \( 0 \leq \beta \leq 1 \). A \( \beta \) close to zero produces a slow learning rate but develops exemplar patterns that better represent the training patterns clustered to create them.

During training, the value of \( \rho \) determines the number of exemplar patterns formed which reflects how finely the pattern space has been partitioned. A large vigilance parameter very close to 1.0 requires that clustered patterns be very nearly identical. This will result in fewer clusterings and more exemplar patterns which reflects a more finely partitioned pattern space. A small vigilance parameter allows the clustering of patterns that are less alike. This will result in a small number of exemplar patterns and a more coarsely partitioned pattern space.

**Signal-to-noise ratio and matching score**

The matching scores, \( T_j \), are used in training and online classification to determine if an incoming process variable history pattern is similar to an exemplar pattern. Since the exemplar patterns represent generic dynamic trends, if \( T_j \) passes the vigilance test, then that process variable is declared to be displaying a dynamic trend. The matching score may also be used to make judgements about the signal-to-noise ratio of the data within the incoming history pattern then the matching score will be 1.0. As the history pattern is corrupted with larger amounts of measurement noise, the matching score will decrease and eventually fail the vigilance test. In this manner, the matching score and vigilance test may also be used to decide whether the process information in the history pattern is masked by measurement noise.

This relationship is shown in Fig. 2(a)–(c). Here, a normalized exemplar pattern showing a dynamic trend, plotted vs sample number, is shown in three separate graphs as a dashed line. The same pattern corrupted with random error of increasing standard deviation is presented as solid lines in Fig. 2(a)–(c). Note that these patterns are 25 samples or one dominant process time constant in duration. Again, if there were no noise present in each case, the matching scores, \( T_j \), would be 1.0 and the signal-to-noise ratios, \( S/N_j \), would be infinity. However, the small amount of noise in Fig. 2(a) results in \( T_a = 0.998 \) and \( S/N_a = 19.3 \). As the noise increases in Fig. 2(b) and (c) the signal-to-noise ratios decrease where \( S/N_b = 4.8 \) and \( S/N_c = 0.8 \). This further decreases the matching scores such that \( T_b = 0.988 \) and \( T_c = 0.872 \). It can be seen that, indeed, there is a relationship between the matching scores and signal-to-noise ratios for these otherwise identical patterns.

With such a relationship, the matching scores from a few test patterns such as those in Fig. 2(a)–(c) may be

![Fig. 2. Relationship of matching score, \( T_j \), and signal-to-noise ratio, \( S/N_j \).](image-url)
used to determine an initial estimate of the vigilance parameter. A large value for $\rho$ very close to 1.0 would require that there be very little noise in the process variable history patterns as in Fig. 2(a). Likewise, a small value of $\rho$ would allow more noise in the process variable history patterns as in Fig. 2(c). The amount of noise present in Fig. 2(c), however, masks the dynamic information in that process variable history pattern. As such, in this work, an appropriate estimate of the vigilance parameter is above $T_{b}$ or 0.872. A $\rho$ between $T_{b}$ and $T_{d}$ of 0.92 was determined through trial and error to give a good balance between the number of exemplar patterns formed (computational load) and the rejection of patterns containing excessive noise. This value of 0.92 is also employed by Carpenter et al. (1991). The optimum or desired value of the vigilance parameter may be different for other applications.

Network training patterns

Whereas an FOPDT model is sufficient for controller design, its inability to accurately describe the oscillatory behavior of higher-order processes makes it insufficient for training pattern generation. To successfully classify oscillatory process variable history patterns, a higher-order process model must be chosen. Since the oscillatory behavior of most chemical processes can be adequately modeled as second order plus dead time (SOPDT), an SOPDT model, realized as two FOPDT models in series, is used to develop the VQN training patterns. Note that if a specific process contains characteristics not adequately described by an SOPDT model, then these characteristics should be included in the training model.

The values of the VQN parameters set here for network training are implemented later for online classification. In order to achieve representative exemplar patterns while training the network at a reasonable rate, the learning parameter, $\beta$, is set to 0.10 initially and is decreased linearly with time. As explained previously, the vigilance parameter, $\rho$, is set to 0.92. Short-term process history patterns are defined as being one dominant process time constant in duration. For an FOPDT model, this $\tau_p$. Since $\Delta t = 0.04\tau_p$, $M$ is equal to 25.

The model-based control algorithm must be chosen before the training patterns are generated because patterns displayed in the process input reflect the character of the controller more than the character of the process. The model-based controller of choice is implemented on the chosen model and a random series of set point changes are made that are between one and two dominant time constants in duration. As the set point is randomly changed, the parameters of the chosen model are varied. For the SOPDT model employed for pattern generation in this work, $K_p$, $\tau_{p1}$, and $\tau_{p2}$, are cycled between 1 and 3 times their original values. The dead time, $\tau_d$, is cycled between zero and 12$\Delta t$, one-half of the dominant time constant. This procedure produces a variety of response patterns with varying degrees of controller performance from slightly sluggish to very aggressive. Since neural networks can only interpolate between patterns that they have been trained on, the operator must be careful to make sure that the pattern variations and resulting pattern space are meaningful and representative of the process to be controlled and the controller to be employed.

Process variable history patterns are collected every five samples and sent to their respective VQNs. The VQNs group or cluster similar history patterns together forming the exemplar patterns as described earlier. Training stops for an individual VQN once no new nodes are committed. At that time the VQN is able to recognize 100% of the patterns in the training pattern space. This is confirmed by allowing the network to continue receiving and classifying training patterns over at least a full cycle of model parameter variations. As the pattern generation is random, these additional patterns are not identical to those used to train the network.

Since the model-based controller of choice must be used in generation of the “library” of exemplar patterns, a set of process input and process output dynamic trend libraries will exist for each model-based controller. Exemplar patterns in this work are therefore generated for both the DMC algorithm and the IMC tuned PI controller with predictor. For a vigilance parameter of 0.92, there resulted 73 input and 29 output exemplar patterns for the DMC and 358 input and 77 output exemplar patterns for the IMC tuned PI controller with predictor. The PI controller requires more patterns because it makes more instantaneous and drastic changes in the process input than the DMC or other long-range predictive controllers. This results in more variation in the process variable pattern spaces.

The VQNs are now capable of recognizing process variable dynamic trends for the two demonstration controllers when implemented on processes adequately described by an SOPDT model and while displaying the range of controller performances experienced during training. Likewise, any model-based controller may be employed with any appropriate process model and model parameter variations to train the networks on patterns expected to be found. Of course, given the nature of the random pattern generation and the effects of the vigilance and learning parameters on exemplar pattern formation, some trained VQNs may do a better job than others of identifying dynamic trends in process variable histories. The performance of the excitation diagnostics must be balanced against the size and computational load of the VQNs.

The decision maker

The decision maker is a simple rule base system that receives results of the classifications from the VQNs and determines both when the process model should be updated and which data should be used. The decision maker looks for relatively simultaneous
multiple dynamic trend classifications from both VQNs by keeping a running sum of the results of the vigilance tests. A passed test is assigned +1 and a failed test is assigned −1. The minimum sum is zero. A process variable dynamic state is defined when that variable’s running sum reaches a trigger value. The trigger value represents a characteristic or dominant process time constant’s duration of dynamics. In this work, the trigger value is set equal to five since the VQNs are activated every five samples and \( \Delta t = 0.04 t_p \). A dynamic flag for that process variable is then set to one and its running sum is reset to zero.

When both process variable dynamics flags are equal to one within one dominant time constant’s duration of each other to allow for dead time between the process variable dynamics, sufficient dynamics for process model updating have been found and the modeling algorithm is activated. In this work, the modeling algorithm is a batchwise regression of eqs (1) and (2) to process input and output data minimizing the sum of squared errors between the predicted and the actual process output trajectories. Process input and output data for the model regression are collected from the present back to the last steady state. A steady state is declared when either both process variable running sums are simultaneously equal to zero for at trigger value’s duration or when one running sum is equal to zero for one estimated response time as long as neither process variable is saturated. To help insure proper characterization and converged process model parameters, an additional model regression is performed just prior to the first steady state after a dynamic event.

Process variable saturation occurs when a variable is constrained by the maximum or minimum value of an actuator or sensor. This causes an apparent steady state in that variable. From an excitation point of view, such an apparent steady state may not truly reflect the dynamic state of the other process variable so process steady states are not declared when either process variable is saturated. From a modeling standpoint, input saturation has little to no effect on process modeling. Output saturation, however, masks the true state of the process because the process output is above or below the range that the sensor can measure. For these reasons, process modeling is performed regardless of any process input saturation during the dynamic event, but process modeling is not performed if the process output is saturated during the dynamic event.

Using these rules, a set point response event, for example, may be modeled 2 or 3 times during its duration. Each modeling instance will contain more information than the last until a new steady state is found. This is demonstrated in Fig. 3. Here the IMC tuned PI controller with predictor is implemented on a second-order simulated process. Figure 3(a) shows the process output, \( y(t) \), and the running output dynamic sum plotted against sample number \( t \). Figure 3(b) shows the process input, \( u(t) \), and the running input dynamic sum plotted against sample number \( t \).

Fig. 3. Excitation diagnostic tool with PI controller showing process variable running dynamic sums for (a) process output and (b) process input.

The process output is corrupted with a small amount of random error to simulate measurement noise.

At the start of this demonstration, the process is at steady state and only measurement noise causes slight variations in the process variables. As there are no true dynamics present in the process variables during the first 100 samples, no process variable history patterns are classified as being dynamic and neither the output nor the input dynamic sums are incremented above zero. This leads to the identification of steady state at samples 50, 75 and 100.

At sample 100, a step change in set point is introduced and the controller responds accordingly. The controller response is made slightly aggressive for this demonstration by underestimating the value of the process gain. Dynamics are found in both process variables as the controller moves the process to a new steady state in order to achieve the new set point. This leads to the incrementing of the dynamic sums and a global dynamic state is declared on two occasions. This results in two process model regressions and subsequent process model parameter updates at samples 130 and 155. For each adaptation, data for model updating are collected from the sample at which the global dynamic state is found back to the last steady state at sample 100.

Additional dynamics are found by both VQNs as the process response to the set point change further dampens beyond sample 155. Although the process input dynamic sum reaches a trigger value signaling that sufficient dynamics exist in the process input, the process output dynamic sum falls short of the trigger
value and is decremented to zero. This is an indication that the dynamics in the process output are becoming masked by the measurement noise as the response dampens.

The last model regression is performed just prior to the next steady state at sample 210. That steady state is declared at sample 215 and additional steady states are found at samples 240, 265, 290, 315, 340 and 375. Apparent dynamic trends are found in the process input between samples 350 and 400 as shown by the input dynamic sum being incremented to 1 on two occasions. However, in each case, the next input VQN vigilance test is failed and the input dynamic sum is returned to zero.

**Model adaptation**

Once the dynamic data have been modeled, the new model parameters must be tested for accuracy and convergence before they are implemented. The EDT identifies when dynamics exist that are not masked by measurement noise in both process variables, but is unable to distinguish between dynamics that are driven by the controller and dynamics that are driven by unmeasured disturbances. As such, the EDT turns on the modeling algorithm in an attempt to update the process model during disturbance rejections. Unfortunately, the dynamics of the disturbance corrupt process information, resulting in inaccurate process model parameters. For this reason, each new process model must be tested for accuracy and convergence before it may be implemented.

The first step in determining the validity of a new model is to consider the new estimate of the process gain. Vogel and Edgar (1982) note that the value of the new estimate of $K_p$ provides an indication of the validity of the new model parameters. In this work, the sign of the new estimate of the process gain is checked against the sign of the presently implemented process gain estimated. If the signs are opposite then the model is determined to be invalid due to corruption by unmeasured disturbance dynamics. Further process variable dynamics are then ignored until the disturbance driven event is determined to be over at the next steady state.

The gain estimate sign change phenomenon is described in the following example. For a positive gain process, as the process input is increased, the process output will increase. The two process variables ultimately move in the same direction. However, when a disturbance causes the process output to increase, the controller will decrease the process input in an effort to reject the effect of that disturbance and maintain the process output at set point. The process variables are now moving in opposite directions. If a model were to be regressed on such data, a negative process gain estimate would result. Likewise, the time constant and dead-time estimates would be influenced by the dynamics of the disturbance. This occurs at all disturbance-driven dynamic events.

If the signs of the two gain estimates are the same, then the new model is determined to be valid. Each process model parameter must then be checked for convergence as estimates will vary depending on noise levels and the length of the data histories. Convergence is checked by calculating a difference factor, $\delta_r$, between new and previous estimates of each parameter $P_r$ using

$$\delta_r = \frac{2|\hat{P}_{new} - \hat{P}_{old}|}{|\hat{P}_{new} + \hat{P}_{old}|}.$$  \hspace{1cm} (15)

If the difference factor for each model parameter is less than a convergence factor, $c_r$, then the process model parameters are declared converged. The new estimates of the process model parameters are then directly implemented in the model-based controller design equations. In this work, the convergence factor is set to 0.2 as determined by experience. A larger or smaller value of $c_r$ may be chosen if the operator wishes to allow more or less variation before declaring convergence.

Obviously, it may be argued that this model parameter updating scheme is very simple and that more complex methods may be used. The method presented here has been chosen for demonstration purposes only and is not meant to serve as the sole approach for the EDT.

**MEASUREMENT NOISE DEMONSTRATION**

A simulated second-order process is employed to demonstrate that the EDT does not require a priori information regarding the process measurement noise. The simulated process consists of two FOPDT models in series. Their two time constants are 1.0 and the overall process gain is 3.0. Six set point steps of magnitude 5.0 are made at 250 sample intervals forming a square wave trajectory.

To simulate process measurement noise, normally distributed error with zero mean is added to the process output. At sample 251, $\sigma$ decreases linearly until it equals 0.1 around sample 1200 and remains constant from then on.

Controller performance is made rather aggressive at the beginning of the trajectory by using an estimate of $K_p$ that is one-third of the true value of 3.0. Correct estimates $\tau_p$ and $\tau_d$ are used. A DMC controller is implemented with $\Gamma = 10$ and $\Delta t = 0.04t_p$.

**Fixed model DMC control**

Figure 4 shows the result of implementing a fixed model DMC controller with this simulation. Since no model updating occurs, the controller performance does not change throughout the trajectory. At the first set point step, made at sample 250, the aggressive behavior is entirely masked by the excessive measurement noise. As $\sigma$ decreases, the aggressive controller performance becomes more evident. By the fifth step, made at sample 1250, $\sigma$ has reached its minimum value revealing the excessive overshoot and moderate damping that exists at each set point response. The challenge put forth to the EDT is to determine when sufficient dynamics exist above the measurement
noise in order to make a reliable adaptation. This should be done as soon as possible without sacrificing controller performance or stability.

**Adaptive DMC control**

The results of using the EDT are shown in Fig. 5. The EDT correctly determines that the first two set point step responses contain insufficient process dynamics above the measurement noise for reliable process model updating. Furthermore, steady states are identified after each set point step response. This aids in proper data collection for use in the first modeling instance.

At sample 810, approximately two and one-half dominant time constants duration after the third set point step, sufficient excitation for process model updating is diagnosed. Accurate model parameters result from a model regression but a second regression is necessary for parameter convergence. The second regression takes place at sample 930 as a final fit is performed just prior to the next steady state. This produces converged accurate estimates of all three model parameters. The estimate of the process gain, $K_p$, is increased to 3.08 and the other two parameters are not appreciably changed. This adaptation correctly relaxes the controller’s effort despite the presence of excessive measurement noise and produces desired performance as witnessed by the response to the fourth set point step made at sample 1000.

Adaptation based on the fourth set point step response begins at sample 1040. Significant noise levels, still present at the beginning of this response, cause the model regressions to produce an estimate of $K_p$ that is slightly above the true value. This causes the response to the fifth set point step at sample 1250 to be slightly sluggish with no significant overshoot. However, two more adaptations based on the response to the fifth set point step regain accurate estimates of the model parameters as shown by the desired performance response to the sixth set point step.

**Noise band estimation**

As a comparison, an alternative method to the excitation diagnostic tool is presented. A noise band similar to that used by Kraus and Myron (1984) is employed using an online method of estimating the standard deviation of the measurement noise, $\sigma$. The method used is an exponentially weighted moving-average technique presented by Rhinehart (1991). In short, $\sigma$ is estimated using filtered measurement-to-measurement deviations, $P_j^2$, determined as

$$P_j^2(t) = \left(1 - \frac{1}{N-1}\right)(y(t) - y(t-1))^2$$

$$+ \frac{(N-2)}{N-1} P_j^2(t-1)$$

where $N$ is a filtering parameter. Rhinehart suggests using $N = 15$ to balance the desires for rapid tracking of, as well as an accurate estimate of $\sigma$. The estimate of the standard deviation of the measurement noise is then calculated as

$$\hat{\sigma}(t) = (P_j^2/2)^{0.5}.$$  

This estimate is then used to construct a noise band around the controller error.

Kraus and Myron define their noise band as an estimate of the peak-to-peak amplitude that is of higher-frequency content than the closed loop can remove. In this work, this is interpreted as $\pm 3\sigma$ around zero controller error. They then begin adaptation when the controller error exceeds twice the noise band. This translates to $\pm 6\sigma$ around zero controller error. Adaptation in that work is based on controller error peak information that is above the process noise.

Figure 6 shows the controller error from the fixed model DMC control simulation as a solid line. The estimated noise band is shown as a dotted line. Using an excitation diagnostic method similar to that used by Kraus and Myron, no adaptation would be made by any adaptive method until the fourth set point step response. Even then, if using a peak feature adaptive approach only the first error peak would be found possibly leading to an incorrect conclusion about the overshoot and damping character of the response. This could lead to an incorrect adaptation. Note that the EDT is able to lead to a correct adaptation at the third set point step response without need for information about the process measurement noise.
THIRD-ORDER PROCESS DEMONSTRATION

A third-order simulated process is used to demonstrate the adaptive method in the presence of model order mismatch, process nonlinearity, nonsustained set point changes and process nonstationarity. The simulated process consists of three FOPDT models in series. Their gains and time constants are all initially 1.0. The use of the FOPDT model for this third-order process introduces the nonideality of model order mismatch. Process nonlinearity is introduced by making the overall process gain a nonlinear function of the process input using:

\[ K_p = (0.05)(20.0)^{[0.15](0.15)} \]  \hspace{1cm} (18)

A series of set point step changes, including a series of nonsustained set point changes, moves the process into a higher gain region in the beginning of the demonstration. As the process input increases to allow the process to track the set point changes, the overall process gain increases from 1.0 to approximately 1.5. Process nonstationarity results from an unmeasured disturbance that both moves the process into a higher gain region and drives the process output with its own damping oscillations that mask the true character of the process.

The result of a batchwise regression of an open-loop step test made in the operating regime gives parameter estimates of \( K_p = 1.00 \), \( \tau_p = 1.99 \) and \( \tau_v = 1.14 \). These estimates are implemented at start up. An IMC tuned PI controller with predictor is employed with \( \tau = \tau_p/2.0 \) and \( \Delta t = 0.04\tau_p \).

Fixed model PI control

Figure 7 shows the results of implementing a fixed model IMC tuned PI controller with predictor with this simulation and illustrates the challenge offered to the EDT. At sample 250, the set point is stepped from 50.0 to 52.5. This increases the process gain only slightly as the process input must be increased to track the changing set point. The response to this set point step displays the desired performance of a 10% overshoot. At sample 300, the set point is ramped from 52.5 to settle at 57.5 at sample 550. The set point then changes to 55.5 at sample 375, 60.0 at sample 600, and finally to 57.5 at sample 650. This series of nonsustained set point changes results in significantly increasing the process gain as the process input is also significantly increased. The result is an aggressive response that is not easily characterized using peak feature or pattern recognition techniques. The set point is again stepped to 60.0 at sample 1000 to show the result of the process gain increase. The response to this step is marked by a 20% overshoot and moderate damping.

An unmeasured disturbance is introduced at sample 1250. This is a step-oscillating disturbance that both imparts a lasting change in process character by moving it to a higher operating regime, and drives the response with its own dynamics masking the true performance of the controller. The disturbance damps by sample 1375 but its effect on the fixed model controller lasts until approximately sample 1600. At sample 1750, the set point is stepped down to 57.5 to show the result of the unmeasured disturbance on controller performance. The response to this set point step is marked by an even higher overshoot and slowly damping oscillations.

The set point is further stepped down to 50.0, its start up value, at sample 2000. The process input is decreased in order for the process to track this set point change. This causes the process gain to decrease and results in a smaller overshoot although the second peak looks larger than that of the previous response. A last set point step is made at sample 2250 where the set point is increased from 50.0 to 52.5. This step is similar to the first set point step of the trajectory, but, because of the lasting change made by the disturbance, the response is more aggressive.

Adaptive PI control

The result of using the EDT for passive adaptive control of the third-order simulated process is shown in Fig. 8. The response to the first set point step at sample 250 is identical to that of Fig. 7 for the fixed model case. Two model adaptations are made during this response, the net effect of which is to slightly increase the model gain. Eight model regressions are performed during the nonsustained set point changes occurring between samples 500 and 650. Each regression results in correctly tracking the changing process.
gain while the other two model parameters are not changed appreciably. The final result is a well-behaved overall response. The next set point step at sample 1000 shows that desired controller performance has been obtained as the response is similar to that of the first set point step.

At sample 1250, the unmeasured disturbance is rejected faster than in the fixed model case. This is due to the updated process model reducing the controller's effort. Once the dynamics of the disturbance no longer drive the response, the process returns to a quiescent state. The EDT does, however, signal that process dynamics are occurring. This leads to several potential modeling instances during the disturbance rejection event. As the first regression produces a negative estimate of the process gain that fails the validity test, successive modeling instances are ignored until the next steady state is found. Hence, although there is much opportunity to lose correct estimates of the process model parameters, they are maintained as a stable controller performance. This is witnessed by the response to the next set point step at sample 1750 as it is slightly more aggressive than the previous set point step since no model adaptation has been performed during the disturbance rejection event.

The next set point step response at sample 2000 illustrates the method's ability to adapt to sluggish controller performance. The process is moved into a region of smaller process gain as the set point is stepped down from 57.5 to 50.0. Whereas this step produces an aggressive response in the fixed model case, a sluggish response results here because the estimate of the process gain is correct at the start of the response. Three model regressions are performed during this response which ultimately decrease the estimate of the process gain. These adaptations are accurate as shown by the response to the last set point step, made at sample 2250, which displays desired performance.

LABORATORY PROCESS DEMONSTRATION

The previous demonstrations have been simulations designed to exhibit process nonidealities typically encountered in real world chemical process control. Simulations, however, are still ideal in nature. To show that this adaptive approach is successful in the real world, set point tracking is demonstrated on a first-order bench-scale laboratory temperature control process. As shown in Fig. 9, the process consists of two mixing streams of hot and cold water. The combined stream enters a stirred tank. The control objective is to maintain the temperature of the water in the tank by manipulating the flow rate of the cold water stream using an EMF to pressure transmitter and a pneumatic valve. The water temperature in the tank is measured using a type J thermocouple which is cold junction compensated for using a temperature transducer in the signal amplifier. The signal amplifier is used to amplify the thermocouple voltage. Common to real processes, there is a significant amount of measurement noise with a standard deviation of approximately 0.033°C. A 486 33 MHz personal computer is used to implement the control algorithms.

Although there is no mismatch between the first-order process and the order of the FOPDT model used in controller design, there is mismatch between the first-order process and the SOPDT model used to generate the VQNs' training patterns. This process also displays nonstationarity, slight nonlinearity and hysteresis.

The adaptive challenge presented by this process begins with an underestimate of the process gain, $K_p$. An estimate of approximately one-half the true value of the process gain, $K_p = 0.12$, is used with a correct estimate of the process time constant, $\tau_p = 700$ s. A DMC algorithm is employed using $\Gamma = 10$ and $\Delta t = 0.04\tau_p$. Two set point steps are made in a square wave at 375 sample intervals between the start up temperature and 2°C higher. These give the adaptive method the opportunity to converge. Three-hundred and fifty samples after the second set point step, approximately 12.5 samples of dead time are added by rerouting the combined flow through a long pipe. Two more set point steps are then made in the same square wave.

**Fixed model DMC control**

A fixed model DMC algorithm is used to illustrate the adaptive challenge presented by this demonstration. Figure 10 shows that the incorrect estimate of $K_p$
Fig. 10. Fixed model DMC control of laboratory process demonstration.

Fig. 11. Adaptive DMC control of laboratory process demonstration.

results in an aggressive response to the first set point step at sample 375 marked by an excessive 40% overshoot. The slightly less aggressive response to the second set point step at sample 750 illustrates the nonlinearity of the process. At the third set point step at sample 1125, the increased dead time causes loss of control. A limit cycle about the set point develops and continues for the remainder of the set point trajectory.

Adaptive DMC control

Figure 11 shows the result of employing the EDT for passive adaptive DMC control. The adaptive method performs three model regressions during the response to the first set point step made at sample 375. At the second model regression, 55 samples after the first set point step, the adaptive method is able to correct for the initial underestimation of the model gain as the model parameters converge. Desired performance is then obtained as witnessed by the response of the second set point step made at sample 750. The model time constant and dead time are not significantly changed. Further, model regressions at the second set point step do not significantly change the implemented model parameters.

As a result of the previous adaptations, control stability is not lost at the third set point step made at sample 1125. However, the increased process dead time causes aggressive controller behavior resulting in excessive overshoot. Three model regressions are performed during this response. The second model regression at sample 1185 produces a converged estimate of the dead time equal to 13.6 samples, which is slightly high, and the response quickly damps. The implemented estimates of the process gain and time constant are not significantly changed. A third model regression at sample 1210 produces an estimate for $t_d$ of 12.3 samples. The results of the adaptations made during the third set point step response are witnessed by the close to desired response to the fourth set point step made at sample 1500.

CONCLUSION

A pattern-based EDT has been developed to determine when sufficient process excitation exist for passive process model updating. The scope of this work is passive model-based adaptive control of SISO, open-loop stable, minimum phase, nonintegrating processes. The method focuses on the on-line diagnosis of closed-loop process data using VQNs. A simple rule base system utilizes the diagnoses of the VQNs and determines when sufficient dynamics exist for regression of a process model. If the resulting model parameters are deemed to be valid and converged, then the new model is used to design and update a model-based controller. This method is general and is applicable to any one of a number of model-based controllers and appropriate process model forms.

Unlike existing excitation diagnostic methods, the work presented here is shown to require minimal a priori process information. This is demonstrated through both simulations and an actual laboratory process where the nonideal process characteristics of nonlinearity, nonstationarity, significant measurement noise and process model order mismatch are considered. Through these demonstrations the method is shown to be stable, simple to used and reasonably robust in a variety of applications. All that is required of an application is that it should not be persistently excited so that the data batch for regression will not grow excessively large. Future work involves the ability to handle such non-steady as well as nonminimum phase and integrating processes.

Acknowledgements—Acknowledgement is gratefully made to the National Science Foundation through Grant CTS-9008596 and the University of Connecticut Precision Manufacturing Center for support of this work.

NOTATION

- $a_i$: model parameter associated with process output
- $b_i$: model parameter associated with process input
- $B$: DMC dynamic matrix
- $C$: DMC control horizon
- $c_i(t)$: predicted PI controller error at sample $t$
- $i$: normalized network input pattern vector
- $I$: normalized network input pattern vector
- $J$: DMC objective function
- $k$: process dead time expressed in integer number of samples
- $K_c$: PI controller gain
steady-state process gain

$K_p$ dimension of network input pattern vector

$N$ filtering parameter used in determination of $P_j$

$P$ DMC prediction horizon

$P_j$ filtered measurement-to-measurement deviation used to estimate $\sigma$

$Q$ sample time

$\Delta t$ DMC input suppression factor

$t$ time expressed as integer number of samples

$\tau_d$ process dead time

$T_j$ network matching score comparing incoming pattern to exemplar of node $j$

$S/N_j$ signal-to-noise ratio

$u(t)$ process input, manipulated variable, at sample $t$

$\Delta u$ DMC vector of future control actions

$\gamma(t)$ DMC vector of predicted process outputs

$\gamma_j(t)$ model estimate of true process output at sample $t$

$y_{sp}(t)$ controller set point at sample $t$

$y_{sp}$ DMC vector of future set points

$z_j$ network exemplar pattern of node $j$

$Z_j$ normalized network exemplar pattern of node $j$

Greek letters

$\beta$ network learning parameter

$\Gamma$ DMC tuning parameter

$\delta_j$ difference factor used for parameter convergence

$\eta$ normalization operator used in VQN

$\rho$ network vigilance parameter

$\sigma$ standard deviation of the process measurement noise

$\hat{\sigma}$ estimate of the standard deviation of the process measurement noise

$\tau_c$ IMC closed loop time constant

$\tau_i$ PI controller integral time constant

$\tau_p$ process time constant

REFERENCES


Cutler, C. R. and Ramaker, B. L., 1979, Dynamic matrix control—a computer control algorithm. AIChE 36th Annual Meeting, Houston, TX.


