Unified Excitation and Performance Diagnostic Adaptive Control Framework

Ralph F. Hinde, Jr. and Douglas J. Cooper
Dept. of Chemical Engineering, University of Connecticut, U-222, Storrs, CT 06269

Model-based controllers contain two elements that must be adjusted to maintain desired performance: parameters of the process model and a tuning parameter in the controller design equation. A unified framework is presented where vector quantizing networks are used in pattern-based methods for diagnosing process excitation and controller performance. Excitation diagnostics identify sufficiently excited dynamic process data for model updating. Performance diagnostics analyze set point response data and determine appropriate updates to the tuning parameter. Supervisory adaptation logic enables these two adaptive mechanisms to work together to maintain model accuracy and desired controller performance. The method is general to a number of model-based control algorithms and process model forms. Demonstrations employ a FOPDT model, as well as both the PI and DMC algorithms for set point tracking and disturbance rejection in a simulation and a bench-scale process.

Introduction

Model-based adaptive controllers update a process model internal to the controller architecture to maintain desired controller performance despite process nonlinearity and nonstationarity. Most model-based design equations such as internal model control (IMC) and dynamic matrix control (DMC) supply the operator with at least one tuning parameter that may be varied to produce desired controller performance (such as the IMC closed-loop time constant and the DMC input suppression factor). This tuning parameter is usually set during controller startup and held constant thereafter in hopes that model updating alone will maintain desired performance. Unmodeled dynamics, however, can cause controller performance to degrade regardless of model updating. In such instances, the controller tuning parameter must be regularly updated to maintain desired performance.

As such, model-based control algorithms actually contain two elements that must be made adaptive, the process model and the tuning parameter of the model-based controller design equation. Closed-loop model updating requires process data that is sufficiently rich in dynamic process input/output information (Gustavsson, 1977). An excitation diagnostic method is therefore required to determine when sufficiently dynamic process data exists. The manner in which a controller is performing is reflected in patterns displayed in the recent history of the controller error after set point changes (Bristol, 1977). A performance diagnostic method is therefore required to analyze controller error patterns and determine appropriate updates to the design equation tuning parameter.

This work presents pattern-based methods of process excitation diagnostics and controller performance diagnostics in a unified model-based adaptive control framework. These diagnostic methods are developed using vector quantizing networks (VQNs) to perform the pattern recognition (Kohonen, 1982; Carpenter and Grossberg, 1987; Carpenter et al., 1991). As shown in Figure 1, excitation diagnostics analyze patterns in the recent histories of the process input and output variables to determine when sufficient dynamics exist for model updating. Controller performance diagnostics analyze patterns in the controller error history to determine appropriate updates to the design equation tuning parameter. A supervisory adaptation logic then combines these adaptive efforts in a manner that allows them to work together to ensure that model accuracy and desired controller performance are maintained.

The unified model-based adaptive control framework is designed to be general to a number of model-based control algorithms employing a range of process model forms. To

Correspondence concerning this article should be addressed to D. J. Cooper.

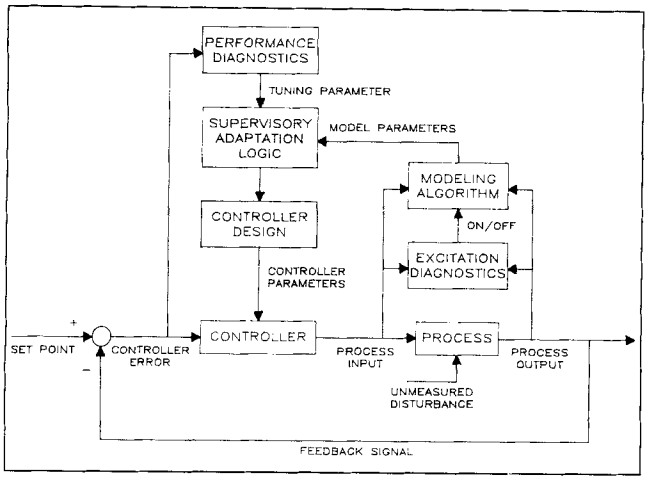

Figure 1. Unified adaptive framework.

illustrate this, both the DMC and PI (proportional integral) with predictor algorithms are used in both simulated and experimental demonstrations that exhibit nonlinearity, nonstationarity (time variance) and significant measurement noise. To facilitate discussion in this work, a simple first-order plus dead time (FOPDT) model form is employed in the demonstrations. The scope of this work is model-based control of single-input/single-output (SISO), open-loop stable, nonintegrating, minimum phase processes.

Vector quantizing networks are employed as the basis of the excitation and performance diagnostic methods. Therefore, the next section presents an overview on VQN design and implementation. This is followed by the details of excitation diagnostics, performance diagnostics and the supervisory adaptation logic. Control algorithms are then summarized and employed in demonstrations of this unified model-based adaptive control framework.

Vector Quantizing Networks

Vector quantizing neural networks (VQNs) are discrete pattern classifiers that implement a vector quantizing algorithm (Gray, 1984). They compare an incoming pattern to a library of exemplar patterns, determine to which exemplar pattern the incoming pattern is most similar and assign the incoming pattern to that exemplar pattern class. Examples of VQNs are Kohonen's network (1982) and the ART family of networks of Carpenter et al. (1987, 1991).

Like backpropagation networks (BPNs) (Rumelhart and McClelland, 1986), VQNs consist of many processing units called nodes. These nodes take a weighted sum of their inputs and pass the sum through a functionality to produce their output. Unlike a BPN where the nodes are arranged in layers receiving inputs from the previous layer and sending their outputs to the next layer, every node in a VQN receives the entire incoming pattern and produces an output of the network. Furthermore, whereas the nodes in a BPN make a collective effort to learn a functional relationship, each node in a VQN makes an individual effort to learn to classify a discrete pattern class. The more nodes a VQN has, the more pattern classes it can recognize. As detailed later, VQNs also possess the ability to assess their pattern recognition performance on-line, unlike a BPN. This makes VQNs inherently more desirable for pattern recognition purposes.

VQN architecture

An incoming pattern or series of sampled data for a VQN to either learn or classify is represented in vector form as:

$$i^T = [i(t-M+1)\, i(t-M+2) \ldots i(t)] \tag{1}$$

where M is the number of data elements or dimension of the vector and t is the sample number. This incoming pattern is compared to every exemplar pattern in the library and matching scores are generated for each comparison. The library of exemplar patterns is stored as the weights on the inputs to the nodes and is represented in vector form as:

$$z_j^T = [z_j(1)\, z_j(2) \ldots z_j(M)] \tag{2}$$

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

$$\eta x = \frac{x}{\|x\|} \tag{3}$$

and $\|x\|$ denotes the norm or Euclidean distance of vector $\boldsymbol{x}$. Hence, the operator η performs Euclidean normalization.

The matching scores, T_j, are generated as each node takes a weighted sum of its inputs. This is identical to the inner product of the two vectors:

$$T_j = I^T Z_j \tag{4}$$

The matching scores are measures of similarity between their respective nodes' exemplar patterns and the incoming pattern. Due to the Euclidean normalization of the vectors, a perfect match has a score of 1.0. Similarly, the matching score between a pattern and its mirror image is -1.0. Hence, all matching scores are within the range $-1.0 \le T_j \le 1.0$.

The matching scores are then used to determine which node's exemplar pattern wins the classification using the choice function:

$$T_J = \max_j |T_j| \tag{5}$$

where the absolute value of T_j allows classification of mirror image patterns effectively doubling the number of patterns the network is able to classify. Exemplar pattern Z_J may not win the classification, though, if T_J is small representing a weak match. Such would be the case if I were unlike anything the network had seen before. This is determined using a vigilance test where T_J is compared to a vigilance parameter ρ. If $T_J \ge \rho$, the vigilance is passed and Z_J wins the classification. If $T_J < \rho$, the vigilance test is failed and no classification is made. The matching scores and vigilance test provide the inherent self performance assessment of VQNs that makes them well suited for pattern recognition tasks.

VQN training

When performing on-line classification, VQN operation ends with the result of the vigilance test and the network is ready to receive another incoming pattern. During VQN training, however, network operation continues as long as the vigilance test is passed or there are available empty nodes. If the vigilance test is passed, the incoming pattern is added to that exemplar pattern. In this manner, similar patterns are "clustered" together. This is done using:

$$Z_J = \eta(\beta I + (1-\beta)Z_J) \tag{6}$$

where the learning parameter is defined as $0 \le \beta \le 1$. A smaller value for β causes slower learning but produces exemplar patterns that better represent the training patterns that were clustered together to form them. If the vigilance test is failed, no exemplar pattern is sufficiently similar to the incoming pattern so a new exemplar pattern must be made. If there are available empty nodes, the new exemplar pattern is created by setting the weights of an empty node equal to the elements of the incoming pattern.

During training, the network's adjustable parameters are β and ρ. Again, β sets the balance between the speed of learning and the quality of the exemplar patterns. The vigilance parameter, ρ, sets the balance between the number of exemplar patterns and the computational load of the VQN. A small ρ allows less similarity among patterns in a cluster producing fewer exemplar patterns and a sparsely partitioned pattern space. With fewer exemplar patterns, classification is more coarse but the computational cost of the network is lower. A large ρ requires more similarity among patterns in a cluster which produces more exemplar patterns and a more finely partitioned pattern space. More exemplar patterns allows finer classification but causes a larger computational cost for the VQN. In this work, $\beta = 0.10$ initially and decreases linearly with time. As used by Carpenter et al. (1991), $\rho = 0.92$ provides a good balance between classification performance and computational cost.

The patterns used to train a VQN must be both representative and comprehensive. A VQN can only classify an incoming pattern if it is similar to the exemplar patterns. The exemplar patterns must then be created by training patterns that are representative of the incoming patterns. If not, the vigilance test will never be passed and no classification will be made. Also, since clustering the training patterns to create the exemplar patterns effectively partitions the pattern space, the training patterns must span the entire pattern space and be fairly well distributed. Hence, the greatest amount of expertise required to train a VQN involves the development of the training patterns.

In this work, the VQNs analyze the recent history of the process input and output variables and the controller error. Patterns displayed in the process output and controller error tend to reflect the character of the process, so a process model form must be specified. The dynamics of most chemical processes are adequately described using a second-order plus dead time (SOPDT) model for stable control and pattern recognition purposes. Hence, a SOPDT model is used in training pattern development. The SOPDT model is realized as two first-order plus dead time (FOPDT) models in series. Note that if a specific application contains dynamics that are not adequately described by a SOPDT model, then either these dynamics or an entirely different and more appropriate model should be used to develop the training patterns.

Since patterns displayed in the process input tend to reflect the character of the controller, a model-based controller must be specified for VQN training. The operator specified model-based controller is then implemented on the SOPDT model. While a series of set point changes is made, the parameters of either the model or the model-based design equation are varied in such a manner as to produce the various patterns that are expected to be encountered. Since excitation diagnostics and performance diagnostics analyze different patterns, training pattern development is detailed later for each specific diagnostic method.

VQN training ends when no new exemplar patterns are created over one full cycle of training patterns. At this point, the VQN is able to recognize all training patterns. This is confirmed by noting that the vigilance test is passed for every training pattern in the full cycle. If all available nodes are filled during training, additional nodes should be made available or a smaller value of the vigilance parameter should be employed. The network should be allowed to develop as many exemplar patterns as necessary for given values of β and ρ so as not to distort the effect those parameters have on VQN pattern classification.

Excitation Diagnostics

On-line process modeling requires data rich in process input/output dynamic information that is not masked by measurement noise or corrupted by unmeasured disturbances (Gustavsson, 1977). Such dynamic data may be produced by intentionally perturbing the system, but such upsets are usually undesirable or intolerable in a production facility. This work explores the use of passive adaptation which performs model updating using naturally occurring dynamic events rather than by perturbing the system to create dynamic events. Hence, a method is required for determining when sufficiently dynamic events occur. This excitation diagnostic method will in essence serve as an on/off switch for the modeling algorithm.

This work employs an excitation diagnostic method that is robust in light of measurement noise variations and requires minimal *a priori* process information. As shown in Figure 2, two vector quantizing networks (VQNs) are employed as the basis of a pattern-based approach to excitation diagnostics. The VQNs analyze the recent history of both process variables and look for dynamic trend patterns. A decision-maker, which is a simple set of rules, then looks for many dynamic trend classifications in both process variables before declaring a global dynamic state and activating the modeling algorithm.

The VQNs are trained on dynamic patterns created by implementing an operator specified model-based controller on the SOPDT model. The model parameters are then cycled while set point step changes of random magnitude and duration are made. The model gain, K_p, is cycled between one and three times its initial value. The model time constants, τ_1 and τ_2, are cycled between one-third and three times their initial values. The model dead time, t_d, is cycled between zero and one-half the initial value of the dominant process time constant. This creates patterns displaying the full range from slightly sluggish

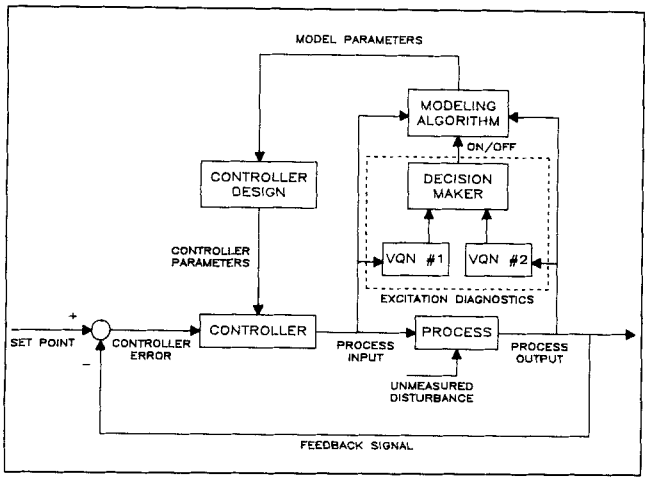

Figure 2. Excitation diagnostics.

to aggressive performance. Process variable histories that are one dominant process time constant in duration are collected and used to train the VQNs.

Once the VQNs are trained and put on-line, process variable dynamic states are diagnosed by exploiting the matching scores and vigilance test of the VQN architecture. Since the training patterns represent dynamic trends, the exemplar patterns that they are clustered to form also represent dynamic trends. If a network's incoming pattern does not contain a dynamic trend, then that pattern will not be similar to any of the dynamic exemplars and the largest matching score, T_J, will not pass the vigilance test. Similarly, a dynamic incoming pattern whose process information is masked by measurement noise will be less similar to the dynamic exemplar patterns. As the signal-to-noise ratio decreases, T_J will decrease and eventually fail the vigilance test. Hence, the matching scores and vigilance test are used to determine when a process variable history pattern contains dynamic information that is not masked by measurement noise.

The decision-maker looks for many dynamic trend classifications in both process variables before activating the modeling algorithm. This is done by keeping running sums of the vigilance test results in both VQNs. These running sums are S_u and S_y for the input and output variables respectively. For example, the input dynamic sum, S_u, is determined as:

$$S_u = S_u + 1, \quad \text{if } T_J \geq \rho$$

or (7)

$$S_u = S_u - 1, \quad \text{if } T_J < \rho$$

where zero is the minimum sum. The output dynamic sum, S_y, is similarly incremented. When a process variable's dynamic sum reaches a trigger value, S_{trig}, the process variable is declared to be in a dynamic state and its dynamic sum is returned to zero. A trigger value represents an initial dominant process time constant's duration. When both process variables are declared to be dynamic at the same time, a global dynamic state is declared and the modeling algorithm is activated. This modeling algorithm may consist of an RLS algorithm, a batch-wise regression of data or any combination of the two.

When the modeling algorithm is activated, data is collected from the present back to the last steady state. That batch of data is then operated on by the modeling algorithm. Steady states are declared when either both dynamic sums are zero for a trigger value's duration, which is a dominant process time constant or when one dynamic sum is zero for an estimated response time, t_{resp}. In either case, neither process variable may be saturated as this will only produce an apparent steady state and not disclose the true state of the process.

Using these rules, it is not possible to distinguish between sufficiently dynamic data and process data that is corrupted by an unmeasured disturbance. Therefore, an additional diagnostic is necessary after the modeling algorithm has operated on the batch of dynamic data. This consists of checking the sign of the new gain estimate against the sign of the presently implemented gain estimate. If the signs are opposite, the dynamics are declared to be disturbance driven and further dynamics are ignored until the next steady state.

If the signs of the two gain estimates are the same, the new model is deemed valid and is checked for convergence before it is implemented. There are many methods available for doing this ranging from simple percent difference criteria to statistical approaches. A final model update is performed just prior to the first steady state after the dynamic event to promote better convergence of the process model. Once the process model parameters are determined to be converged they may be directly implemented in the controller design equations to update the controller.

Note that this technique requires little *a priori* process information. All that is required is an initial estimate of a dominant process time constant and an estimated response time, t_{resp}, that are available in most process model forms popular in control theory. This dominant process time constant is used to determine the history pattern length, M, and the minimum duration of dynamics for model updating, S_{trig}. The response time is used with the dominant process time constant as minimum periods of nondynamics for identifying steady states.

Excitation diagnostics demonstration

Using this technique, a set point step response, for example, may be modeled three or four times during its duration. Each modeling instance will contain more information than the last until a steady state is found. This is demonstrated in Figure 3 where excitation diagnostics are shown identifying dynamics and initiating process model updating. Here, a long-range model predictive controller has been implemented on a simulated second-order process. Random error is added to the process output to simulate measurement noise. Figures 3a and 3b show both the process variable and dynamic sum trajectories for the process output and process input respectively. In this implementation, $S_{trig} = 5$ and $M = 25$ since the sample rate is 25 times per dominant process time constant and VQN analysis of process variable history patterns is performed once every five samples.

In the beginning of this demonstration, the process is at steady state. As such, the process output VQN correctly finds no dynamic trends in the recent output history patterns, and S_y is not incremented above zero prior to sample 100. A few

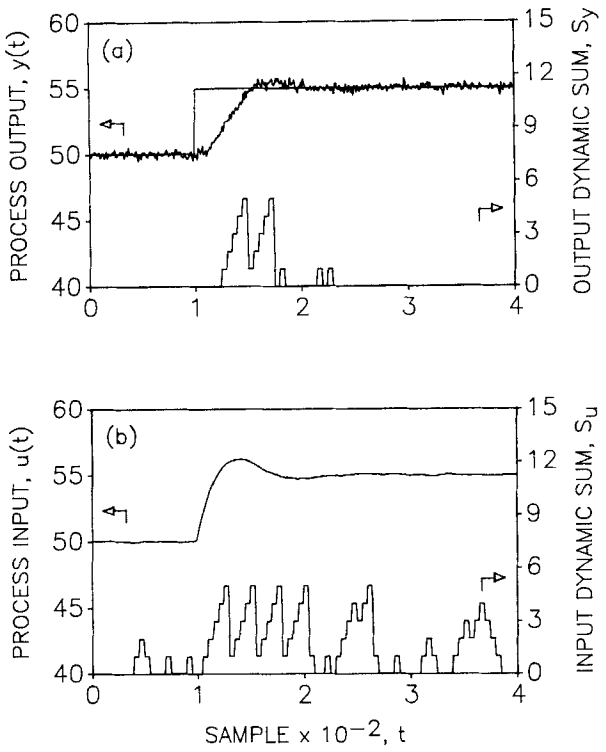

Figure 3. Identifying excitation in process (a) output and (b) input.

dynamic classifications are made by the process input VQN during the steady state, but the trigger value is not reached ($S_u < S_{trig}$) so a dynamic state is not declared. Such apparent input dynamic trends tend to occur more often in highly structured long-range predictive controllers. A steady state is found at sample 85.

At sample 100, the set point is stepped from 50 to 55 and both process variables begin to respond accordingly. Although S_u reaches the trigger value four times in succession, S_y only reaches the trigger value twice before the dampened response becomes masked by measurement noise. As such, model updating is performed twice during the corresponding dynamic states at samples 145 and 170. A last model update is performed at sample 350 just prior to the next steady state declared when $S_y = 0$ for an estimated response time regardless of dynamics found in the process input. In each modeling instance, data is collected from that sample back to sample 85. Hence, the first model update employs 60 samples, the second employs 85 samples, and the last employs 265 samples.

Performance Diagnostics

Model adaptation alone is insufficient in the presence of nonlinear and nonstationary unmodeled dynamics. Regardless of model updating, some process dynamics often remain unmodeled due to plant-model mismatch. The controller performance then degrades as the unmodeled portion of the process character changes. This necessitates periodic updating of the tuning parameter in the model-based controller design equations. To make a model-based controller fully adaptive, a method is required of evaluating controller performance and

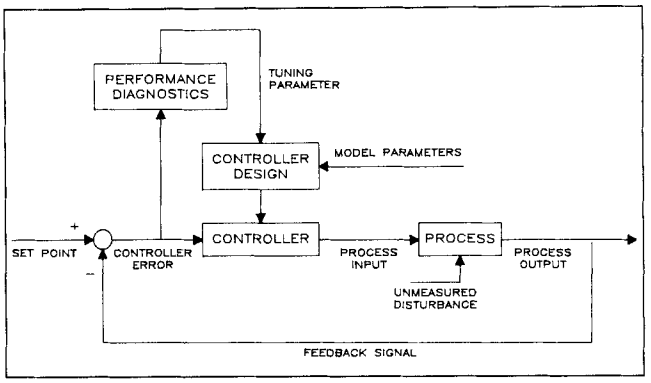

Figure 4. Performance diagnostics.

determining tuning parameter updates that maintain desired controller performance.

This work employs a performance diagnostic method, shown in Figure 4, that uses a VQN to analyze patterns displayed in the controller error history to determine the performance of the controller. Adjustments are then made in the model-based controller design equation tuning parameter based on observed performance. This is combined with process model parameters to update the character of the model-based controller.

The VQN is trained to recognize response patterns by implementing an operator specified model-based control algorithm on the SOPDT model (remember that the dynamics of most chemical processes are adequately described by this model). A value for the design equation tuning parameter that produces desired performance is determined and is generally referred to as Γ in this work. The tuning parameter is then varied above and below its desired value, Γ_{des}, by multiplying it by an adjustment factor δ_Γ. An appropriate range for δ_Γ depends on the controller performance range expected during on-line operation. Hence, the adjustment factor's range will depend on how sensitive controller performance is to its variation.

For each $\delta_\Gamma \Gamma_{des}$, a sustained set point step change is made and the resulting controller error response pattern is collected. Response patterns are one estimated response time, t_{resp}, in duration. Each response pattern and associated adjustment factor are then sent to the network for training. The VQN clusters similar response patterns to form the performance exemplar patterns. The associated adjustment factors are also clustered in a similar manner using:

$$\delta_\Gamma(J) = \beta\delta_\Gamma + (1-\beta)\delta_\Gamma(J) \tag{8}$$

where β is the VQN learning parameter and $\delta_\Gamma(J)$ is the adjustment factor associated with the performance exemplar pattern of node J.

Once the VQN is put on-line, controller error response patterns are collected for one t_{resp} after sustained set point step changes. To facilitate successful pattern recognition, performance diagnostics are abandoned if the set point changes again before one t_{resp} has expired. If an incoming response pattern is successfully classified (the vigilance test is passed), the winning exemplar pattern's adjustment factor is then used to update the tuning parameter using:

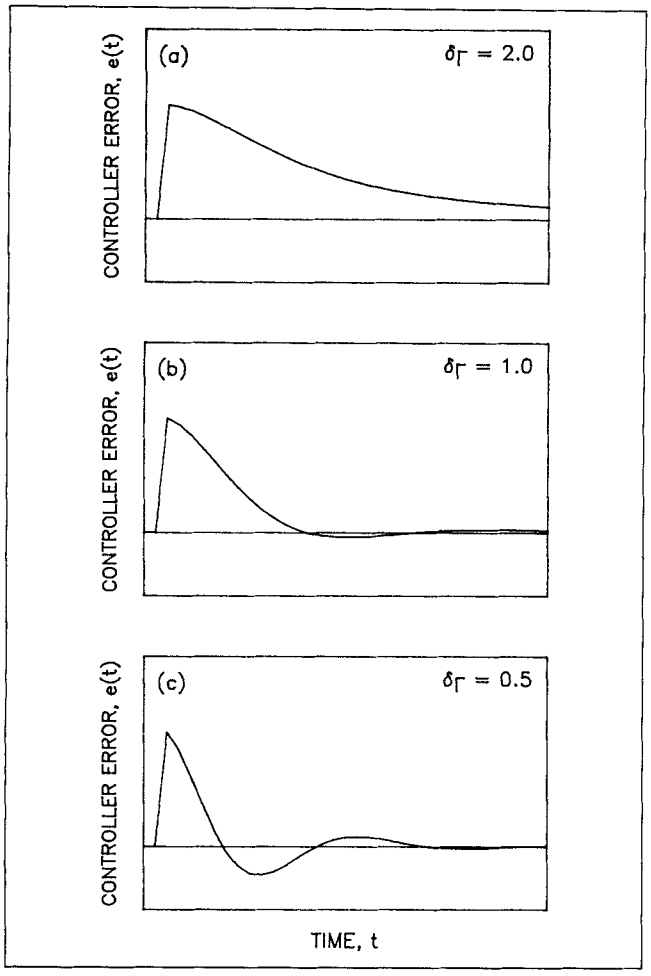

Figure 5. Example response patterns and associated adjustment factors.

$$\Gamma_{new} = \delta_\Gamma^{-1}\Gamma_{old} \tag{9}$$

In this manner, if an incoming response pattern is found to be similar to a performance exemplar pattern with an associated adjustment factor of 2.0, then the present value of the tuning parameter is divided by 2.0.

Note that this technique requires little *a priori* process information. All that is required is an initial estimate of a complete response time, t_{resp}, that is a multiple of the dominant process time constant available in most process model forms popular in control theory. This estimated response time is used to determine the length of the response pattern collected after a set point step change.

Figure 5 shows three example response patterns used to train the network and their associated adjustment factors, δ_Γ. These patterns are created using an IMC tuned PI controller with predictor (described later) implemented on the SOPDT process. The tuning parameter, Γ, is used to determine τ_c, the closed-loop time constant in the IMC design equations.

Figure 5a shows a sluggish response pattern created using an accurate process model and an adjustment factor of 2.0. If an examplar pattern similar to this were to win the classification of an incoming response pattern, the closed-loop time constant would be multiplied by 0.5 to obtained desired performance. Likewise, Figures 5b and 5c show desired and aggressive response patterns, respectively, with their associated adjustment factors of 1.0 and 0.5.

Again, a specific range of δ_Γ may not be specified for all design equation tuning parameters. Instead, the range of adjustments must be determined as that which produces a given range of controller performance. In this work, the performance range used for VQN training is extremely sluggish to steady oscillations on the verge of instability.

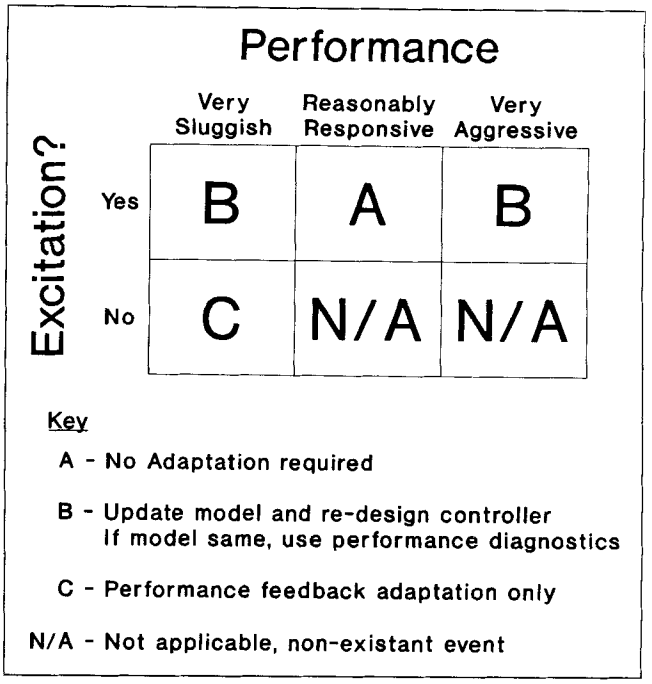

Figure 6. Supervisory adaptation logic for unified adaptive framework.

Supervisory Adaptation Logic

Whereas these two diagnostic methods are individually effective and powerful, this work focuses on how these two methods can work together in a unified model-based adaptive control framework. As shown in Figure 1, the excitation diagnostic method identifies sufficiently dynamic data used to keep an accurate process model. The performance diagnostic method updates the design equation tuning parameter to maintain desired performance. These two adaptive mechanisms, however, may interact and fight each other without the guidance of a supervisory adaptation logic.

The supervisory adaptation logic for the unified adaptive framework is shown in Figure 6. Here, in a tabular format, the adaptive action that is taken based on performance and excitation diagnostics is shown. Three columns for performance diagnostics are given for the three cases of "very sluggish," "reasonably responsive," and "very aggressive" performance. Two rows for excitation diagnostics are given for the two cases of "yes" and "no" regarding whether sufficient excitation exists for process model updating.

Of the six combinations of diagnostic cases, two are not applicable (N/A) and nonexistent. These are the cases of "reasonably responsive" and "very aggressive" performance in

combination with "no" sufficient dynamics. Under the definitions of this work, it is impossible to recognize such oscillatory performance without those oscillations providing sufficient dynamics. Likewise, insufficient dynamics due to excessive measurement noise would also mean that the performance would be masked and not identifiable. As such, the combination of these cases are not encountered.

The remaining four combinations of diagnostic cases lead to three different adaptive events. The "A" event is the result of "reasonably responsive" performance combined with sufficient dynamics. In this event, no adaptation is really necessary since nearly desired performance already exists. To track gradual changes in process character, though, model updating is still performed and used to redesign the model-based controller. Small tuning parameter adaptations are also allowed to fine-tune the controller performance.

The "B" event is the result of either "very sluggish" or "very aggressive" performance combined with sufficient excitation. In this event, it must be determined if the poor performance is a result of an inaccurate process model or an inappropriate value of the tuning parameter. This is determined by first updating the process model and then checking if the new model is significantly different from the present model. If the model parameters have not significantly changed, then the process character is declared to have not changed. As such, the poor performance may not be attributed to an inaccurate process model. The adaptation suggested by performance diagnostics is then implemented to obtain desired performance with the accurate process model. If the new model parameters are significantly different, then the poor performance is attributed to a poor process model. The new model is then implemented and the recommended performance-based adaptation is ignored. As with model parameter convergence, methods for determining significant parameter variation may range from simple percent difference criteria to statistical approaches.

The combination of "very sluggish" performance and "no" sufficient dynamics results in a "C" event. Here, no model updating may be performed due to the lack of sufficient dynamics. The only available adaptive action is to implement the tuning parameter update recommended by performance diagnostics in hopes that this will increase the controller effort enough to induce sufficient process dynamics for model updating.

Using this supervisory adaptation logic and unified adaptive framework allows excitation diagnostics and performance diagnostics to work together and help one another. Without such a logic table the two adaptive mechanisms may fight one another or overadapt the controller.

Control Algorithms

Process model

The form of the process model to be used in model-based controller design must adequately describe the character of the process. A poorly chosen model form that does not describe important process characteristics will lead to poor control. All that excitation diagnostics and performance diagnostics require of the process model, though, is a dominant process time constant and an estimated response time, t_{resp}. These are provided by most process model forms popular in control theory.

For demonstration purposes in this work, the first-order plus dead time (FOPDT) model is employed. The FOPDT model adequately describes the steady-state gain, K_p, dominant process time constant, τ_p, and dead time, t_d, of most open-loop stable chemical processes for controller design purposes. Although a 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. For instance, the dead time may be used to compensate for higher order lag and the inverse response portion of a nonminimum phase process. This is referred to as apparent dead time and should not be confused with the true dead time of process.

In difference form, the FOPDT model is expressed as:

$$y'(t) = y'(t-1) + a_1 \Delta y'(t-1) + b_k \Delta u(t-k) \qquad (10)$$

where $y'(t)$ is the prediction of the process output, $u(t)$ is the manipulated process input, and Δ is the difference operator. The linear difference parameters a_1 and b_k are determined using:

$$a_1 = \exp(-\Delta t/\tau_p) \quad b_k = K_p(1-a_1) \quad k = \text{int}\,(t_d/\Delta t) + 1 \qquad (11)$$

The estimated response time for a FOPDT process is determined as:

$$t_{resp} = 5\tau_p + t_d \qquad (12)$$

The sample time, Δt, is defined in this work as 0.04 times the initial estimate of τ_p. As process variable history patterns are one dominant process time constant in duration, this means that M, the history pattern length, is 25 samples. 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 dominant process time constant were employed. To reduce computational cost of the performance diagnostic method, response patterns that are typically 125 samples long are recast as 50 samples across the response time duration.

Note that the FOPDT model is used here for demonstration purposes only. The pattern-based methods employed here are designed to use any process model form that adequately describes the process to be controlled and contains a dominant process time constant and estimated response time.

Model-based control algorithms

Once a process model form is chosen, the parameters of that model may be used to design any one of a number of model-based control algorithms. Hence, designing the excitation and performance diagnostic methods to work with model-based controllers makes the unified adaptive framework widely applicable. In this work, a model-based PI controller with predictor for dead time compensation and the dynamic matrix control (DMC) algorithm are employed for demonstration purposes. These algorithms, well documented in the literature, are summarized here.

The velocity form of the PI algorithm (Smith and Corripio, 1985; Seborg et al., 1989) computes the incremental control action at sample number t, $\Delta u(t)$, as:

$$\Delta u(t) = K_c(e'(t) - e'(t-1) + e'(t)\Delta t/\tau_i) \quad (13)$$

where K_c is the controller gain and τ_i is the integral or reset time. To compensate for dead time, the controller error, $e'(t)$, is predicted using:

$$e'(t) = y_{sp}(t) - y'(t+k-1|t) - y(t) + y'(t|t-k+1) \quad (14)$$

where the future process output based on present information, $y'(t+k-1|t)$, is computed by iterating on Eq. 10 up to the most recent input manipulation at sample $t-1$. The present measurement and the prediction from a dead time ago are used to correct for model error and unmodeled dynamics.

In this work, the PI controller is tuned using the internal model control (IMC) architecture (Rivera et al., 1986). Since the t_d of our FOPDT model has been compensated for, only K_p and τ_p are employed with a first-order filter to arrive at the IMC tuning relations:

$$K_c = \frac{\tau_p}{\tau_c K_p} \quad \tau_i = \tau_p \quad (15)$$

where τ_c, 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 and the tuning parameter. In this work, $\tau_c = \Gamma\tau_p$ and is updated as τ_p changes.

The form of the DMC algorithm (Cutler and Ramaker, 1979; Prett and Garcia, 1988) employed here is the least-squares, unconstrained, single-input/single-output (SISO) implementation. At each sample t, future control actions are determined that minimize the objective function:

$$f = \sum_{i=1}^{P} (y_{sp}(t+i) - y'(t+i))^2 + Q \sum_{i=1}^{C} (\Delta u(t+i-1))^2 \quad (16)$$

where $y'(t+i)$ are the predicted process outputs and $y_{sp}(t+i)$ are the expected future set points assumed to equal the present set point value. The length of the prediction horizon, P, is set equal to an estimated response time and the length of the control horizon, C, is set equal to one dominant process time constant. Using a FOPDT model:

$$P = (5\tau_p + t_d)/\Delta t \quad C = \tau_p/\Delta t \quad (17)$$

The input suppression factor, Q, is set equal to ΓK_p^2. This serves to cast the two terms of Eq. 16 into consistent units. The parameter Γ is a tuning parameter chosen by the operator to provide desired performance.

A least-squares minimization of Eq. 16 over the control horizon, C, provides the solution of the future control actions:

$$\Delta \boldsymbol{u} = (\boldsymbol{B}^T\boldsymbol{B} + \boldsymbol{Q})^{-1}\boldsymbol{B}^T(\boldsymbol{y}_{sp} - \boldsymbol{p}) \quad (18)$$

where $\boldsymbol{Q}$ is a diagonal matrix of input suppression factors, $\boldsymbol{y}_{sp}$ is a vector of future set point values, $\boldsymbol{p}$ is a vector of future predicted outputs based solely on past control actions and $\boldsymbol{B}$ 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 in input. The first element of $\Delta \boldsymbol{u}$ 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 and the tuning parameter which are then used to update the dynamic matrix $\boldsymbol{B}$, 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, τ_p and t_d in this work, must be specified at startup. For all demonstrations, these parameters are based on the regression of data collected from a single open-loop step test made in the startup operating regime. This regression, and model updating in general in this work, is a batchwise regression of Eqs. 10 and 11 to process input and output data minimizing the sum of squared errors between the predicted and actual process output trajectories. Again, the sample time, Δt, is set equal to 0.04 times the initial estimate of τ_p.

The IMC and DMC tuning parameters are then set to values that produce desired controller performance defined in this work as a short rise time with 10% overshoot. For the processes employed in this work, such desired performance is initially obtained using $\Gamma = 0.5$ for an IMC tuned PI controller and $\Gamma = 150.0$ for a DMC algorithm. Note that different initial values of the tuning parameters may be required or desired for different controllers and other implementations. Also note that smaller values of Γ 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 Γ, as doing so may mask process excitation.

Adaptive Control Simulation

The unified adaptive framework is demonstrated using a simulated second-order process consisting of two FOPDT models in series. The gain and time constants of these models are all initially 1.0 and the overall true dead time is zero. Measurement noise is simulated by adding normally distributed random error with standard deviation of 0.10 to the process output. The use of a FOPDT model in controller design for this second-order process introduces the nonideality of model order mismatch. This mismatch of unmodeled higher order dynamics will necessitate tuning parameter adaptation, especially as the apparent dead time varies.

A DMC controller is employed in this demonstration and is therefore used in the development of the VQN training patterns. Using the DMC and VQN parameters discussed previously, excitation diagnostic input and output VQNs contain 73 and 29 dynamic exemplar patterns, respectively, after training. The performance diagnostic VQN contains 66 performance exemplar patterns with adjustment factors, σ_Γ, ranging from 0.002 to 500.0 and 150.0 as the desired tuning parameter, Γ_{des}.

Figure 7 shows adaptive DMC control of this simulated process. As shown in Figure 7, three sustained set point step changes of magnitude 5.0 are made in a square wave at 500

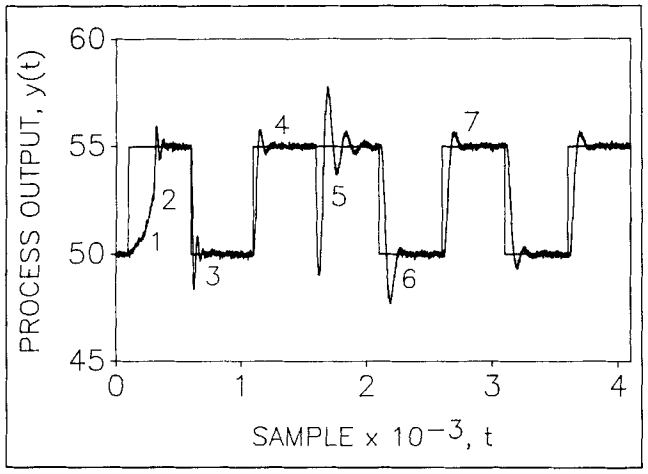

Figure 7. Adaptive control of simulated process.

sample intervals starting at sample 100. At sample 1,600, a step disturbance is introduced to the process that moves the process away from set point and adds 13 samples of dead time, approximately one-half τ_p, to the process response. At sample 2,100, four more set point step changes of magnitude 5.0 are made in the same square wave at 500 sample intervals. The unmodeled dynamics due to the model order mismatch become amplified with the addition of the large dead time. This makes

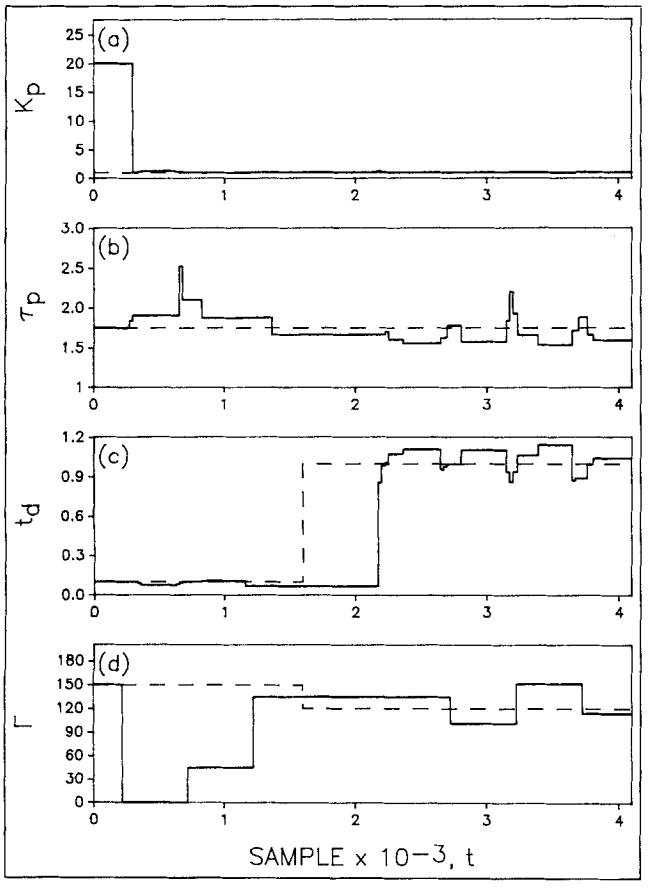

Figure 8. Parameter trajectories from adaptive control of simulated process.

tuning parameter adaptation necessary to maintain desired controller performance. The points of particular adaptive interest during this demonstration are numbered from 1 to 7 in Figure 7.

Figure 8 shows the model and controller parameter trajectories that result from adaptive DMC control of this simulated process. Figures 8a–8d plot the parameters K_p, τ_p, t_d and Γ, respectively, against sample number t as solid lines. The accurate or desired values of the parameters are plotted as dashed lines.

The challenge put forth to the unified adaptive framework begins early with an initially poor process model. The process gain estimate is overestimated by a factor of 20.0. This overestimation of K_p causes very sluggish performance at point 1 in Figure 7 seen after the first set point step made at sample 100. In fact, the closed-loop response is so sluggish that very little dynamic process information exists above the measurement noise. This response would cause most model-based adaptive control algorithms to fail since a poor model would result from the data.

The use of performance diagnostics and the supervisory adaptation logic, however, allows recovery from the sluggish performance. Point 1 is diagnosed as a C event where very sluggish performance exists with insufficient dynamics for process modeling. Performance-based adaptation is used to increase the effort of the controller and induce sufficient process dynamics. Figure 8d shows the tuning parameter, Γ, being reduced from 150 to 0.3 at sample 225.

At point 2, sample 300 in Figure 7, two successive model updates have provided a converged accurate estimate of the process gain. As shown in Figure 8a, the new model is implemented and the controller is updated through the design equations. This allows the controller to bring the process to set point in a somewhat aggressive manner. The other model parameters are not changed appreciably.

At sample 600, the second set point step results in aggressive performance now that the process model is accurate and the input suppression factor is much smaller than it should be. Performance diagnostics and excitation diagnostics determine this to be a B event in which very aggressive performance exists with sufficient dynamics for process model updating. Figure 8b shows that τ_p is significantly increased at sample 655. This, however, is only a result of the very aggressive performance. The next modeling instance recovers an accurate estimate of τ_p and the model parameters are determined to have ultimately not changed significantly. The poor performance is then rightfully determined to be due to an improper value of the tuning parameter and the δ_Γ recommended by performance diagnostics is implemented at sample 725 increasing Γ from 0.3 to 45.0.

The result of these previous adaptations is shown at point 4 in Figure 7 after the third set point step at sample 1,100. Whereas successful model and tuning parameter updating have produced performance that appears to be close to desired, there is actually a 20% overshoot and a B event is declared. Figures 8a–8c show that the model parameters do not significantly change during model updating so once again, the performance-based adaptation is implemented and Γ is increased from 45.0 to 135.0 at sample 1,225. The unified adaptive framework has now successfully recovered from severe controller sluggishness and regained accurate and desired values for K_p, τ_p, t_p and Γ.

Point 5 in Figure 7 shows the rejection of the disturbance that is made at sample 1,600. Excitation diagnostics activate the model updating algorithm at sample 1,630 but find that the sign on the new model gain estimate is negative and opposite of the sign of the presently implemented estimate. The dynamics are then correctly identified as being caused by an unmeasured disturbance. The new model is discarded and further dynamics are ignored until the disturbance rejection event is over at the next steady state at sample 2,055. Although there is much opportunity to lose the accurate process model gain and time constant during this disturbance rejection, the additional excitation diagnostics for disturbance dynamics prevent doing so. Unfortunately, there is no opportunity to track the increased dead time, as shown in Figure 8c.

Point 6 in Figure 7 shows the effect of the increased and unmodeled dead time on controller performance at the fourth set point step change at sample 2,100. This is another B event that displays excessively aggressive performance marked by a 50% overshoot. At sample 2,175, model updating converges and an accurate estimate of the dead time is implemented. As the estimate of t_d has significantly changed, the aggressive performance is declared to be due to the previously poor process model and the adjustment factor recommended by performance diagnostics is ignored.

The last three set point steps made at samples 2,600, 3,100 and 3,600, point 7 in Figure 7, are declared A events. The response patterns to these steps display reasonably responsive and close to desired performance. Figures 8a–8c show that model updating does not significantly change the model parameters. Figure 8d, however, shows that tuning parameter updating makes oscillatory convergent adaptations to Γ. In all, the unified adaptive framework has obtained and maintained desired controller performance despite an incorrect initial tuning and process nonstationarity.

Adaptive Control of a Heated Stirred Tank

The previous demonstration, although designed to display process nonidealities, is still a simulation and ideal in nature. To show that this unified adaptive framework is successful in the real world, set point tracking and disturbance rejection are demonstrated on a bench-scale laboratory heated stirred tank. As shown in Figure 9, this process consists of a cold water stream that enters a stirred tank and is heated by a coiled steam pipe. The volume of water in the tank is held constant using a side drawn outlet stream. The temperature of the volume in the tank is measured using a type J thermocouple and controlled by manipulating a pneumatic steam valve that regulates the pressure in the coiled pipe. Additional lag is added to the process character by shielding the tip of the thermocouple with a small piece of tubing. A 486, 33 MHz computer is employed as the controller and data acquisition system.

This process displays the typical real world nonidealities of measurement noise, nonlinearity, nonstationarity, and hysteresis. The nonstationarity offers the most challenge to the unified adaptive framework. This is shown in Figure 10 which is an open-loop, steady input implementation. The measured output, percent of tank temperature range, is shown to stray and wander from its initial value of 61.5% or 61.0°C. This is caused by unmeasured disturbances from sources such as inlet water temperature variation affecting the heat load and upstream pressure variation causing inlet flow rate fluctuations. These disturbances will distort the history and response patterns making the diagnosis of process excitation and controller performance more difficult.

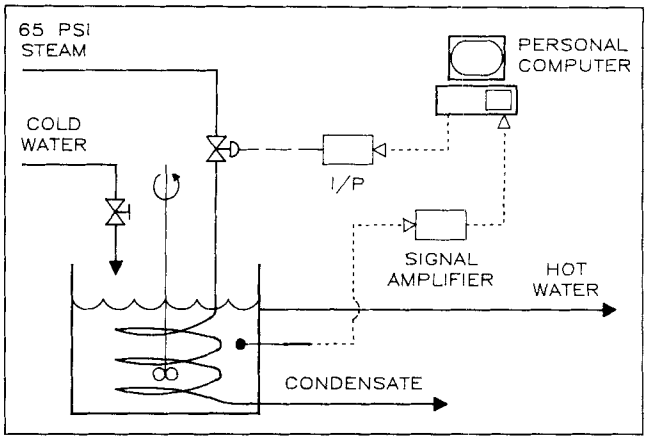

Figure 9. Heated stirred tank process.

An IMC tuned PI controller with predictor is employed in set point tracking and disturbance rejection for the heated stirred tank. As such, this controller is used with a simulated SOPDT process to retrain the VQNs. Using the previously specified VQN and PI parameters, the input and output excitation diagnostic VQNs each contain 358 and 77 dynamic exemplar patterns, respectively, and the performance diagnostic VQN contains 70 performance exemplar patterns. The adjustment factor, δ_Γ, ranges from 0.1 to 3.0 with 0.5 as the desired value, Γ_{des}. More dynamic exemplar patterns are required for the PI controller as opposed to DMC because the PI controller makes more sudden and drastic input manipulations than a DMC algorithm. This causes more variation in the process input pattern space.

As shown in Figure 11, four set point step changes of 5% range or approximately 4.2°C are manually made in a square-wave trajectory at approximately 150 sample intervals beginning at sample 100. At sample 700, a disturbance is made to the system by manually reducing the cold water inlet stream's valve position from 75% to 25% full range. Three more set

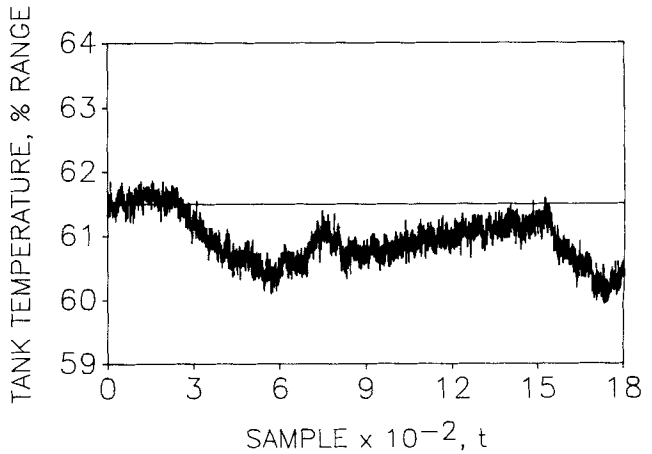

Figure 10. Adaptive challenge of heated stirred tank process.

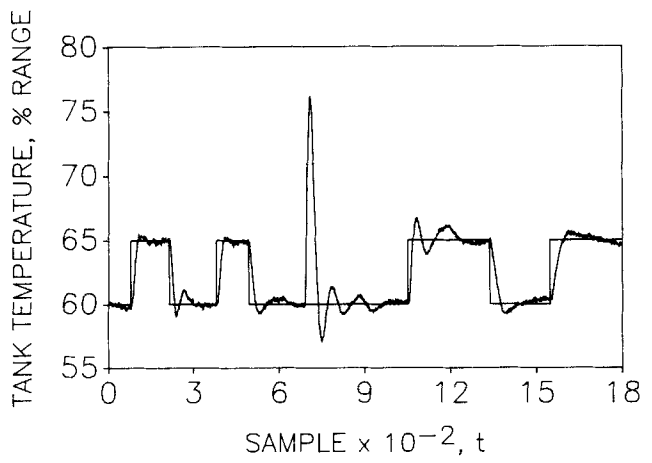

Figure 11. Adaptive control of heated stirred tank process.

point step changes are then made at samples 1,050, 1,340, and 1,550 to illustrate the effect that the disturbance has on controller performance and to provide the unified adaptive framework the opportunity to converge on desired performance.

Adaptation is turned off during the first two set point step changes in Figure 11 to show that reasonably responsive performance is produced using the step test FOPDT model parameters and $\Gamma = 0.5$. The nonlinearity and nonstationarity of this process cause the responses to these two set point changes to differ, especially in damping character. This illustrates the main adaptive challenge offered by this demonstration process.

The unified adaptive framework is implemented at sample 300. A first steady state is found at sample 375 and the excitation diagnostic method begins looking for sufficiently dynamic data to model. The response to the third set point step change made at sample 380 leads to three model updates that converge on an estimate of τ_p that is one-half the original estimate. Performance diagnostics are aborted as this parameter has significantly changed.

The result of this adaptive action is seen in the response to the fourth set point step change made at sample 495. This response is marked with a faster damping than that of the second set point step change. The model parameters are not significantly changed from model updates during this response so the performance-based adaption is implemented. Performance diagnostics find this response to be desirable and the recommended adjustment factor is 1.0. The unified adaptive framework has just demonstrated its ability to obtain and verify desired closed-loop controller performance despite an apparently inaccurate initial process model.

The disturbance rejection response begins at sample 690. The step disturbance moves the process to a lower operating regime and significantly changes the character of the process. The first modeling instance of this response produces an estimate of the process gain that has a negative sign. Controller stability would be lost if this new estimate were to be implemented. However, the new model fails the gain sign check and correctly indicates that this is a disturbance rejection event. The new model is rejected and further dynamics are ignored until the next steady state at sample 1,015, where the disturbance rejection event is determined to be over. Although the change in process character cannot be tracked, the unified adaptive framework is able to maintain meaningful process model parameters and controller stability.

The result of the unmodeled change in process character is seen in the response to the fifth set point step change at sample 1,050. This is an aggressive response marked by an excessive 35% overshoot and slow damping. The abundant dynamics produced by the slow damping lead to seven model regressions that converge on a new process model. The converged estimate of the process gain is approximately twice the predisturbance estimate. The other parameters are not changed appreciably. As the overall process model has changed, the poor performance is determined by the supervisory adaptation logic to be due to an inaccurate process model and the performance-based adaptation is ignored.

The response to the sixth set point step change at sample 1,340 displays closer to desired performance with a 15% overshoot. Model updating finds that the model parameters do not significantly change. The performance-based adaptation is then implemented and the IMC tuning parameter, τ_c, is reduced to 87% of its previous value. This is shown to be an appropriate adaptation by the response to the last set point step change made at sample 1,550. The 15% overshoot of the previous response is reduced to a 12% overshoot. The apparent inability of these two responses to converge on the set point is due to the nonstationarity of the process causing the process output to stray. This nonstationarity, however, does not significantly affect the adaptive mechanisms as model and controller parameter updates are accurate.

Conclusion

A unified model-based adaptive control framework has been presented that ensures both model accuracy and desired controller performance. This is accomplished through the use of vector quantizing neural networks (VQNs) as the building blocks for an excitation diagnostic method and a performance diagnostic method. Whereas these two methods are individually powerful, it is through their combination with a supervisory adaptation logic that the real power and success of the algorithm is realized. This adaptive approach is model-based making it widely applicable to any number of model-based controllers employing any number of process model forms. All that is required is that the model form provide excitation diagnostics with a dominant process time constant and both diagnostic methods with an estimated response time.

The effectiveness and wide applicability of the unified adaptive framework are demonstrated through both simulation and bench-scale application. These demonstrations show that the unified approach is able to obtain and maintain desired controller performance regardless of process nonlinearity and nonstationarity even when initial controller design does not use correct values for the model and tuning parameters.

Future work involves the extension and implementation of this unified adaptive framework as a piggyback module, as shown in Figure 12. Instead of adapting a single controller, excitation and performance diagnostics are used to adapt a second controller that is piggybacked to an existing controller. In this manner, the entire adaptive approach may be modularized and fitted to any existing controller. This promotes wider applicability and controller redundancy (backup controllers are generally desirable).

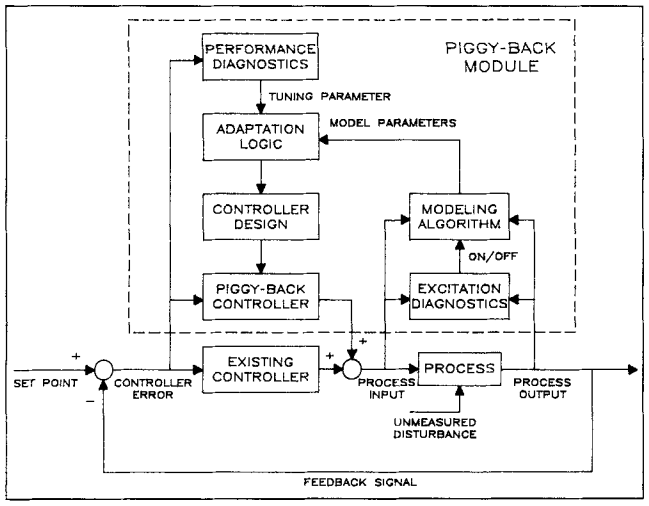

Figure 12. Piggyback module.

The piggyback methodology, a structured takeover of an existing controller, is envisioned as follows. First, a process model must be developed. Excitation diagnostics watch for sufficiently dynamic data and initiate model updating when appropriate. When the model converges, the piggyback module then designs the piggyback controller. Set point response data is collected and used to regress the piggyback controller's model-based design equation tuning parameter, Γ, so that it will mimic the existing controller.

Once the tuning parameter converges, the piggyback controller is put on-line in a manner that overrides the existing controller without replacing it. Since the tuning parameter has been regressed, the takeover is smooth and bumpless. The piggyback module then continues to update the piggyback controller using excitation diagnostics to identify opportunities for process model updating and performance diagnostics to update the design equation tuning parameter. The existing controller is left intact as a backup should anything happen to the piggyback module. Such an approach should make adaptive control easier to implement and more appealing to the end user.

Acknowledgment

Acknowledgment 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_1 = FOPDT model parameter for process output
b_k = FOPDT model parameter for process input
C = DMC control horizon
$e'(t)$ = PI predicted controller error
i = VQN incoming pattern vector
I = Euclidean normalization of vector i
k = integer number of samples in one dead time
K_c = PI controller gain
P = DMC prediction horizon
$y_{sp}(t)$ = set point, desired process output at sample t
z_j = exemplar pattern vector for VQN node j
Z_j = Euclidean normalization of vector z_j

Greek letters

Δ = difference operator
ρ = VQN vigilance parameter
τ_c = IMC closed-loop time constant
τ_i = PI reset time, integral time constant

Literature Cited

Bristol, E. H, "Pattern Recognition: An Alternative to Parameter Identification in Adaptive Control," *Automatica*, **13**, 197 (1977).
Carpenter, G. A., and S. Grossberg, "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Appl. Op.*, **26**(23), 4919 (1987).
Carpenter, G. A., S. Grossberg, and D. B. Rosen, "ART2-A: An Adaptive Resonance Algorithm for Rapid Category Learning and Recognition," *Neural Networks*, **4**(4), 493 (1991).
Cutler, C. R., and B. L. Ramaker, "Dynamic Matrix Control—A Computer Control Algorithm," AIChE Meeting, Houston (1979).
Gray, R. M., "Vector Quantization," *IEEE ASSP Mag.*, **4**, 4 (1984).
Gustavsson, I., L. Ljung, and T. Söderström, "Identification of Processes in Closed Loop—Identifiability and Accuracy Aspects—Survey Paper," *Automatica*, **13**, 59 (1977).
Kohonen, T., "Self-Organized Formation of Topologically Correct Feature Maps," *Biol. Cybern.*, **43**, 59 (1982).
Prett, D. M., and C. G. Garcia, *Fundamental Process Control*, Butterworths, Boston (1988).
Rivera, D. E., M. Morari, and S. Skogestad, "Internal Model Control: 4. PID Controller Design," *Ind. Eng. Chem. Process Des. Dev.*, **25**, 252 (1986).
Rumelhart, D. E., and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: 1. Foundations*, MIT Press, Cambridge (1986).
Seborg, D. E., T. F. Edgar, and D. A. Mellichamp, *Process Dynamics and Control*, Wiley, New York (1989).
Smith, C. A., and A. B. Corripio, *Principles and Practice of Automatic Process Control*, Wiley, New York (1985).

Manuscript received Aug. 13, 1993, and revision received Jan. 24, 1994.